ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός και Παρακολούθηση Χρηστών σε Περιβάλον Επικοινωνίας Ανθρώπου-Μηχανής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευστράτιος Χ. Χαρίτος

Επιβλέπων : Στέφανος Δ. Κόλλιας
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός και Παρακολούθηση Χρηστών σε Περιβάλον Επικοινωνίας Ανθρώπου-Μηχανής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ευστράτιος Χ. Χαρίτος

Επιβλέπων : Στέφανος Δ. Κόλλιας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8/5/2007.

.....	.....	.....
Στέφανος Κόλλιας	Ανδρέας-Γεώργιος Σταφυλοπάτης	Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2007

.....
ΕΥΣΤΡΑΤΙΟΣ Χ. ΧΑΡΙΤΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι ο εντοπισμός και η παρακολούθηση χρηστών σε περιβάλλον επικοινωνίας ανθρώπου-μηχανής. Συγκεκριμένα δημιουργήθηκε μια σειρά από προγράμματα χρησιμοποιώντας OpenCV (Open Computer Vision Library), τα οποία μπορούν σε ένα βίντεο η σε μια λήψη από webcamera, όπου θα υπάρχει σειρά ακολουθιών εικόνων ενός ανθρώπου που μιλάει και κινείται , να εντοπίζουν το κεφάλι και τα χέρια αλλά και το τι κινήσεις αυτά πραγματοποιούν κατά την διάρκεια του βίντεο. Η πλατφόρμα OpenCV που χρησιμοποιήθηκε είναι μια σειρά από έτοιμες συναρτήσεις και δομές γραμμένες σε C++ για την επεξεργασία εικόνων και βίντεο. Η εφαρμογή αυτή είναι χρήσιμη για την επικοινωνία ανθρώπου μηχανής μιας και επιτρέπει στον υπολογιστή να καταλαβαίνει ένα μέρος της γλώσσας του σώματος.

Λέξεις-Κλειδιά

OpenCV (Open Computer Vision Library), εντοπισμός κίνησης, επεξεργασία εικόνας, εντοπισμός δέρματος, αλληλεπίδραση ανθρώπου-μηχανής.

Abstract

This dissertation deals with analysis of users' movements in human computer interaction environments. In particular, a set of programs has been created using OpenCV (Open Computer Vision Library) , enabling the detection of the head and the arms of a human shown in a video or in a web camera video, as well as their movements. The OpenCV platform is a package of functions and structures written on C++ for achieving satisfactory image and video editing. The outcome of the work done is especially useful for man machine interaction, allowing the machine to understand a part of the human body language.

Key Words

OpenCV (Open Computer Vision Library), movement detection, image editing, skin detection, human-machine interaction

Π ε ρ ι ε χ ό μ ε ν α

Κ Ε Φ Α Λ Α Ι Ο 1

Εισαγωγή

1.1 Αντικείμενο της μελέτης

Η επικοινωνία ανθρώπου μηχανής ήταν μία από τις βασικές επιδιώξεις του ανθρώπου από την μέρα που δημιουργήθηκε ο πρώτος υπολογιστής. Με την βοήθεια της τεχνητής νοημοσύνης και των νευρωνικών δικτύων έχουν γίνει πολλές προσπάθειες ώστε να εξομειωθεί από τον υπολογιστή η ανθρώπινη συμπεριφορά. Το επόμενο βήμα ήταν η ικανοποιητική επικοινωνία του ανθρώπου με τους υπολογιστές, το να μπορεί δηλαδή να καταλάβει ο υπολογιστής τον άνθρωπο σε ικανοποιητικό βαθμό. Η επικοινωνία αυτή για να πραγματοποιηθεί χωρίζεται σε δύο βασικές διεργασίες, την κατανόηση της φωνής και την κατανόηση της γλώσσας του σώματος. Στην παρούσα εργασία ασχολούμαστε με την κατανόηση της γλώσσας του σώματος (χειρονομίες, κινήσεις του κεφαλιού, μορφασμοί του προσώπου, κινήσεις του σώματος, κ.λ.π.). Για να μπορέσει ο υπολογιστής να κατανοήσει την γλώσσα του σώματος πρέπει να μπορεί να επεξεργάζεται εικόνα και βίντεο ώστε ΄΄βλέποντας΄΄ των χρήστη να κατανοεί τις κινήσεις που αυτός κάνει. Μόλις ο υπολογιστής άρχισε να μπορεί να επεξεργάζεται εικόνες με αποτελεσματικό τρόπο, δημιουργήθηκαν πάρα πολλές εφαρμογές με σκοπό να εξομοιωθεί μέσω του υπολογιστή η λειτουργία του ανθρώπινου ματιού. Οι δυσκολίες ήταν πολλές, αλλά η συνεχής βελτίωση των υπολογιστικών συστημάτων σε συνδυασμό με την βελτίωση των μέσων λήψης εικόνων και βίντεο είχε σαν αποτέλεσμα να δημιουργηθεί μια πληθώρα εφαρμογών με σκοπό την κατανόηση της ανθρώπινης κίνησης και συμπεριφοράς μέσα από την επεξεργασία εικόνων και βίντεο. Το ανθρώπινο μάτι δεν έχει κατορθωθεί να αντικατασταθεί , αλλά βρισκόμαστε κοντά στην μέρα όπου ο υπολογιστής θα μπορεί να αντιλαμβάνεται ακριβώς ό,τι αντιλαμβάνεται και το ανθρώπινο μάτι. Η εφαρμογή που αναπτύχθηκε σε αυτήν την διπλωματική εργασία θα βοηθήσει κωφάλαλα άτομα, μιας και κάνοντας χειρονομίες μπροστά σε μια κάμερα ενός υπολογιστή, θα μπορεί ο υπολογιστής να καταλαβαίνει τι ακριβώς θέλουμε να πούμε μιας και αναγνωρίζοντας την κίνηση θα μεταφράζει την νοηματική γλώσσα.

1.2 Εφαρμογές και μελέτες που έχουν πραγματοποιηθεί με σκοπό την κατανόηση της γλώσσας του σώματος

Η γλώσσα του σώματος χρησιμοποιείται στο 80% των περιπτώσεων επικοινωνίας μεταξύ των ανθρώπων. Πολλοί νομίζουν ότι το μεγαλύτερο μέρος της

επικοινωνίας πραγματοποιείται μεταξύ των ανθρώπων με λόγια, αλλά στην πραγματικότητα, είναι τα νοήματα,τα νεύματα,ο τόνος της φωνής και οι κινήσεις του σώματος που παίζουν τον πιο σημαντικό ρόλο στην επικοινωνία. Μια επικοινωνία μόνο με λόγια, χωρίς έκφραση, είναι πολύ δύσκολη. Υπάρχουν φράσεις που αν ειπωθούν με διαφορετικό τόνο φωνής, ή με άλλες κινήσεις, ή ακόμα και απο άτομα διαφορετικού φύλου, παίρνουν άλλη σημασία. Μια επικοινωνία ας πούμε μεταξύ δύο ατόμων μέσω του διαδικτύου, πολλές φορές συνοδεύεται από σύμβολα που ειδοποιούν τον συνομιλητή για τον τόνο του άλλου. Την γλώσσα του σώματος την κατανοούμε κυρίως μέσω οπτικών ερεθισμάτων (κίνηση χεριών, κίνηση κεφαλιού, μορφασμοί προσώπου, κίνηση σώματος, στάση σώματος κλπ) επομένως βλέποντας μια εικόνα κάποιου η ακόμα καλύτερα ένα βίντεο μπορούμε να βγάλουμε κάποια συμπεράσματα για το πως συμπεριφέρεται. Οι διαδικτυακές κάμερες που έχουν εγκατασταθεί πλέον σε κάθε σπίτι και μεταφέρουν την εικόνα των συνομιλητών έχουν βοηθήσει πολύ την επικοινωνία των ανθρώπων. Με την δημιουργία των υπολογιστών και την συνεχή βελτίωσή τους αλλά και τις συνεχείς και πολλά υποσχόμενες μεθόδους επεξεργασίας εικόνας και βίντεο άρχισε να γίνεται μια προσπάθεια να δημιουργηθούν εφαρμογές που μέσα από μια εικόνα η ένα βίντεο να βγάζουν συμπεράσματα για την συμπεριφορά ενός ατόμου. Τέτοιες εφαρμογές δημιουργήθηκαν και χρησιμοποιούνται εκτενώς κυρίως από την αστυνομία και τις υπηρεσίες ασφαλείας προκειμένου να εντοπίζονται παραβατικές συμπεριφορές ατόμων (σε αεροδρόμια, γήπεδα, υπόγειους σιδηρόδρομους κλπ.), προκειμένου να προλαμβάνονται εγκλήματα, ή ακόμα και να εξιχνιάζονται (μέθοδοι εντοπισμού αν ένα άτομο λέει ψέματα μελετώντας την διαστολή της κόρης κλπ).

Στην προσπάθεια επικοινωνίας ανθρώπου μηχανής δεν θα μπορούσε να απουσιάζει η προσπάθεια να κάνουμε τον υπολογιστή να κατανοεί την γλώσσα του σώματος μέσα από εικόνες έτσι ώστε να μπορεί να επικοινωνεί αποτελεσματικότερα με τον άνθρωπο. Στις παραπάνω εφαρμογές απαιτείται και η συνδρομή ατόμων τα οποία αξιολογούν τα αποτελέσματα που δίνει ο υπολογιστής επεξεργαζόμενος μια εικόνα. Πλέον γίνεται μια προσπάθεια να μπορεί ο υπολογιστής χωρίς καμία βοήθεια (με το κατάλληλο λογισμικό) να κατανοεί την γλώσσα του σώματος. Σε αυτήν την προσπάθεια συμβάλει και η τεχνητή νοημοσύνη καθώς και τα νευρωνικά δίκτυα μιας και η πείρα του ανθρώπου είναι κάτι βασικό στην προσπάθεια κατανόησης της γλώσσας του σώματος. Μια χειρονομία μπορεί να έχει άλλη σημασία από κοινωνία σε κοινωνία, ο άνθρωπος έχει την πείρα και την ευφυΐα να το κατανοήσει και να αντιδράσει ανάλογα, ο υπολογιστής όμως πρέπει με κάποιο τρόπο να μπορεί να το καταλαβαίνει αλγοριθμικά, στηριζόμενος σε μια γνωσιακή βάση δεδομένων. Προκειμένου να κατανοηθεί η γλώσσα του σώματος πρέπει να μπορεί ο υπολογιστής να κατανοεί το τι κινήσεις γίνονται από τους ανθρώπους. Προκειμένου να γίνει αυτό πρέπει να μπορεί να εντοπίζει την κίνηση αλλά και το ανθρώπινο δέρμα. Μια σειρά ερευνών έχουν γίνει σε αυτήν την κατεύθυνση. Μια πολύ σημαντική δημοσίευση αναφέρεται στον εντοπισμό δέρματος με χρήση μόνο χρωματικών παραμέτρων με τίτλο Statistical Color Models with Application to Skin Detection. Οι μέθοδοι που εφαρμόζονται σε αυτήν την δημοσίευση είχαν 80% αποτελεσματικότητα στον εντοπισμό δέρματος ακόμα και στις πιο δύσκολες από άποψη εντοπισμού δέρματος εικόνες, έχοντας εφαρμοστεί σε πλήθος δειγμάτων ενός δισεκατομμυρίου εικόνων. Ο εντοπισμός αυτός επιτυγχάνεται με την χρήση των ιδιοτήτων των αποθηκευμένων

εικόνων δημιουργώντας συναρτήσεις που υπολογίζουν την πιθανότητα για το αν ένα pixel θεωρείται δέρμα η όχι και επιλέγοντας ένα κατάλληλο κατώφλι. Μια επίσης πολύ σημαντική δημοσίευση με τίτλο Face Segmentation Using Skin-Colour Map in Videophone Applications παρέχει μια σειρά από συνθήκες και τύπους προκειμένου να μπορεί κάποιος να εντοπίζει, σε μια φωτογραφία, πρόσωπα. Γενικεύοντας, δίνει και μερικά στοιχεία για τον εντοπισμό δέρματος και ορισμένοι από τους τύπους που χρησιμοποιούνται στη συνέχεια προέρχονται από αυτήν την δημοσίευση.

Η διπλωματική εργασία αυτή ασχολείται με την επικοινωνία ανθρώπου μηχανής δίνοντας ιδιαίτερη βάση στην κατανόηση της γλώσσας του σώματος και ειδικότερα στην κατανόηση των χειρονομιών και των κινήσεων του κεφαλιού. Η αναγνώριση χειρονομιών βασισμένη σε μια πραγματική ακολουθία εικόνων (βίντεο) χωρίζεται σε δύο βασικά μέρη, τον εντοπισμό δέρματος και τον εντοπισμό κίνησης. Έχοντας εντοπίσει την κίνηση και το δέρμα μπορούμε να εντοπίσουμε την δερματική κίνηση. Στα επόμενα κεφάλαια γίνεται μια αναλυτική αναφορά στον εντοπισμό δέρματος και στον εντοπισμό κίνησης σε μία εικόνα καθώς και στο πώς αυτές υλοποιούνται στις μέχρι τώρα εφαρμογές αλλά και στην εφαρμογή που δημιουργήθηκε σε αυτήν την διπλωματική εργασία.

1.3 Οργάνωση της εργασίας

Στο κεφάλαιο 2 αναλύεται ο εντοπισμός δέρματος γενικά, αλλά γίνεται αναφορά και στον εντοπισμό δέρματος σε αποθηκευμένες εικόνες. Γίνεται αναφορά στις έρευνες και στις εφαρμογές που έχουν πραγματοποιηθεί μέχρι σήμερα. Στην συνέχεια γίνεται μια αναλυτική περιγραφή του πώς με χρήση των υπαρχουσών μεθόδων και εφαρμογών εντοπίζεται το δέρμα σε αυτήν την διπλωματική εργασία. Με παραδείγματα διάφορες εικόνες και αναλύσεις, φαίνεται πώς το πρόγραμμα που δημιουργήθηκε εντοπίζει το δέρμα. Αναλύεται η μέθοδος που χρησιμοποιήθηκε βήμα προς βήμα. Τέλος γίνεται αναφορά στα προβλήματα του εντοπισμού δέρματος αλλά και στα συμπεράσματα που βγήκαν.

Στο κεφάλαιο 3 αναλύεται ο εντοπισμός κίνησης. Κομμάτια πολλών ερευνών που έχουν γίνει παρατίθενται και μέσα από αυτά επιλέγονται κάποιες αποτελεσματικές μέθοδοι προκειμένου να χρησιμοποιηθούν και σε αυτήν την διπλωματική εργασία. Ιδιαίτερη σημασία δίνεται στον εντοπισμό κίνησης μέσα από μια εικόνα. Δίνονται παραδείγματα και εικόνες των αποτελεσμάτων του εντοπισμού κίνησης που επιχειρήθηκε. Αξιολογείται η εφαρμογή που δημιουργήθηκε σε αυτήν την εργασία η οποία επεξηγείται βήμα προς βήμα. Τέλος γίνεται αναφορά στα προβλήματα του εντοπισμού κίνησης αλλά και τα συμπεράσματα που βγήκαν.

Στο κεφάλαιο 4 αναλύεται ο εντοπισμός δερματικής κίνησης. Τα συμπεράσματα που βγάλαμε στα δύο προηγούμενα κεφάλαια για τον εντοπισμό κίνησης και δέρματος χρησιμοποιούνται και συνδυάζονται προκειμένου να εντοπίσουμε την δερματική κίνηση. Με παραδείγματα και εικόνες επεξηγείται βήμα προς βήμα η διαδικασία που ακολουθήθηκε από το πρόγραμμά μου προκειμένου να

εντοπιστεί η δερματική κίνηση. Τέλος υπάρχει αναλυτικό σχέδιο του αλγορίθμου της διαδικασίας που ακολουθήθηκε.

Στο κεφάλαιο 5 δίνονται ορισμένα στοιχεία για την πλατφόρμα OpenCV η οποία χρησιμοποιήθηκε για την υλοποίηση του προγράμματος που εφαρμόζει της τεχνικές και τις μεθόδους που έχω αναλύσει στα προηγούμενα κεφάλαια προκειμένου να γίνει ο εντοπισμός δερματικής κίνησης. Δίνονται μερικές νέες δομές δεδομένων που μας παρέχει καθώς και μερικές συναρτήσεις που εμπλουτίζουν το σύνολο των συναρτήσεων της C++.

Στο κεφάλαιο 6 δίνονται οι συναρτήσεις του προγράμματος που δημιουργήθηκε προκειμένου να γίνει η επεξεργασία των εικόνων και να εντοπιστεί το δέρμα η κίνηση αλλά και η δερματική κίνηση. Η κάθε συνάρτηση δίνεται αναλυτικά και επεξηγούνται τα ορίσματα της, το τί επιστρέφει και το τί ακριβώς κάνει.

Στο κεφάλαιο 7 δίνεται συνοπτικά ο κώδικας των συναρτήσεων του προγράμματος (σε C++) που υλοποιεί τον εντοπισμό δέρματος, κίνησης και δερματικής κίνησης. Επίσης δίνεται και ο κώδικας που χρησιμοποιεί αυτές τις συναρτήσεις .

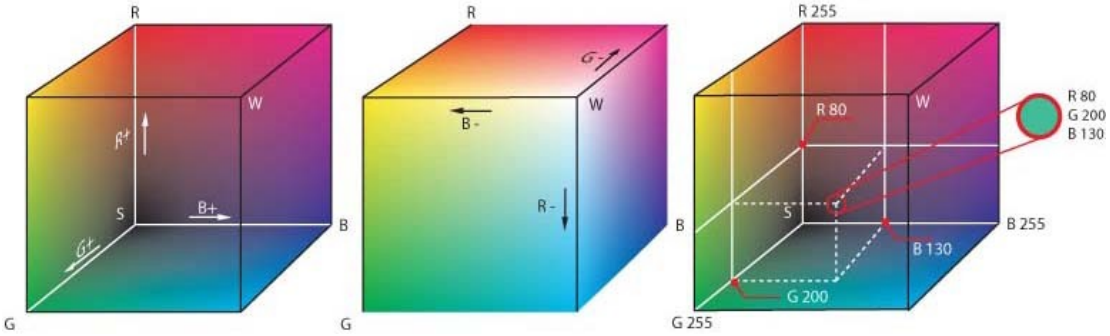
Στο κεφάλαιο 8 δίνεται μια σειρά από φωτογραφίες που δείχνουν τα αποτελέσματα του προγράμματος. Στην συνέχεια αναλύονται τα αποτελέσματα και δίνονται ορισμένες παρατηρήσεις. Τέλος βγαίνουν ορισμένα συμπεράσματα για μελλοντικές εργασίες πάνω στο ίδιο ή σε παρόμοια θέματα αλλά και το πώς μπορεί αυτή η εργασία να επεκταθεί με εφαρμογές που θα εξυπηρετήσουν άλλους τομής (υγείας,ασφάλειας κλπ) βασιζόμενες στον εντοπισμό δερματικής κίνησης. Μετά το $8^{ο}$ κεφάλαιο ακολουθεί η βιβλιογραφία που χρησιμοποιήθηκε προκειμένου να εκπονηθεί αυτή η διπλωματική εργασία.

Κ Ε Φ Α Λ Α Ι Ο 2

Εντοπισμός Δέρματος

2.1 Γενικά στοιχεία για τον εντοπισμό δέρματος

Το χρώμα για τον υπολογιστή δεν είναι τίποτα περισσότερο από μια αριθμητική τιμή (συνήθως δέκαεξαδική). Μια τιμή που στέλνει στο μέσο απεικόνισης (οθόνη, εκτυπωτή κ.λ.π.) και αυτό διαβάζοντάς την μας δείχνει κάποια χρωματική απόχρωση. Μια αποθηκευμένη εικόνα είναι ένας πίνακας ακεραίων αριθμών. Κάθε στοιχείο αυτού του πίνακα αντιστοιχεί σε ένα pixel. Ως pixel ορίζεται η ελάχιστη ψηφίδα της εικόνας. Μια εικόνα διαστάσεων N x N έχει N*N pixels. Οι εικόνες που μελετάμε αποτελούνται από τρία κανάλια. Δηλαδή από τρεις πίνακες με διαστάσεις όσο και οι διαστάσεις της εικόνας οι οποίοι ως στοιχεία τους έχουν αριθμητικές τιμές. Τα κανάλια αυτά μπορούν να μας δώσουν τον πίνακα της εικόνας μέσα από κάποιες αριθμητικές πράξεις τις οποίες ο κάθε τύπος εικόνας ορίζει. Οι πιο διαδεδομένες και πιο πολύ χρησιμοποιούμενες εικόνες είναι οι YCbCr και οι RGB. Οι RGB εικόνες έχουν τρία κανάλια το R (Red), το G (Green) και το B (Blue). Οι τρεις αυτοί πίνακες περιέχουν ακέραιους αριθμούς. Η τιμή του κάθε pixel της εικόνας υπολογίζεται από το άθροισμα των τιμών των αντίστοιχων στοιχείων των τριών πινάκων. Παρακάτω ακολουθεί μια εικόνα που δείχνει πως ακριβώς δουλεύει το RGB σύστημα αποθήκευσης.



Το YCbCr σύστημα αποθήκευσης έχει και αυτό τρία κανάλια το Y, το Cb και το Cr. Όπως και στο RGB σύστημα, το κάθε pixel της εικόνας υπολογίζεται από την άθροιση των τιμών των αντίστοιχων στοιχείων των τριών πινάκων. Μια βασική αρχή είναι ότι μια εικόνα αποθηκευμένη με οποιοδήποτε σύστημα μπορεί μέσω μαθηματικών υπολογισμών να μετατραπεί σε εικόνα αποθηκευμένη σε οποιοδήποτε άλλο σύστημα. Στην προκειμένη περίπτωση οι μαθηματικοί τύποι που χρειάζονται προκειμένου να πάμε από την RGB αναπαράσταση στην YCbCr είναι για κάθε στοιχείο των καναλιών η εξής:

$Y = Kr * R + (1 - Kr - Kb) * G + Kb * B$ (1)
$Cb = 0.5 * (B - Y) / (1 – Kb)$ (2)
$Cr = 0.5 * (R - Y) / (1 - Kr)$ (3)
όπου $Kb = 0.114$ και $Kr = 0.299$.

Σκοπός μας είναι από μια RGB ή μια YCbCr εικόνα να εντοπίσουμε αν μέσα σε αυτήν απεικονίζονται περιοχές ανθρώπινου δέρματος. Αρχικά φαίνεται κάτι απλό μιας και ένας άνθρωπος βλέποντας μια φωτογραφία μπορεί πολύ εύκολα να καταλάβει αν μέσα σε αυτήν την εικόνα υπάρχει ανθρώπινο δέρμα η όχι. Ο υπολογιστής όμως δεν έχει την ικανότητα του ανθρώπινου ματιού και έτσι πρέπει εμείς να βρούμε μια μέθοδο που να εντοπίζει αν υπάρχει δέρμα η όχι καθώς και τον ακριβή χώρο όπου αυτό εμφανίζεται. Ουσιαστικά από έναν πίνακα με ακεραίους που αντιστοιχούν σε χρωματικές αποχρώσεις να βρούμε ποιες περιοχές μπορεί να απεικονίζουν ανθρώπινο δέρμα. Ο άνθρωπος έχει την πείρα και την αφαιρετική ικανότητα να καταλάβει ποια περιοχή είναι δέρμα και ποια όχι. Όταν ας πούμε βλέπει μια μπλούζα σε μια εικόνα ξέρει ότι στο τέλος του μανικιού θα υπάρχει το χέρι οπότε με την βοήθεια και του χρώματος του δέρματος, το οποίο του είναι οικείο, μπορεί να καταλάβει ότι εκεί υπάρχει δέρμα. Επίσης λόγο της πείρας του ο άνθρωπος ξέρει ότι υπάρχουν και άνθρωποι με σκούρο δέρμα οπότε όταν δει ένα μαύρο χέρι δεν θα μπερδευτεί αλλά θα καταλάβει αμέσως ότι αυτό είναι δέρμα. Ο υπολογιστής από την άλλη δεν έχει ούτε πείρα ούτε την αφαιρετική ικανότητα του ανθρώπου. Εντοπισμός δέρματος για τον υπολογιστή είναι να μπορεί συγκρίνοντας τις τιμές των στοιχείων της εικόνας να μπορεί να αποφανθεί για το αν μια περιοχή είναι δέρμα η όχι.

Πολλές έρευνες και πολλές εφαρμογές έχουν πραγματοποιηθεί με σκοπό την δημιουργία ενός αλγορίθμου ο οποίος να μπορεί επεξεργαζόμενος μια εικόνα να εντοπίσει της δερματικές περιοχές. Μία έρευνα πραγματοποιήθηκε από το πανεπιστήμιο του Cambridge με τίτλο Statistical Color Models with Application to Skin Detection (βιβλιογραφία [29]) και τα αποτελέσματα τις ήταν πολύ ενθαρρυντικά για τον εντοπισμό δέρματος με βάση το χρώμα. Η έρευνα χρησιμοποιώντας τις αμέτρητες εικόνες που βρίσκονται στο διαδίκτυο έκανε εφικτή την δημιουργία ισχυρών γενικών μοντέλων για χαρακτηριστικά χαμηλού επιπέδου όπως το χρώμα, χρησιμοποιώντας απλές τεχνικές μάθησης. Η έρευνα έδειξε έναν εκπληκτικό βαθμό διαχωριστικότητας μεταξύ περιοχών με δέρμα και περιοχών χωρίς, η οποία επιτεύχθηκε δημιουργώντας έναν ανιχνευτή δέρματος με εντοπισμό δέρματος της τάξης του 80% με μόνο 8,5% αποτυχία. Η έρευνα κατόρθωσε να φτιάξει ένα πρόγραμμα, βασισμένο στα παραπάνω μοντέλα, που μπορούσε να εντοπίσει σε εικόνες και βίντεο τα γυμνά άτομα. Το συμπέρασμα της έρευνας ήταν ότι το χρώμα, σε συνδυασμό με μαθησιακές τεχνικές, μπορεί να παίξει έναν σημαντικό ρόλο που κανείς δεν είχε υποψιαστεί στον εντοπισμό δέρματος και γενικότερα ανθρώπων. Αυτά τα συμπεράσματα έχουν υποστηριχτεί και από άλλες έρευνες που βρίσκονται στην βιβλιογραφία. Ο τρόπος εντοπισμού αυτός χρησιμοποιήθηκε και σε αυτήν την διπλωματική εργασία.

Μία άλλη προσέγγιση στον εντοπισμό δέρματος και πιο συγκεκριμένα στον εντοπισμό ανθρώπων έχει τίτλο Face Segmentation Using Skin-Color Map in Videophone Applications (βιβλιογραφία [30]). Η έρευνα αυτή προτείνει μεθόδους προκειμένου σε εικόνες που αποτελούνται από το πρόσωπο ενός ατόμου και τους ώμους του καθώς και ένα πολύπλοκο φόντο να εντοπίζεται το πρόσωπο του ατόμου.

Η μέθοδος που αναλύεται σε αυτήν την δημοσίευση βασίζεται σε έναν αξιόπιστο γρήγορο και αποτελεσματικό αλγόριθμο που εκμεταλλεύεται την χωρική κατανομή των χαρακτηριστικών του χρώματος του ανθρώπινου δέρματος. Ένας γενικής χρήσης χρωματικός-δερματικός χάρτης δημιουργείται και χρησιμοποιείται στα χρωματικά συστατικά της εικόνας προκειμένου να εντοπίσει pixel με χρώμα δέρματος. Μετά η μέθοδος βασιζόμενη στα διάσπαρτα pixel που έχουν εντοπιστεί ως δέρμα και τις σχετικές τιμές φωτεινότητας εφαρμόζει μια νέα διαδικασία ομαλοποίησης η οποία ενισχύει τα pixel που ανήκουν σε περιοχές με μεγάλη πιθανότητα να ανήκουν στο πρόσωπο και απορρίπτει αυτά που ανήκουν σε περιοχές με μικρή πιθανότητα να ανήκουν στο πρόσωπο. Η έρευνα αυτή χρησιμοποιείται σε εφαρμογές βιντεοτηλεφωνίας βελτιώνοντας κατά πολύ την ποιότητα της υπηρεσίας.

Με βάση το χρώμα και με την χρήση μιας πολύ απλής μαθησιακής μεθόδου η οποία περιγράφεται παρακάτω επιχειρήθηκε να γίνει ο εντοπισμός δέρματος και σε αυτήν την διπλωματική εργασία. Τα αποτελέσματα ήταν πολύ καλά ακόμα και σε πολύπλοκες περιπτώσεις όπου ο εντοπισμός δέρματος καθίσταται αρκετά δύσκολος. Το χρώμα αποδείχτηκε ένα εργαλείο πολύ χρήσιμο και όπως και οι μέχρι τώρα έρευνες δείχνουν θα καταφέρουμε κάποια στιγμή να εντοπίζουμε το ανθρώπινο δέρμα μέσω του χρώματος σε τέτοιο βαθμό όπως το ανθρώπινο μάτι μπορεί. Μια τέτοια δυνατότητα θα κάνει την επικοινωνία ανθρώπου μηχανής πιο εύκολη και θα απλοποιήσει κατά πολύ την καθημερινή μας ζωή. Ήδη οι υπολογιστές βρίσκονται παντού και επηρεάζουν ποικιλοτρόπως την ζωή μας κάθε προσπάθεια απλοποίησης της σχέσης μας με αυτούς τους καθιστά ακόμα πιο χρήσιμους. Υπάρχουν άτομα που θεωρούν ότι η χρήση ενός υπολογιστή είναι κάτι πολύπλοκο. Όταν όμως θα μπορούν να επικοινωνούν με τον υπολογιστή πολύ απλά και γρήγορα και θα βλέπουν άμεσα αποτελέσματα μέσω εφαρμογών όπως αυτή, η αντιμετώπισή τους θα αλλάξει και θα βρουν ένα νέο εργαλείο εύχρηστο, λειτουργικό και αξιόπιστο. Παρακάτω εξηγούνται αναλυτικά οι μέθοδοι που χρησιμοποιούνται για τον εντοπισμό δέρματος καθώς και το πώς λειτουργούν.

2.2 Μάσκα δέρματος

Η τεχνική που εφαρμόζεται στο περιγραφόμενο στην εργασία αυτή πρόγραμμα είναι η εξής: Πριν αρχίσει η επεξεργασία του βίντεο ζητείται από τον χρήστη να προσδιορίσει μια περιοχή της εικόνας στην οποία υπάρχει δέρμα. Στην συνέχεια με βάση την πληροφορία αυτή και δεδομένου ότι ο φωτισμός θα παραμείνει σταθερός κατά την διάρκεια του βίντεο (αν αλλάξει ο φωτισμός θα πρέπει να επαναληφθεί η διαδικασία εισαγωγής περιοχής από το χρήστη). Υπολογίζεται μια μήτρα ίσων διαστάσεων με αυτή των εικόνων του βίντεο η οποία γα κάθε εικόνα περιέχει την πιθανότητα το pixel που αντιστοιχεί σε κάθε στοιχείο της μήτρας να είναι δέρμα. Με αυτόν τον τρόπο έχουμε μια ικανοποιητική επιλογή ορισμένων pixels ως δέρμα. Η μήτρα αυτή δημιουργείται με χρήση των Cb και Cr διανυσμάτων της εικόνας αλλά και της περιοχής την οποία έχει ορίσει ο χρήστης ως μια ενδεικτική χρωματική περιοχή που στην κάθε εφαρμογή θεωρείται ως χρώμα του δέρματος. Η

διαδικασία που ακολουθείται (και έχει βρεθεί από την βιβλιογραφία) είναι η εξής: Αρχικά υπολογίζεται ο μέσος όρος των τιμών των στοιχείων των Cb και Cr πινάκων που βρίσκονται στην περιοχή που έχει δηλώσει ο χρήστης. Mx= μέσος όρος του Cb και My= μέσος όρος του Cr.

$$M_x = \frac{1}{N}\sum_i\sum_j Cb(i,j) \quad (4)$$

για όλα τα i ,j που έχει ορίσει ο χρήστης και N το πλήθος των στοιχείων και

$$M_y = \frac{1}{N}\sum_i\sum_j Cr(i,j) \quad (5)$$

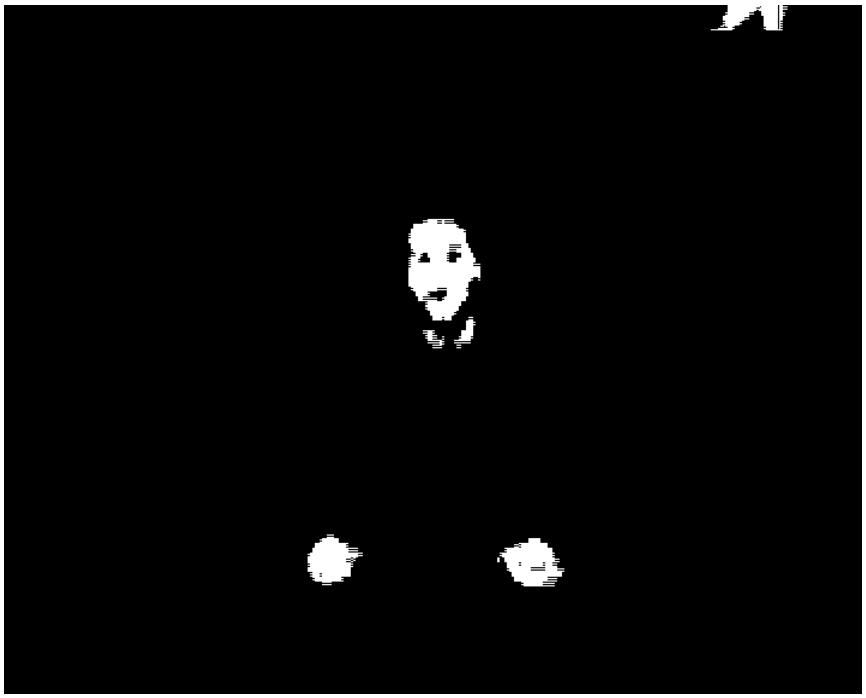
για όλα τα i ,j που έχει ορίσει ο χρήστης και N το πλήθος των στοιχείων. Στην συνέχεια με χρήση των τύπων που έχουν δημοσιευθεί στο paper των Rick Kjeldsen and John Kender [24] υπολογίζω τον πίνακα πιθανότητας δέρματος Z με χρήση του τύπου

$$Z(i,j) = \exp(-0{,}5*((0{,}0611*(Cr(i,j)-M_x)-0{,}9981*(Cb(i,j)-M_y))^2\frac{1}{s_x}+ \\ +(-0{,}9981*(Cr(i,j)-M_x)-0{,}0611(Cb(i,j)-M_y))^2\frac{1}{s_y})) \quad (6)$$

Όπου $s_x = s_y = 0.005$ το οποίο προήλθε μετά από πειράματα σύμφωνα με τις παραπάνω δημοσιεύσεις. Όταν έχουμε τον πίνακα που περιέχει τις πιθανότητες το κάθε pixel να είναι δέρμα, επιλέγουμε ένα κατώφλι και κρατάμε όσες τιμές βρίσκονται πάνω από αυτό και τις περιοχές που βρίσκονται πάνω από αυτό το κατώφλι τις θεωρούμε δερματικές περιοχές. Η επιλογή του κατωφλίου αυτού γίνεται συναρτήσει του πόση ακρίβεια θέλουμε να έχουμε. Αν θέσουμε το κατώφλι ίσο με 80%, παίρνουμε περιοχές που έχουν πιθανότητα μεγαλύτερη από 80% να είναι δέρμα, οι πιθανότητες μειώνονται ή αυξάνονται αντίστοιχα και επομένως και η ακρίβεια. Στην προκειμένη περίπτωση επιλέχθηκε κατώφλι ίσο με 0.8 μιας και είχε ικανοποιητικά αποτελέσματα και προτείνεται και από την βιβλιογραφία. Ακολουθεί ένα παράδειγμα δημιουργίας μάσκας με χρήση του πίνακα πιθανότητας δέρματος.

Εικόνα 1: Αρχική εικόνα

Εικόνα 2: Μάσκα δέρματος(κατώφλι=0.8)

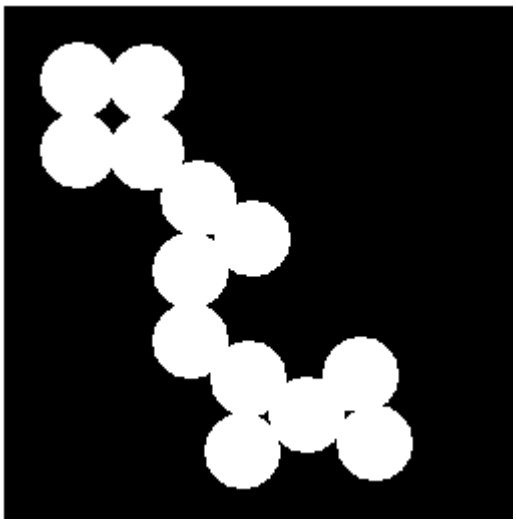


Εικόνα 3: Μάσκα δέρματος(κατώφλι=0.5)

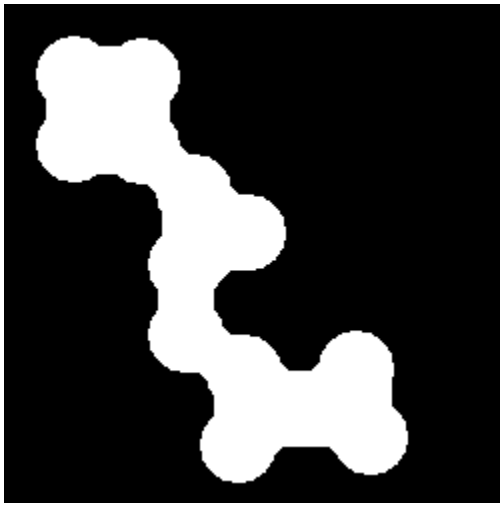
Εικόνα 4: Μάσκα δέρματος(κατώφλι=0.9)

Μετά την δημιουργία της μάσκας δέρματος χρησιμοποιούνται μορφολογικά φίλτρα προκειμένου να κάνουμε συμπαγείς τις περιοχές του δέρματος και να τις χωρίσουμε έτσι σε ενιαία κομμάτια προκειμένου να επεξεργαστούν μετά σαν αντικείμενα.

Συγκεκριμένα γίνεται μορφολογικό κλείσιμο στην μάσκα. Η διαδικασία αυτή μας παρέχεται έτοιμη από το OpenCV. Το τι κάνει αυτή η διαδικασία φαίνεται παρακάτω. Δίνοντάς της την παρακάτω εικόνα



παίρνουμε το εξής αποτέλεσμα!



Στην μάσκα δέρματος αν εφαρμόσουμε αυτήν την διαδικασία παίρνουμε την εξής μάσκα:

Εικόνα 5: Μάσκα δέρματος μορφολογικά επεξεργασμένη

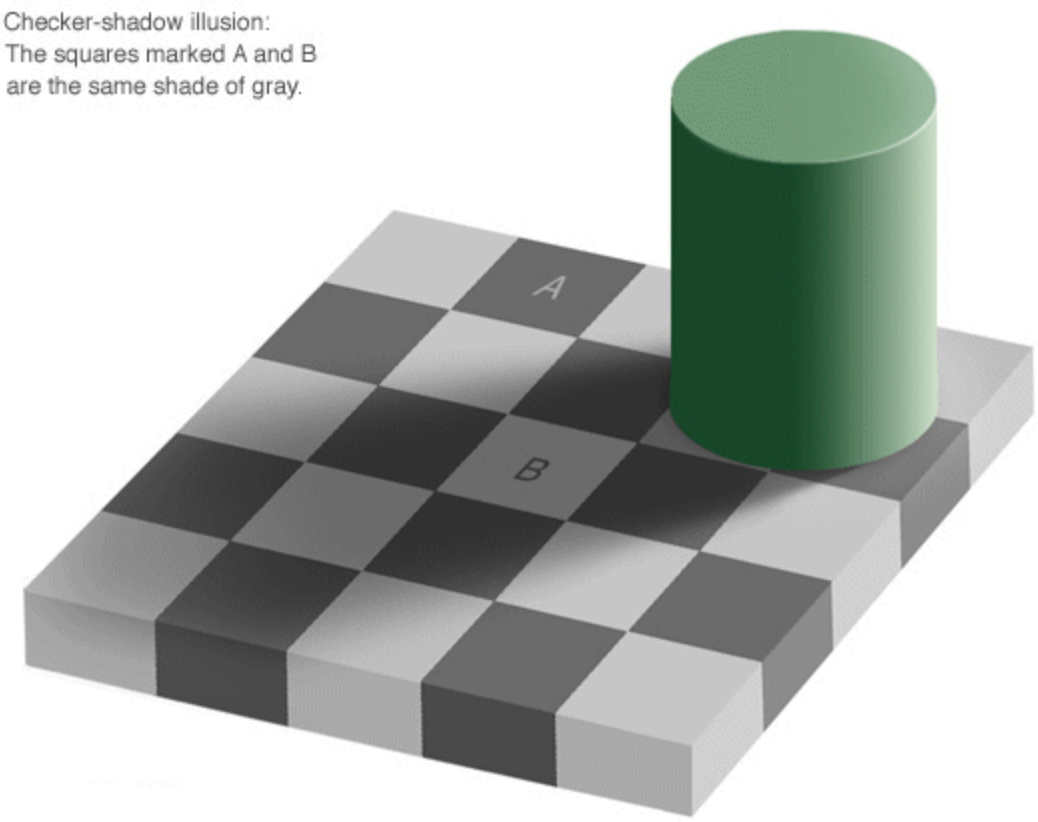
Στην συνέχεια υπολογίζονται τα μεγέθη των αντικειμένων που έχουν δημιουργηθεί απορρίπτοντας όσα αντικείμενα έχουν μέγιστη απόσταση μεταξύ των σημείων τους μικρότερη του 0,01 του ύψους της εικόνας. Το κατώφλι αυτό επιλέχτηκε μιας και συνιστάται από την βιβλιογραφία. Έτσι φτάνουμε στην τελική μάσκα δέρματος η οποία περιέχει δερματικές περιοχές ικανοποιητικού μεγέθους μέσα στις οποίες πρέπει να είναι τα χέρια και το κεφάλι. Στο παράδειγμα μας η δερματική μάσκα είναι η εξής:

Εικόνα 6: Διαχωρισμός αντικειμένων στην μάσκα δέρματος

Ο προσδιορισμός των χεριών και του κεφαλιού δεν θα γίνει τώρα αλλά αφού υπολογιστεί η μάσκα δερματικής κίνησης παρακάτω. Το επόμενο βήμα είναι να εντοπίσουμε από την περιοχή του δέρματος το κινούμενο δέρμα και για να γίνει αυτό θα υπολογίσουμε πρώτα την κίνηση και μετά την κίνηση του δέρματος.

2.3 Προβλήματα που δυσχεραίνουν τον εντοπισμό δέρματος

Την δημιουργία μιας μάσκας ακριβούς υπολογισμού του δέρματος δυσχεραίνουν ορισμένοι παράγοντες, που παρόλες τις προσπάθειες μας, δεν έχουμε καταφέρει να εξαλείψουμε τις επιπτώσεις τους στις μεθόδους εντοπισμού δέρματος που εφαρμόζουμε. Αυτοί οι παράγοντες είναι ο φωτισμός του χώρου, τα περιβάλλοντα αντικείμενα, συγκεκριμένα χαρακτηριστικά του δέρματος του ατόμου που εμφανίζεται στο βίντεο και η λήψη εικόνας κακής ποιότητας. Όσον αφορά στον φωτισμό, ο χώρος στον οποίο γίνεται η λήψη των εικόνων μπορεί να φωτίζεται με τέτοιο τρόπο έτσι ώστε το δέρμα να έχει ένα χρώμα εντελώς διαφορετικό από αυτό που στην πραγματικότητα έχει, υπό κανονικές συνθήκες φωτισμού. Έτσι η χρωματική περιοχή που θα ψάχναμε στην εικόνα προκειμένου να εντοπίσουμε το δέρμα θα είναι άλλη από αυτήν που το δέρμα κανονικά έχει και κατά συνέπεια δεν θα εντοπίζεται δέρμα στην εικόνα. Ένα άλλο πρόβλημα φωτισμού είναι οι σκιές. Έστω ότι έχουμε τις ιδανικές συνθήκες φωτισμού και ένα αντικείμενο σκιάζει το ένα χέρι του ατόμου. Το χέρι αυτό δεν θα μπορεί να αναγνωρισθεί ως δέρμα μιας και θα είναι εκτός της χρωματικής περιοχής εντός της οποίας κάποιο χρώμα θεωρείται δέρμα. Παρακάτω παρατίθεται ένα παράδειγμα.



Τα Α και Β έχουν το ίδιο χρώμα ενώ γνωρίζουμε ότι κανονικά το ένα είναι μαύρο και το άλλο άσπρο. Η σκιά του κυλίνδρου όμως προκαλεί αυτό το φαινόμενο, έχοντας Χρώμα Α (RGB = 107, 107, 107) Χρώμα Β (RGB = 107, 107, 107). Ακόμα ένα πρόβλημα είναι τα περιβάλλοντα αντικείμενα Τα αντικείμενα του χώρου γύρω από το άτομο, τις κινήσεις του οποίου προσπαθούμε να εντοπίσουμε, δεν είναι προκαθορισμένα ούτε ως προς το σχήμα ούτε ως προς το χρώμα. Αυτό μπορεί να μας επηρεάσει μιας και κάποιο αντικείμενο μπορεί να έχει χρώμα παραπλήσιο με αυτό του δέρματος και έτσι να το μπερδέψουμε με δέρμα και ειδικά αν βρίσκεται κοντά σε δέρμα μπορεί να μας οδηγήσει σε σημαντικά λάθη. Τα περιβάλλοντα αντικείμενα επίσης προκαλούν διάφορες σκιάσεις με αποτέλεσμα να αλλάζουν τα χρώματα και να μην μπορούμε να τα εντοπίσουμε όπως θα τα εντοπίζαμε αν δεν ήταν τα αντικείμενα αυτά εκεί. Τα ρούχα του ατόμου μπορεί επίσης να προκαλέσουν σημαντικά προβλήματα αν είναι παραπλήσια με το χρώμα του δέρματος, ή αν γυαλίζουν στο φως ή ακόμα και αν κρύβουν πολύ το δέρμα (καλό είναι τα άτομα να φοράνε ρούχα που να κάνουν αντίθεση με το δέρμα τους). Ακόμα το δέρμα ατόμου αν δεν είναι σε καλή κατάσταση προκαλεί προβλήματα στον εντοπισμό δέρματος. Υπάρχουν άτομα τα οποία παρουσιάζουν δερματικά φαινόμενα τα οποία επηρεάζουν τον εντοπισμό δέρματος. Όταν η επιδερμίδα δεν είναι λεία και καθαρή (εξανθήματα, σπυριά, ουλές , εκ γενετής σημάδια, έντονη τριχοφυΐα κλπ) αλλάζει σημαντικά το χρώμα του δέρματος όταν αυτό αποθηκεύεται από ψηφιακό μέσο και έτσι, ενώ μπορεί να έχουμε τέλειες συνθήκες φωτισμού, ο εντοπισμός δέρματος μπορεί να αποτύχει μιας και δεν θα βρίσκεται στην εικόνα το χρώμα του δέρματος. Ο ιδρώτας επίσης μπορεί να προκαλέσει αντανάκλαση και έτσι το δέρμα να πάρει λευκό χρώμα πράγμα πολύ

σύνηθες, ειδικά όταν το δέρμα αλλάζει θέση ως προς τον φωτισμό. Τέλος ένα κακής ποιότητας μέσο λήψης εικόνας, ή βίντεο, μπορεί να έχει καταστροφικές συνέπειες. Μια κάμερα με μικρή ανάλυση και μικρό βάθος χρώματος μπορεί να έχει σαν αποτέλεσμα το χρώμα του δέρματος να είναι παρόμοιο με αλλά χρώματα στην εικόνα που στην πραγματικότητα είναι εντελώς διαφορετικά και έτσι ο εντοπισμός δέρματος να αποτύχει. Για να αποφύγουμε όλες αυτές τις προαναφερθείσες δυσκολίες πρέπει με κάποιο τρόπο ο χρήστης πριν αρχίσει τον εντοπισμό του δέρματος να δηλώσει ποιά περιοχή θεωρείται δέρμα υπό τον υπάρχοντα φωτισμό και την υπάρχουσα διάταξη των αντικειμένων και των σκιάσεων. Έτσι αποφεύγουμε μια σειρά από προβλήματα και είμαστε σε καλό δρόμο στην προσπάθεια μας να εντοπίσουμε το δέρμα. Επίσης το άτομο που θα προσπαθήσουμε να εντοπίσουμε θα πρέπει να έχει καθαρό και λείο δέρμα και να φοράει ρούχα που να κάνουν αντίθεση με το δέρμα του.

Κ Ε Φ Α Λ Α Ι Ο 3

Εντοπισμός Κίνησης

3.1 Γενικά στοιχεία για τον εντοπισμό κίνησης

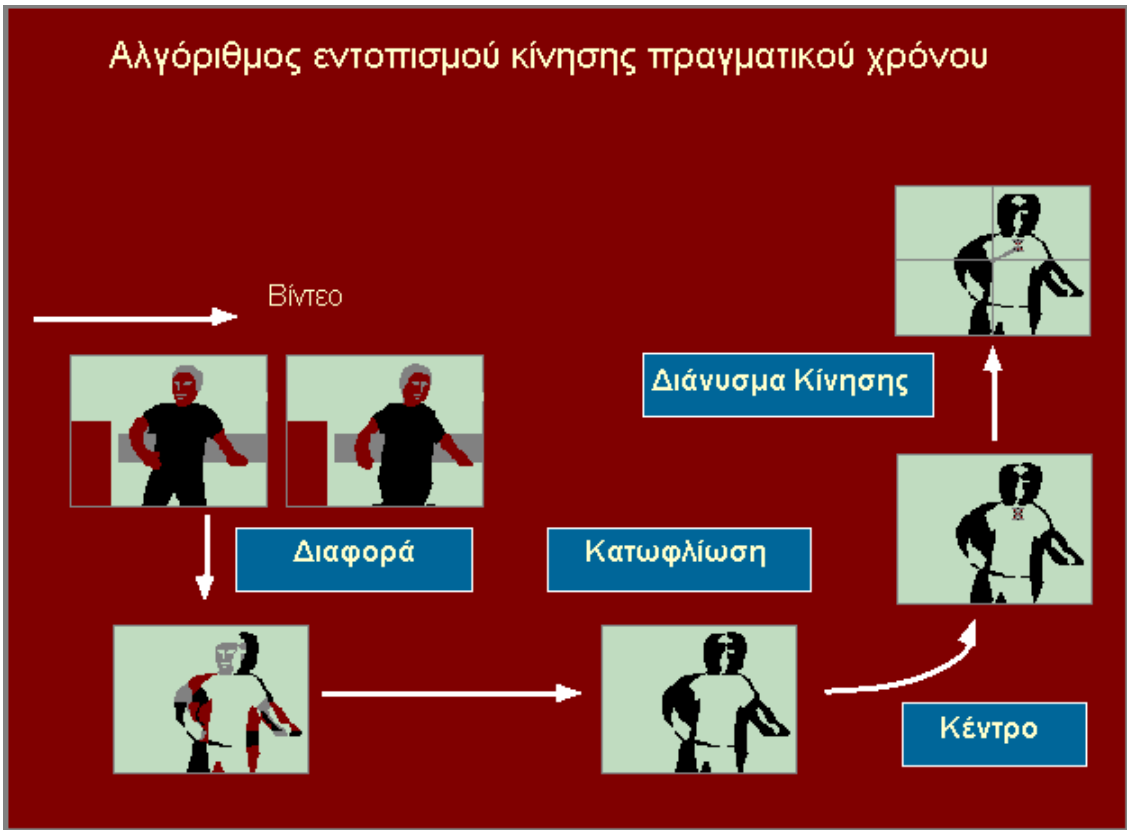
Ως εντοπισμός κίνησης ορίζεται η διαδικασία μέσω της οποίας μπορούμε να αποφανθούμε για το αν σε έναν δεδομένο χώρο υπήρξε κάποιας μορφή κίνηση (αντικειμένου, ανθρώπου και γενικά του οτιδήποτε μπορεί να κινηθεί). Ως κίνηση ορίζεται η αλλαγή θέσης ενός αντικειμένου στο τρισδιάστατο χώρο. Για τον εντοπισμό αυτό υπάρχουν πολύ μέθοδοι. Υπάρχουν συσκευές που με φωτοκύτταρα εντοπίζουν την κίνηση, ή συσκευές που με ακουστικούς αισθητήρες αντιλαμβάνονται αν σε κάποιο χώρο κάποιο αντικείμενο κάνει κάποια κίνηση, εντοπίζοντας τους ήχους που αυτή η κίνηση προκαλεί. Επίσης υπάρχουν οπτικοί τρόποι οι οποίοι με κάμερες εντοπίζουν την κίνηση. Ο οπτικός εντοπισμός κίνησης είναι αυτός για τον οποίο ενδιαφερόμαστε πιο πολύ σε αυτήν την διπλωματική εργασία. Γίνεται δηλαδή απόπειρα εντοπισμού κίνησης μέσα από μια λήψη βίντεο. Το θέμα αυτό έχει απασχολήσει πολλούς ερευνητές και έχει πολλές εφαρμογές σήμερα. Υπάρχουν κάμερες παρακολούθησης που, όταν εντοπίσουν κίνηση, ενεργοποιούν κάποιο συναγερμό, τηλεσκόπια, που αρχίζουν την καταγραφή δεδομένων μόλις εντοπίσουν μια κίνηση στην εικόνα του ουρανού και γενικά μια πληθώρα εφαρμογών. Ο εντοπισμός κίνησης μέσα από μια σειρά εικόνων είναι κάτι εύκολο για τον άνθρωπο άλλα κάτι αρκετά δύσκολο για τον υπολογιστή. Σε ένα βίντεο κινούνται πολλά αντικείμενα εκτός από αυτά που εμείς θέλουμε να εντοπίσουμε. Ο υπολογιστής δεν έχει την αφαιρετική ικανότητα του ανθρώπινου ματιού ώστε να ξέρει ποιο αντικείμενο πρέπει να εντοπίσει, παρά μόνο εντοπίζει όλα τα αντικείμενα που κινούνται και μάλιστα όχι στον τρισδιάστατο χώρο αλλά στις δύο διαστάσεις του βίντεο. Είναι δική μας δουλειά να βρούμε έναν τρόπο ώστε ο υπολογιστής να εντοπίζει μόνο την κίνηση που πρέπει να εντοπίζει και να αγνοεί κινήσεις που δεν μπορούμε να απαλείψουμε κατά την διάρκεια λήψης ενός βίντεο. Οι μέθοδοι που έχουν προταθεί για τον εντοπισμό κίνησης σε βίντεο είναι πολλές και συνέχεια προτείνονται νέες με συνεχώς καλύτερα αποτελέσματα. Ένας ολόκληρος τομέας της όρασης υπολογιστών ασχολείται με τον εντοπισμό κίνησης σε βίντεο. Ακόμα έχει χρησιμοποιηθεί η τεχνητή νοημοσύνη αλλά και τα νευρωνικά δίκτυα προκειμένου να μπορούμε να εντοπίσουμε την κίνηση (και την διεύθυνση της) σε ένα βίντεο. Πολλές δημοσιεύσεις έχουν γίνει πάνω σε αυτό το θέμα αλλά ακόμα δεν έχουμε ικανοποιητικά αποτελέσματα. Οι δυσκολίες είναι πολλές και παρόλο που έχουμε εφαρμογές που εντοπίζουν την κίνηση σε ικανοποιητικό βαθμό δεν έχουμε ακόμα

υπερβεί κάποιες δυσκολίες όπως ο θόρυβος, η περίπτωση να κινείται το μέσο λήψης του βίντεο κ.α.

Ο εντοπισμός κίνησης έχει και άλλες παραμέτρους εκτός από τις προαναφερθείσες. Η κίνηση είναι ένα διανυσματικό μέγεθος. Έτσι έχει κατεύθυνση , φορά, σημείο εφαρμογής και μέτρο. Αντικείμενο πολλών ερευνών είναι σε ένα βίντεο να υπολογίσουμε την ταχύτητα ή την κατεύθυνση της κίνησης ενός αντικειμένου. Ένας τέτοιος εντοπισμός είναι αρκετά δύσκολος μιας και ορισμένες λεπτομέρειες (κίνηση πολλών αντικειμένων, κίνηση της κάμερας που λαμβάνει το βίντεο, κίνηση ενός αντικειμένου προς την κάμερα κ.λ.π.) μπορούν να αλλοιώσουν τα αποτελέσματα του εντοπισμού. Αυτό γίνεται διότι στην εικόνα δεν έχουμε μια τρισδιάστατη αντίληψη του χώρου αλλά ουσιαστικά έχουμε την προβολή ενός τρισδιάστατου χώρου σε ένα δισδιάστατο επίπεδο την θέση του οποίου ορίζει ουσιαστικά η θέση της κάμερας. Κατά συνέπεια έχουμε μια αφαίρεση τμημάτων της πραγματικότητας και όχι όλη την πληροφορία που έχει ένας παρατηρητής όταν βρίσκεται μέσα σε έναν τρισδιάστατο χώρο. Επίσης η κάμερα κάνει μια λήψη 25 εικόνων το δευτερόλεπτο, που είναι μια ικανοποιητική μεν δειγματοληψία, μιας και ‘μπερδεύει’ το ανθρώπινο μάτι και αντί για συνεχείς εικόνες βλέπει βίντεο, αλλά ένα κομμάτι της πληροφορίας δεν αποθηκεύεται. Οι παραπάνω ατέλειες μπορούν να προκαλέσουν προβλήματα στην προσπάθεια εντοπισμού της κίνησης. Αν ένα αντικείμενο που κινείται στον τρισδιάστατο χώρο, ας πούμε, αρχίσει να κάνει μια παλινδρομική κίνηση με φορά κάθετη στο επίπεδο της κάμερας το αντικείμενο αυτό θα φαίνεται ακίνητο στο βίντεο με αποτέλεσμα η κίνηση του να μην μπορεί να εντοπιστεί. Ακόμα, αν ένα αντικείμενο εμφανιστεί ξαφνικά στο χώρο και κάνει μια πολύ γρήγορη κίνηση, υπάρχει το ενδεχόμενο να μην βρίσκεται σε κάποια από τις 25 εικόνες που παίρνει η κάμερα το δευτερόλεπτο και έτσι η κίνηση δεν θα εντοπιστεί. Ένα επίσης δύσκολο στον εντοπισμό χαρακτηριστικό είναι η λεπτομέρεια που η κάμερα μπορεί να συλάβει. Μπορεί, ας πούμε, να φαίνεται σε ένα βίντεο η κίνηση του κεφαλιού ενός ατόμου, αλλά αν δεν έχει καλή ανάλυση η κάμερα, μία κίνηση του φρυδιού μπορεί να μην είναι ευδιάκριτη.

Τις δυσκολίες που προαναφέρθηκαν προσπαθούν να απαλείψουν διάφορες μέθοδοι που έχουν προταθεί για να δώσουν έναν ακριβή και ολοκληρωμένο εντοπισμό κίνησης. Οι μέθοδοι αυτοί χωρίζονται στις μεθόδους πραγματικού χρόνου, που εντοπίζουν την κίνηση μέσα από μια κάμερα την ώρα που αυτή κάνει την λήψη (κάμερες συστημάτων ασφαλείας) και σε αυτές που επεξεργάζονται αποθηκευμένα δεδομένα ετεροχρονισμένα, με σκοπό να βγάλουν διάφορα συμπεράσματα (κάμερες μετεωρολογικών δορυφόρων). Εμάς μας ενδιαφέρουν οι μέθοδοι πραγματικού χρόνου μιας και σκοπός μας είναι ο εντοπισμός κίνησης σε μια λήψη από κάμερα, την ώρα που η κάμερα αυτή κάνει την λήψη, προκειμένου ο υπολογιστής να προβεί σε κάποιες ενέργειες όταν αντιλαμβάνεται ότι υπάρχει κίνηση. Η πιο ευρέως διαδεδομένη μέθοδος πραγματικού χρόνου είναι η εξής. Σε ένα βίντεο ο εντοπισμός κίνησης γίνεται μεταξύ δύο συνεχόμενων εικόνων για όλες τις εικόνες του βίντεο. Το επόμενο βήμα είναι ο υπολογισμός της διαφοράς μεταξύ των εικόνων. Δηλαδή ο υπολογισμός ενός πίνακα (μάσκας) ιδίων διαστάσεων με τις δύο εικόνες, όπου κάθε στοιχείο του είναι η διαφορά των αντίστοιχων στοιχείων των δύο εικόνων. Η διαφορά αυτή ουσιαστικά είναι η κίνηση μιας και ως κίνηση σε αυτήν την μέθοδο θεωρείται οτιδήποτε αλλάζει στην εικόνα. Το τρίτο βήμα είναι η επιλογή ενός

κατωφλίου και η κατωφλίωση με αυτό το κατώφλι του πίνακα της διαφοράς. Κατωφλίωση είναι η διαδικασία κατά την οποία όσες τιμές ενός πίνακα βρίσκονται πάνω απο το κατώφλι μένουν αμετάβλητες ενώ όσες βρίσκονται κάτω από αυτό μηδενίζονται. Με αυτόν τον τρόπο οι μικρές τιμές του πίνακα διαφοράς απαλείφονται και παύουν να θεωρούνται κίνηση. Το κατώφλι είναι αυτό που καθορίζει την ευαισθησία της μεθόδου εντοπισμού. Αν το κατώφλι είναι μικρό έχουμε μεγάλη ευαισθησία ενώ αν είναι μεγάλο έχουμε μικρή ευαισθησία. Το τέταρτο βήμα είναι η εύρεση του κέντρου της κίνησης. Αυτό υπολογίζεται με μαθηματικούς τύπους εύρεσης του κέντρου μιας περιοχής. Τέλος υπολογίζεται το διάνυσμα κίνησης. Το διάνυσμα κίνησης έχει αρχή το κέντρο της εικόνας και τέλος το κέντρο της κίνησης. Τα διανύσματα κίνησης μεταξύ των εικόνων αποθηκεύονται και χρησιμοποιούνται προκειμένου να βγάλουμε τα οποιαδήποτε συμπεράσματα χρειαζόμαστε. Ακολουθεί ο αλγόριθμος της μεθόδου εντοπισμού που περιγράφηκε σχηματικά.



Η μέθοδος αυτή είναι χρονοβόρα και χρειάζεται αρκετούς υπολογιστικούς πόρους, μιας και οι εικόνες είναι συνήθως μεγάλης ανάλυσης, με αποτέλεσμα να χρειάζεται αρκετή ώρα ο επεξεργαστής προκειμένου να κάνει τα βήματα που προαναφέρθηκαν. Και δεδομένου ότι το βίντεο αποτελείται από πολλές εικόνες (δειγματοληψία 25 εικόνων το δευτερόλεπτο) γίνεται εύκολα αντιληπτό ότι ο χρόνος επεξεργασίας αυξάνεται κατά πολύ. Η μέθοδος που περιγράφηκε χρησιμοποιείται για τον εντοπισμό της κίνησης και στην παρούσα εργασία. Παρακάτω επεξηγείται

αναλυτικά βήμα προς βήμα και με παραδείγματα το πώς έγινε ο εντοπισμός της κίνησης και πώς καταλήξαμε στην μάσκα κίνησης όπου φαίνονται οι περιοχές στις οποίες έγινε η κίνηση.

3.2 Μάσκα κίνησης

Στο πρόγραμμά μας ο εντοπισμός της κίνησης γίνεται ως εξής. Η κίνηση εντοπίζεται ανά δυο εικόνες για όλες τις εικόνες του βίντεο. Υπολογίζεται μια μήτρα με την χρωματική διαφορά μεταξύ των pixels των δυο εικόνων (δηλαδή ένας πίνακας με τιμές τα απόλυτα της αφαίρεσης των τιμών των αντίστοιχων θέσεων των πινάκων των δύο εικόνων). Όπου η διαφορά αυτή είναι μεγαλύτερη από μία τιμή τότε θεωρούμε ότι εκεί υπήρξε κίνηση. Συγκεκριμένα η διαδικασία γίνεται ως εξής. Αρχικά υπολογίζεται ένας πίνακας διαφοράς μεταξύ δύο εικόνων. Έστω ότι θέλουμε να εντοπίσουμε την κίνηση μεταξύ των δύο εικόνων που ακολουθούν.

Εικόνα 7: Αρχική εικόνα

Εικόνα 8: Η αρχική εικόνα μετά την κίνηση

Ο πίνακας διαφοράς μεταξύ των εικόνων υπολογίζεται από τον εξής τύπο

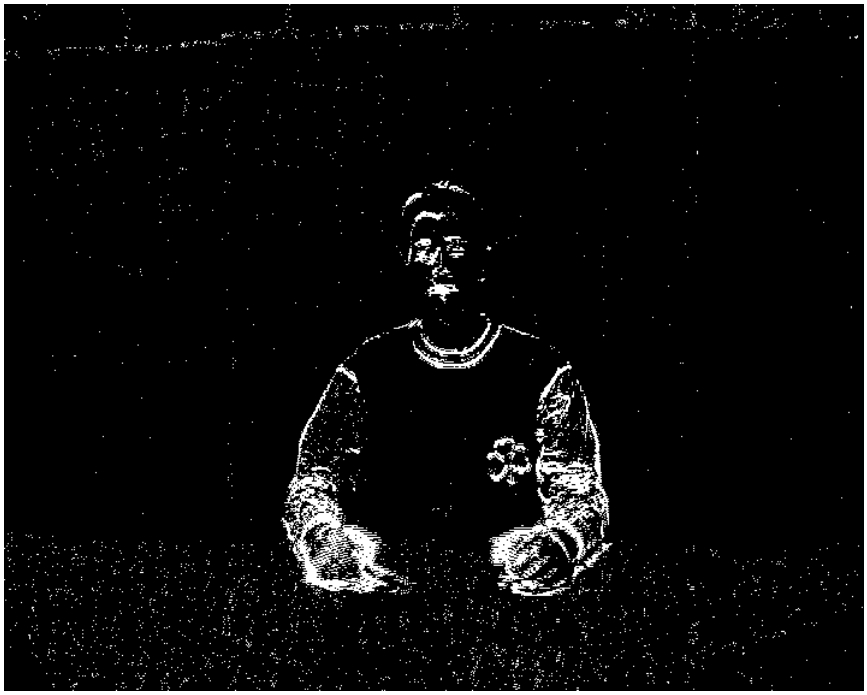
$$Motion(i,j) = \left|\mathrm{Im}age1(i,j) - \mathrm{Im}age2(i,j)\right| \qquad (7)$$

Εικόνα 9: Εικόνα του πίνακα διαφοράς μεταξύ των εικόνων 7 και 8

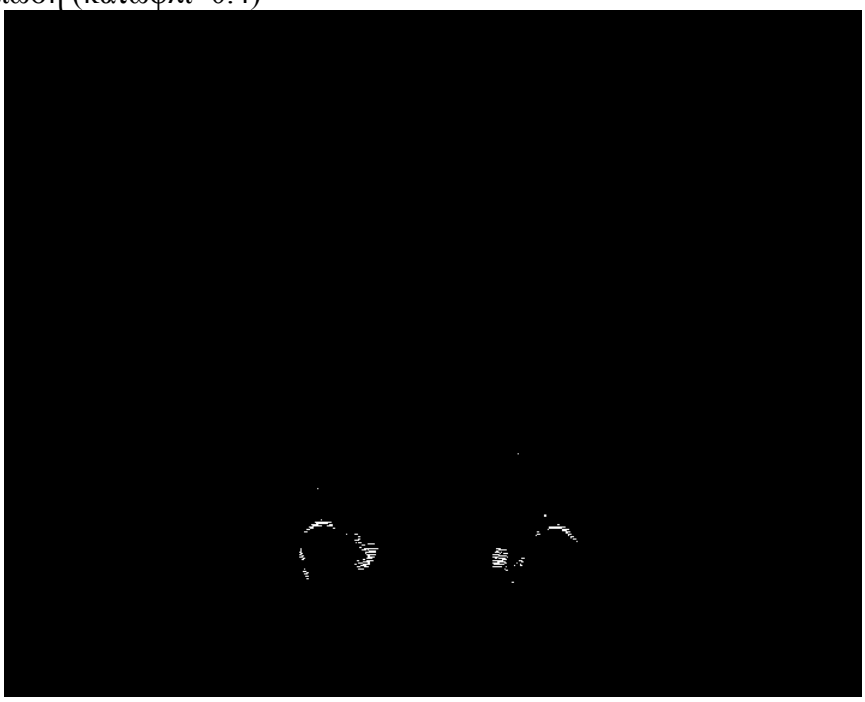
Στην συνέχεια επιλέγουμε το σύνολο των τιμών που βρίσκονται πάνω από ένα κατώφλι. Το κατώφλι αυτό επιλέχτηκε να είναι η τιμή 0.1 μιας και είχε ικανοποιητικά αποτελέσματα αλλά εφαρμόζεται και προτείνεται και στην βιβλιογραφία.

Εικόνα 10: Εικόνα του πίνακα διαφοράς μεταξύ των δύο εικόνων (6 και 7) μετά από κατωφλίωση (κατώφλι=0.1)

Εικόνα 11: Εικόνα του πίνακα διαφοράς μεταξύ των δύο εικόνων (6 και 7) μετά από κατωφλίωση (κατώφλι=0.05)



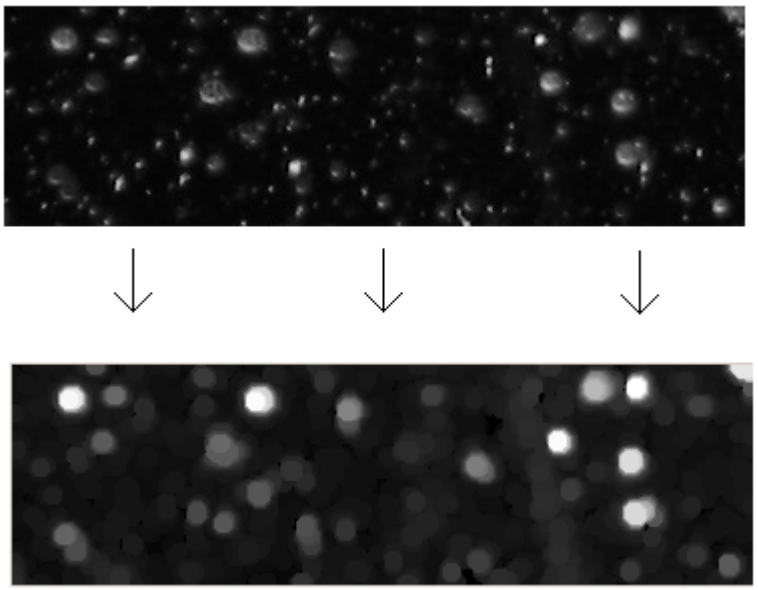
Εικόνα 12: Εικόνα του πίνακα διαφοράς μεταξύ των δύο εικόνων (6 και 7) από κατωφλίωση (κατώφλι=0.4)



Μετά την δημιουργία της μάσκας, χρησιμοποιώντας το κατώφλι, εφαρμόζουμε μορφολογικά φίλτρα προκειμένου να κάνουμε συμπαγείς τις περιοχές που εμφανίζονται και να τις χωρίσουμε έτσι σε ενιαία κομμάτια προκειμένου να τα επεξεργαστούμε μετά σαν ενιαία αντικείμενα. Αρχικά κάνουμε μορφολογικό κλείσιμο οι ιδιότητες του οποίου έχουν αναφερθεί παραπάνω. Έτσι παίρνουμε την παρακάτω μάσκα

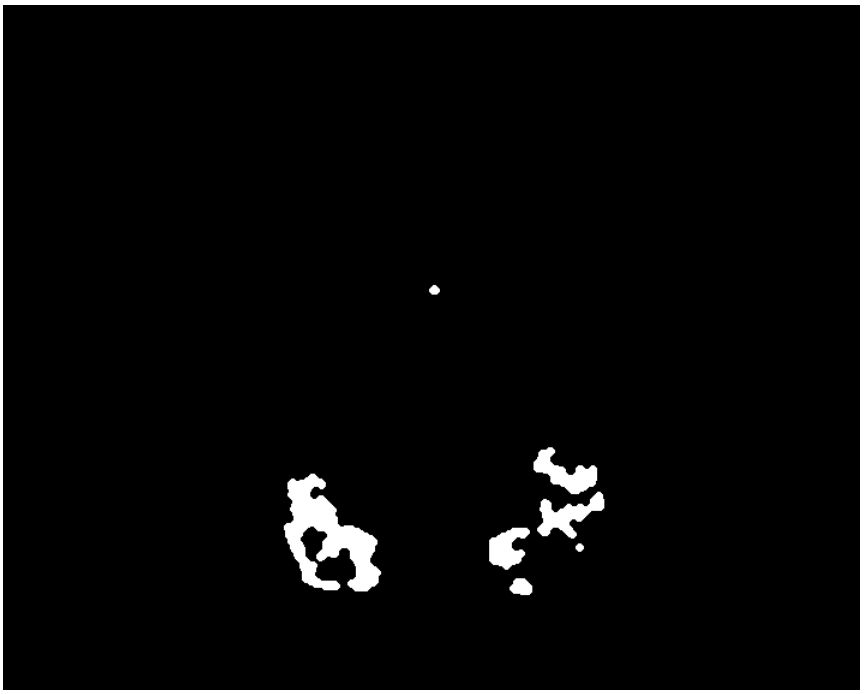
Εικόνα 13: Εικόνα του κατωφλιωμένου πίνακα μετά από μορφολογικό κλείσιμο

Στην συνέχεια πραγματοποιούμε μορφολογικό άνοιγμα το οποίο βοηθάει στη απομάκρυνση ανεπιθύμητων μικρών σημείων που για την εφαρμογή θεωρούνται άχρηστος θόρυβος (παράσιτα). Η διαδικασία του μορφολογικού ανοίγματος υλοποιείται από το OpenCV. Η διαδικασία λειτουργεί απομονώνοντας μικρές περιοχές μέσα στις οποίες αγνοεί ορισμένα στοιχεία τα οποία θεωρεί ανεπιθύμητα. Ένα κατατοπιστικό παράδειγμα λειτουργίας της είναι το ακόλουθο.



Η εφαρμογή αυτής της διαδικασίας στην κατωφλιωμένη μάσκα έχει το εξής αποτέλεσμα

Εικόνα 14: Εικόνα του κατωφλιωμένου πίνακα μετά από μορφολογικό κλείσιμο και μετά μορφολογικό άνοιγμα



3.3 Συμπεράσματα-Προβλήματα

Έτσι φτάνουμε να έχουμε μια μάσκα με αντικείμενα τα οποία απεικονίζουν την κίνηση που πραγματοποιήθηκε μεταξύ των δύο εικόνων που μελετάμε. Η μάσκα αυτή είναι η μάσκα κίνησης. Απώτερος σκοπός μας όμως είναι να βρούμε το κινούμενο δέρμα. Με την μάσκα δέρματος και την μάσκα κίνησης θα πετύχουμε αυτόν τον στόχο. Στην προσπάθεια εντοπισμού της κίνησης έχουμε τις εξής δυσκολίες: θόρυβος, κίνηση αντικειμένων στο χώρο, αλλαγή φωτισμού και κακής ποιότητας του μέσου λήψης εικόνας. Η οποιαδήποτε μορφή θορύβου μπορεί να θεωρηθεί ως κίνηση με αποτέλεσμα να μπερδέψει την όλη διαδικασία εντοπισμού της κίνησης. Μια μετακίνηση της κάμερας ή μια μετακίνηση του ατόμου που χειρονομεί μπορεί να προκαλέσει τεράστια προβλήματα στον εντοπισμό κίνησης. Η κίνηση αντικειμένων στο χώρο είναι ένα άλλο πρόβλημα με σοβαρές επιπτώσεις στον εντοπισμό κίνησης. Εάν κατά την διάρκεια της λήψης του βίντεο περάσει κάποιο αντικείμενο ή κάποιος άνθρωπος μπροστά από την κάμερα, ή κάποιο αντικείμενο με κινητά μέρη κινηθεί, αυτό εντοπίζεται ως κίνηση από το πρόγραμμα και μπορεί να επηρεάσει σημαντικά τα αποτελέσματα. Παράδειγμα αποτελεί ένας εκτυπωτής. Αν κατά την διάρκεια της ομιλίας εκτυπωθεί ένα χαρτί σε αυτόν (το οποίο χρωματικά βρίσκεται κοντά στο δέρμα) μπορεί να έχουμε έναν λανθασμένο εντοπισμό κίνησης. Ακόμα η αλλαγή φωτισμού έχει καταστροφικές συνέπειες. Σχεδόν όλα τα pixels της εικόνας αλλάζουν κατά πολύ. Η μέθοδος εντοπισμού κίνησης που χρησιμοποιείται θα μου δείξει ότι όλα τα pixel κινήθηκαν. Έτσι δεν υπάρχει περίπτωση να εντοπιστεί ούτε το κινούμενο δέρμα, μιας και αυτό θα έχει χαθεί, δεδομένου ότι άλλαξε ο φωτισμός. Επομένως ο φωτισμός πρέπει να παραμένει σταθερός συνεχώς

προκειμένου να μην αλλοιώνονται τα αποτελέσματα. Τέλος ένα κακής ποιότητας μέσο λήψης εικόνας έχει ως αποτέλεσμα την εισαγωγή θορύβου και κατά συνέπεια την αλλοίωση των αποτελεσμάτων με την εισαγωγή ανύπαρκτης κίνησης. Ο ταυτόχρονος εντοπισμός κίνησης και δέρματος λύνει μερικά από τα παραπάνω προβλήματα.

Κ Ε Φ Α Λ Α Ι Ο 4

Εντοπισμός Δερματικής Κίνησης

4.1 Εισαγωγικά

Στα προηγούμενα δύο κεφάλαια αναλύθηκε εκτενώς τι είναι και πώς γίνεται ο εντοπισμός δέρματος και ο εντοπισμός κίνησης. Σε αυτό το κεφάλαιο αναλύεται το πώς έχοντας εντοπίσει την κίνηση και το δέρμα θα εντοπιστεί η δερματική κίνηση και κατ' επέκταση οι χειρονομίες σε ένα βίντεο.

4.2 Μάσκα δερματικής κίνησης.

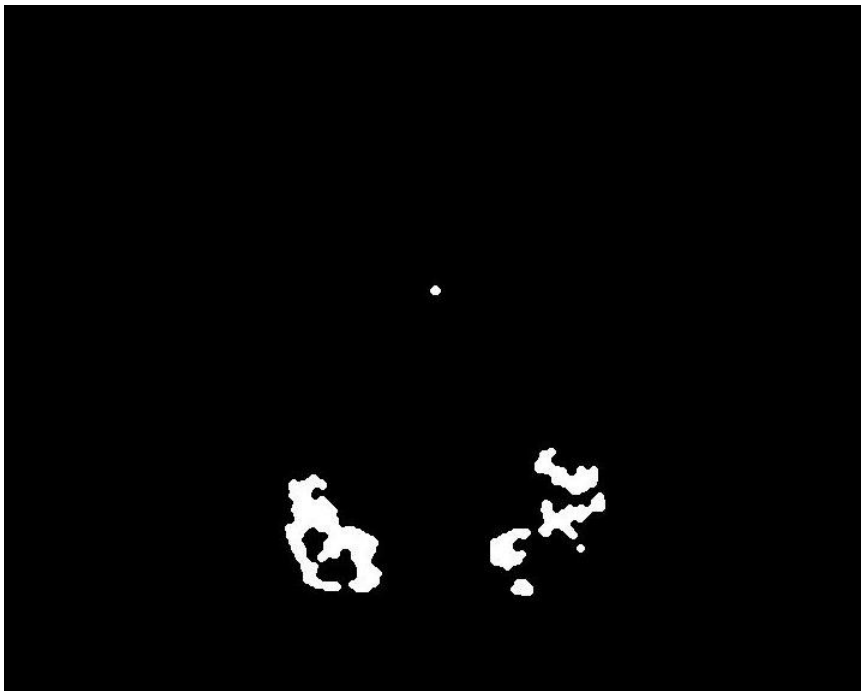
Έχοντας υπολογίσει την μάσκα δέρματος και την μάσκα κίνησης των αντικειμένων στην εικόνα μπορούμε να υπολογίσουμε την μάσκα δερματικής κίνησης. Η μάσκα αυτή θα περιέχει αντικείμενα που θα ανήκουν στην δερματική μάσκα αλλά ταυτόχρονα θα πρέπει κάποιο κομμάτι τους να ανήκει και στην μάσκα κίνησης ώστε να είναι κινούμενο δέρμα. Η δημιουργία αυτής της μάσκας γίνεται με χρήση της μάσκας δέρματος και της μάσκας κίνησης και επανακατασκευάζοντας μια νέα μάσκα που θα είναι η μάσκα δερματικής κίνησης και θα πληρεί τις παραπάνω προϋποθέσεις. Η συνάρτηση που χρησιμοποιείται για την δημιουργία της μάσκας δερματικής κίνησης ονομάζεται Imreconstruct. Δέχεται δύο μάσκες σαν ορίσματα. Η πρώτη μάσκα περιέχει ολόκληρα τα αντικείμενα τα οποία θα υπάρχουν η όχι στην νέα μάσκα. Το δεύτερο όρισμα είναι μια μάσκα η οποία θα καθορίσει το αν ένα αντικείμενο της μάσκας του πρώτου ορίσματος θα υπάρχει στην τελική μάσκα η όχι. Οι μάσκες των δύο ορισμάτων πρέπει να είναι των ίδιων διαστάσεων. Η επιλογή των αντικειμένων γίνεται ως εξής: Έστω Α η μάσκα του πρώτου ορίσματος (μάσκα δέρματος) , Β η μάσκα του δεύτερου ορίσματος (μάσκα κίνησης) και έστω C η μάσκα που επιστρέφει η διαδικασία δημιουργίας της μάσκας δερματικής κίνησης. Ένα αντικείμενο της μάσκας Α θα υπάρχει στην Μάσκα C μόνο και μόνο αν υπάρχει τουλάχιστον ένα σημείο του αντικειμένου της μάσκας Α με συντεταγμένες i,j για το οποίο να ισχύουν τα εξής :

$A(i,j) <> 0 \ \& \ B(i,j)<>0 \quad (8)$

Το επόμενο παράδειγμα δείχνει τα αποτελέσματα της διαδικασίας.

Εικόνα 15: Εικόνα της μάσκα δέρματος (Όρισμα $1^{ο}$)

Εικόνα 16: Εικόνα της μάσκα κίνησης (Όρισμα $2^{ο}$)



Εικόνα 17: Εικόνα του αποτελέσματος εφαρμογής της επανακατασκευής

4.3 Εντοπισμός συντεταγμένων Χεριών και Κεφαλιού

Το επόμενο βήμα είναι να εντοπιστούν οι συντεταγμένες των χεριών και του κεφαλιού. Για να γίνει αυτό αρχικά θα πρέπει από τα αντικείμενα που απαρτίζουν την δερματική μάσκα να προσδιοριστεί ποιο είναι το κεφάλι και τα χέρια και ποια δεν πρέπει να χρησιμοποιηθούν. Στην συνέχεια θα βρούμε τις συντεταγμένες τους και με βάση την μάσκα δερματικής κίνησης θα δούμε αν πραγματοποιούν κάποια κίνηση, οπότε και να αλλάξουμε τις συνταγμένες του κέντρου τους με τις νέες συντεταγμένες τους. Οι συντεταγμένες του κέντρου ενός αντικειμένου βρίσκονται ως εξής.

$$X_{Center} = \frac{1}{N}\sum_{i=1}^{N} x(i) \qquad (9)$$

όπου N το πλήθος των σημείων του αντικειμένου και x(i) η x συντεταγμένη του i-οστού σημείου του αντικειμένου.

$$Y_{Center} = \frac{1}{N}\sum_{i=1}^{N} y(i) \qquad (10)$$

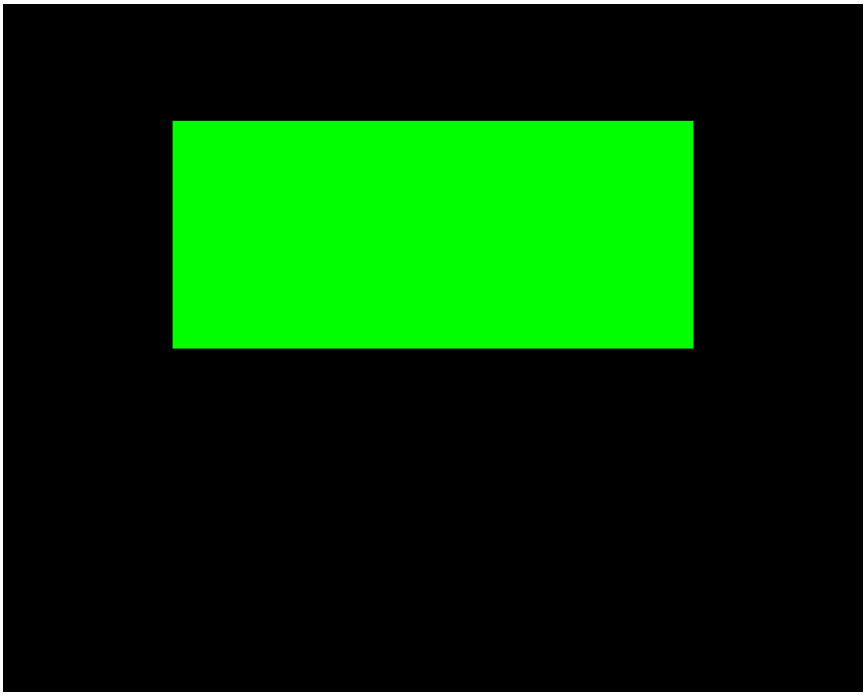
όπου N το πλήθος των σημείων του αντικειμένου και y(i) η y συντεταγμένη του i-οστού σημείου του αντικειμένου. Έχοντας υπολογίσει τις συντεταγμένες του κάθε αντικειμένου που θεωρείται δέρμα μπορούμε να κάνουμε ορισμένους λογικούς ελέγχους προκειμένου να προσδιορίσουμε ποια είναι αυτά που μας ενδιαφέρουν. Να

σημειωθεί ότι σκοπός μας είναι να εντοπίσουμε το κεφάλι και τα χέρια σε μια σειρά συγκεκριμένων κινήσεων και όχι σε οποιαδήποτε στάση πάρει το άτομο (αν π.χ. ο χρήστης βάλει το κεφάλι του στο κάτω μέρος της εικόνας η κίνηση αυτή δεν θεωρείται αποδεκτή και κατά συνέπεια το πρόγραμμα δεν θα εντοπίσει την κίνηση).

4.3.1 Προσδιορισμός συντεταγμένων κεφαλιού.

Το κεφάλι θα πρέπει να βρίσκεται στο πάνω μέρος της εικόνας και στο μέσο αυτής. Οι λογικοί έλεγχοι που γίνονται για τον προσδιορισμό του κεφαλιού είναι οι εξής: Η X συντεταγμένη θα πρέπει να βρίσκεται πάνω από ½ του ύψους της εικόνας και να είναι τουλάχιστον ίση με το 1/5 του ύψους κάτω από την κορυφή της εικόνας. Η Y συντεταγμένη θα πρέπει να βρίσκεται μεταξύ του 2/5 και 3/5 του πλάτους της εικόνας. Οι τιμές αυτές προτείνονται από την βιβλιογραφία και τις δημοσιεύσεις που έχουν αναφερθεί παραπάνω. Η μάσκα μέσα στην οποία μπορούν να κινούνται οι συντεταγμένες του κεφαλιού είναι η εξής.

Εικόνα 18: Το πεδίο μέσα στο οποίο μπορεί να κινείται το κεφάλι.



Η εύρεση των αντικείμενων που αντιστοιχούν στα χέρια είναι πιο πολύπλοκη διαδικασία και χρησιμοποιεί και τις συντεταγμένες του κεφαλιού προκειμένου να μας δώσει τις συντεταγμένες των χεριών. Τα χέρια κινούνται πιο πολύ από ότι το κεφάλι και ως αριστερό χέρι θεωρούμε αυτό που βρίσκεται αριστερά του κεφαλιού και δεξιό αυτό που βρίσκεται δεξιά του κεφαλιού.

4.3.2 Προσδιορισμός δεξιού χεριού.

Από όλα τα αντικείμενα της μάσκας αυτό που βρίσκεται δεξιά του κεφαλιού και είναι τουλάχιστον κάτω από 1/5 του ύψους της εικόνας είναι το αντικείμενο που αντιστοιχεί στο δεξί χέρι.

4.3.3 Προσδιορισμός αριστερού χεριού.

Από όλα τα αντικείμενα της μάσκας αυτό που βρίσκεται αριστερά του κεφαλιού και είναι τουλάχιστον κάτω από 1/5 του ύψους της εικόνας είναι το αντικείμενο που αντιστοιχεί στο αριστερό χέρι.

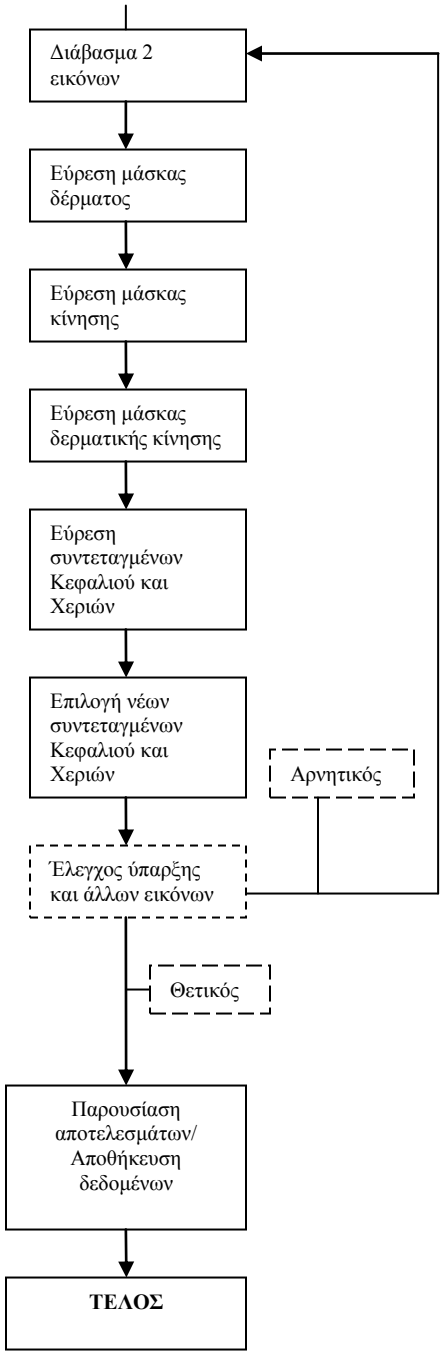
Έχοντας υπολογίσει τώρα τις θέσεις των χεριών και του κεφαλιού από την μάσκα δερματικής κίνησης τις συγκρίνουμε με τις συντεταγμένες που είχαν αυτές από την προηγούμενη εφαρμογή του παραπάνω εντοπισμού. Αν πληρούνται οι παρακάτω συνθήκες, οι νέες συντεταγμένες που υπολογίστηκαν θεωρούνται οι νέες συντεταγμένες των χεριών και του κεφαλιού αλλιώς θεωρούμε ότι δεν υπήρξε κίνηση . Το μέλος που δεν παρουσίασε κίνηση κρατάει τις συντεταγμένες που είχε πριν την διαδικασία εντοπισμού κίνησης. Έτσι κάτι που έμεινε ακίνητο δεν θα αλλάξει συντεταγμένες.

Συνθήκες αποδοχής νέων συντεταγμένων: $\left|X_{New} - X_{Old}\right| > U \;\&\; \left|Y_{New} - Y_{Old}\right| > U$ (11)

όπου, $U = \frac{1}{10}\sqrt{(\mathrm{Im}ageHeight)^2 + (\mathrm{Im}ageWidht)^2}$ (12).

Το U προτείνεται από την βιβλιογραφία ως ένα κατώφλι με το οποίο οποιαδήποτε μετακίνηση συντεταγμένων, μεγαλύτερη από U, θεωρείται κίνηση. Η συνθήκες αυτές εφαρμόζονται για τα X και Y του κεφαλιού και των δύο χεριών.

4.4 Μπλοκ-Διάγραμμα αλγορίθμου εντοπισμού δερματικής κίνησης



Κ Ε Φ Α Λ Α Ι Ο 5

Πλατφόρμα OpenCV

5.1 Γενικά στοιχεία

Το openCV είναι μια σειρά συναρτήσεων και νέων δομών δεδομένων για την καλύτερη επεξεργασία εικόνας και βίντεο. Βασιζόμενοι σε αυτήν την πλατφόρμα αναπτύσσουμε τα προγράμματα που χρειάζονται για τον εντοπισμό της δερματικής κίνησης. Είναι μια εφαρμογή ανοικτού κώδικα φτιαγμένη από την Intel.Η γλώσσα προγραμματισμού στην οποία είναι γραμμένες οι συναρτήσεις και οι δομές της OpenCV είναι η C++. Περιέχει μια σειρά από μαθηματικές συναρτήσεις εφαρμοζόμενες πάνω σε πίνακες επιτρέποντας έτσι την γρήγορη επεξεργασία των εικόνων.

5.2 Χρησιμοποιούμενες δομές

Μερικές βασικές δομές τις οποίες χρησιμοποιούμε στο πρόγραμμα είναι:

1) IplImage= Πίνακας στον οποίο αποθηκεύονται εικόνες. Μπορεί να έχει περισσότερα του ενός κανάλια η εικόνα .

2) CvSize= Δομή για την αποθήκευση του μεγέθους των εικόνων

3)CvScalar= Δομή που έχει 4 μεταβλητές double μια για κάθε κανάλι μιας εικόνας.

Για περισσότερες πληροφορίες για το OpenCV υπάρχει εκτενής βιβλιογραφία στην ιστοσελίδα http://cgi.cs.indiana.edu/~oleykin/website/OpenCVHelp/

.

Κ Ε Φ Α Λ Α Ι Ο 6

Συναρτήσεις προγράμματος σε OpenCV

6.1 Δημιουργηθείσες συναρτήσεις

1. cvNormal01
2. cvRGB2YCbCrChanelY
3. cvRGB2YCbCrChanelCr
4. cvRGB2YCbCrChanelCb
5. cvGetSkinProps
6. cvobjxy
7. cvFindMovingSkin
8. cvProcSkinDetection
9. cvProcessImgSeq
10. ShowSkinMovementOnImgManual
11. Bring3digitsBack
12. DrawR
13. DrawL
14. DrawH
15. cvTakeRowFromImage
16. cvMakeRowImage
17. cvAddColumToImage
18. cvMakeImageCol
19. cvImReconstruct
20. cvBwLabel
21. cvMakeMyCloseFriendLikeMe
22. MakeRoiPOly
23. MakeEveryKToG

24. on_mouse

25. cvMeanM

6.2 Κατηγοριοποίηση των συναρτήσεων

6.2.1 Συναρτήσεις χρησιμοποιούμενες για τον εντοπισμό δέρματος

cvNormal01

cvRGB2YCbCrChanelY

cvRGB2YCbCrChanelCr

cvRGB2YCbCrChanelCb

cvGetSkinProps

cvprocskindetection

6.2.2 Συναρτήσεις χρησιμοποιούμενες για τον εντοπισμό κίνησης

cvFindMovingSkin

cvProcessImgSeq

6.2.3 Συναρτήσεις χρησιμοποιούμενες για εντοπισμό δερματικής κίνησης

cvGetSkinProps

cvobjxy

cvFindMovingSkin

cvProcSkinDetection

cvProcessImgSeq

ShowSkinMovementOnImgManual

6.2.4 Συναρτήσεις επεξεργασίας εικόνων

cvTakeRowFromImage

cvMakeRowImage

cvAddColumToImage

cvMakeImageCol

cvImReconstruct

cvBwLabel

cvMakeMyCloseFriendLikeMe

MakeRoiPOly

MakeEveryKToG

6.3 Επεξήγηση των συναρτήσεων

1) CvNormal01

IplImage* cvNormal01 (IplImage* x, IplImage* y, double mx, double my, double sx, double sy)

Η συνάρτηση cvNormal01 επιστρέφει έναν double πίνακα ιδίων διαστάσεων με αυτές των frame του βίντεο (ιδίων διαστάσεων με αυτές των πινάκων x και y όπου τα x και y είναι ο Cb και Cr πίνακας αντίστοιχα) ο οποίος περιέχει τις πιθανότητες το pixel να είναι δέρμα. Κάθε στοιχείο αυτού του πίνακα είναι θετικό και μικρότερο του ένα. Ο υπολογισμός γίνεται χρησιμοποιώντας τα mx, sx, sy και my και υπολογίζουμε με πράξεις πινάκων την τελική πιθανότητα υψώνοντας σε αρνητική εκθετική δύναμη το τελικό αποτέλεσμα προκειμένου να είμαστε εντός των περιορισμών που έχουμε λόγω του ότι το αποτέλεσμα πρέπει να είναι στην περιοχή [0,1]. Αυτή η συνάρτηση είναι καθοριστική μιας και μέσω αυτής μπορούμε να εντοπίσουμε το δέρμα με ικανοποιητική ακρίβεια και χρησιμοποιείται πολύ στο πρόγραμμα.

IplImage* x =ο πίνακας Cb της εικόνας.

IplImage* y = ο πίνακας Cr της εικόνας.

double mx= μεταβλητή που υπολογίζεται από την συνάρτηση

CvGetSkinProps.

double my= μεταβλητή που υπολογίζεται από την συνάρτηση

CvGetSkinProps.

double sx=0.001 συνήθως.

double sy=0.001 συνήθως.

2) CvRGB2YCbCrChanelY

IplImage* cvRGB2YCbCrChanelY (IplImage* X)

Η συνάρτηση cvRGB2YCbCrChanelY παίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει το κανάλι Y. Χρησιμοποιείται ο γνωστός τρόπος μετατροπής μιας εικόνας από RGB σε YCbCr. Χρησιμοποιείται ο πίνακας τρία επί τρία TAB ο οποίος ουσιαστικά είναι η μήτρα μετατροπής από την βάση RGB στην βάση YCbCr και με ένα πολλαπλασιασμό πινάκων επιτυγχάνουμε αυτήν την μετατροπή. Συγκεκριμένα για τον υπολογισμό του καναλιού Y χρησιμοποιούμε την πρώτη γραμμή του πίνακα TAB. Επίσης διαιρώντας με 255 κανονικοποιούμε τα αποτελέσματα και τα κάνουμε μορφής double.

IplImage* X = RGB εικόνα τριών καναλιών.

TAB=

0.299	0.587	0.114
-0.169	-0.331	0.5
0.5	-0.419	-0.081

3)cvRGB2YCbCrChanelCr

IplImage* cvRGB2YCbCrChanelCr(IplImage* X)

Η συνάρτηση cvRGB2YCbCrChanelCr παίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει το κανάλι Cr. Χρησιμοποιείται ο γνωστός τρόπος μετατροπής μιας εικόνας από RGB σε YCbCr. Χρησιμοποιείται ο πίνακας τρία επί τρία TAB ο οποίος ουσιαστικά είναι η μήτρα μετατροπής από την βάση RGB στην βάση YCbCr και με ένα πολλαπλασιασμό πινάκων επιτυγχάνουμε αυτήν την μετατροπή. Συγκεκριμένα για τον υπολογισμό του καναλιού Y χρησιμοποιούμε την δεύτερη γραμμή του πίνακα TAB. Επίσης διαιρώντας με 255 κανονικοποιούμε τα αποτελέσματα και τα κάνουμε μορφής double.

IplImage* X = RGB εικόνα τριών καναλιών.

TAB=

0.299	0.587	0.114
-0.169	-0.331	0.5
0.5	-0.419	-0.081

4) cvRGB2YCbCrChanelCb

IplImage* cvRGB2YCbCrChanelCb (IplImage* X)

Η συνάρτηση cvRGB2YCbCrChanelCbπαίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει το κανάλι Cb.Χρησιμοποιείται ο γνωστός τρόπος μετατροπής μιας εικόνας από RGB σε YCbCr. Χρησιμοποιείται ο πίνακας τρία επί τρία TAB ο οποίος ουσιαστικά είναι η μήτρα μετατροπής από την βάση RGB στην βάση YCbCr και με ένα πολλαπλασιασμό πινάκων επιτυγχάνουμε αυτήν την μετατροπή. Συγκεκριμένα για τον υπολογισμό του καναλιού Y χρησιμοποιούμε την τρίτη γραμμή του πίνακα TAB. Επίσης διαιρώντας με 255 κανονικοποιούμε τα αποτελέσματα και τα κάνουμε μορφής double.

IplImage* X = RGB εικόνα τριών καναλιών.

TAB=

0.299	0.587	0.114
-0.169	-0.331	0.5
0.5	-0.419	-0.081

5) cvGetSkinProps

IplImage* cvGetSkinProps (IplImage* X, double* zhue, double* zsat, double* zval, double* mx, double* my, double* mhue, double* msat, double* mval)

Η συνάρτηση cvGetSkinProps παίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει έναν double πίνακα ιδίων διαστάσεων με αυτές των frame του βίντεο (ιδίων διαστάσεων με αυτές των πινάκων x και y όπου τα x και y είναι ο Cb και Cr πίνακας αντίστοιχα) ο οποίος περιέχει τις πιθανότητες το pixel να είναι δέρμα. Κάθε στοιχείο αυτού του πίνακα είναι θετικό και μικρότερο του ένα. Η συνάρτηση αυτή επίσης υπολογίζει και τα mx και my τα οποία είναι απαραίτητα για τον υπολογισμό των πιθανοτήτων για το αν ένα pixel είναι δέρμα η όχι. Ο Υπολογισμός των mx,my γίνεται μέσω της εξής διαδικασίας. Κατά την εκτέλεση της cvGetSkinProps ζητείται από τον χρήστη να επισημάνει πάνω στην εικόνα ποια περιοχή θεωρείται δέρμα. Παίρνοντας το μέσω όρο των τιμών αυτής της περιοχής υπολογίζουμε το mx από το κανάλι Cr και το my από το κανάλι Cb.

IplImage* X = RGB εικόνα τριών καναλιών.

double* zhue →Βοηθητική μεταβλητή.

double* zsat →Βοηθητική μεταβλητή.

double* zval →Βοηθητική μεταβλητή.

double* mx →Βοηθητική μεταβλητή.

double* my →Βοηθητική μεταβλητή.

double* mhue →Βοηθητική μεταβλητή.

double* msat →Βοηθητική μεταβλητή.

double* mval →Βοηθητική μεταβλητή.

6) cvobjxy

void cvobjxy(IplImage* mask,int* xmean,int* ymean)

Η συνάρτηση cvobjxy υπολογίζει το κέντρο ενός αντικειμένου. Το αντικείμενο βρίσκεται σε ένα frame του βίντεο. Ο προσδιορισμός του αντικείμένου γίνεται από την μάσκα mask η οποία είναι ένας δισδιάστατος πίνακας 2xN όπου N το πλήθος των pixel του αντικειμένου. Το κέντρο του αντικειμένου υπολογίζεται ως το μέσο όρο των σημείων που αποτελούν το αντικείμενο ως προς το X και το μέσο όρο ως προς το Y.Τα αποτελέσματα περνάνε στις μεταβλητές xmean και ymean. Σκοπός της συνάρτησης είναι να

μας δίνει το κέντρο των αντικειμένων μιας εικόνας (αντικείμενα θεωρούνται τα μη μηδενικά pixel της εικόνας) και πιο συγκεκριμένα για το πρόγραμμα μου μας ενδιαφέρει το κέντρο του κεφαλιού και των δύο χεριών. Το κέντρο των αντικειμένων είναι απαραίτητο μιας και με αυτό κάνουμε ορισμένους λογικούς ελέγχους σε άλλες συνάρτισεις προκείμενου να προσδιορίσουμε πιο κέντρο αντιστοιχεί στο κεφάλι ποιο στο δεξί χέρι και πιο στο αριστερό χέρι.

IplImage* mask = Η μάσκα που καθορίζει το ποια σημεία ανήκουν στο αντικείμενο.

int* xmean =Η μια συντεταγμένη του κέντρου του αντικειμένου.

int* ymean= Η δεύτερη συντεταγμένη του κέντρου του αντικειμένου.

7) cvFindMovingSkin

void cvFindMovingSkin(IplImage* ImgA,IplImage* ImgB,double mx,double my,IplImage* ms,IplImage* cmout,IplImage* mmout)

Η συνάρτηση cvFindMovingSkin υπολογίζει τρεις πολύ βασικούς πίνακες καθοριστικής σημασίας για τον εντοπισμό του δέρματος αλλά και της κίνησης. Όι πίνακες αυτοί είναι λογικοί πίνακες(περιέχουν άσους και μηδενικά). Ο ένας περιέχει την κίνηση που παρατηρήθηκε μεταξύ των δύο εικόνων ο άλλος περιέχει της περιοχές που θεωρούνται δέρμα και ο τρίτος την κίνηση του δέρματος που παρουσιάστηκε ανάμεσα στις δυο εικόνες. Η συνάρτηση αυτή είναι η πιο σημαντική συνάρτηση του προγράμματος μου μιας και χρησιμοποιεί σχεδόν όλες τις υπόλοιπες και βγάζει τα πιο ουσιαστικά αποτελέσματα μέσα από τα οποία παίρνουμε τα τελικά συμπεράσματά μας.

IplImage* ImgA= Το πρώτο frame.

IplImage* ImgB= Το δεύτερο frame.

double mx = μεταβλητή που υπολογίζεται από την cvGetSkinProps.

double my = μεταβλητή που υπολογίζεται από την cvgetskinprops.

IplImage* ms = πίνακας κινούμενου δέρματος.

IplImage* cmout = πίνακας που εντοπίζει που υπάρχει δέρμα.

IplImage* mmout = πίνακας που εντοπίζει που γίνεται κίνηση.

8) cvProcSkinDetection

IplImage* cvProcSkinDetection(IplImage* ms,IplImage* cmout,IplImage* mmout,IplImage* prevcoords)

Η συνάρτηση cvProcSkinDetection είναι αυτή που καθορίζει μεταξύ δύο εικόνων ποια θα είναι η νέα θέση του κεφαλιού και των δύο χεριών. επιστρέφει έναν μονοδιάστατο πίνακα έξι θέσεων όπου ανα δυο βρίσκονται οι συντεταγμένες των τριών βασικών σημείων. Χρησιμοποιούνται οι βασικοί πίνακες που έχουν υπολογιστεί από την cvFindMovingSkin και με μερικούς λογικούς ελέγχους καταλήγουμε να αποφανθούμε για το ποια από τα υπάρχοντα αντικείμενα είναι αυτά που μας ενδιαφέρουν και αντίστοιχα επιλέγουμε τις συντεταγμένες τους και της επιστρέφει η συνάρτηση σε έναν πίνακα ώστε να χρησιμοποιηθούν παρακάτω στο πρόγραμμα.

IplImage* ms = πίνακας κινούμενου δέρματος.

IplImage* cmout = πίνακας που εντοπίζει που υπάρχει δέρμα.

IplImage* mmout = πίνακας που εντοπίζει που γίνεται κίνηση.

IplImage* prevcoords = Η συνταταγμένες του κεφαλιού και των δύο χεριών πριν την εκτέλεση της cvProcSkinDetection.Αν δεν έχει εκτελεσθεί καμία φορά η cvProcSkinDetection τότε η prevcoords περιέχει άσους.

9) cvProcessImgSeq

IplImage* cvProcessImgSeq(char MainImgName[],int NumberOfImgs,char ImgType[])

Η συνάρτηση cvProcessImgSeq είναι η συνάρτηση μέσω της οποίας όλες οι συναρτήσεις του προγράμματός μου συνεργάζονται προκείμενα να πάρουμε το επιθυμητό αποτέλεσμα. Ουσιαστικά εργαζόμαστε ανα δύο εικόνων με όλα τα frame του βίντεο και εντοπίζουμε την κίνησης του δέρματος που παρατηρείται σε αυτά τα ζευγάρια και παίρνουμε τα αποτελέσματα. Η συνάρτηση επιστρέφει έναν πίνακα διαστάσεων 6 επί το πλήθος των ζευγαριών εικόνων τα οποία επεξεργαζόμαστε και ο οποίος περιέχει τα κέντρα του κεφαλιού και των δύο χεριών ανά κάθε ζευγάρι εικόνων. Αυτό γίνεται γιατί μετά το τέλος αυτής της συνάρτησης θα απεικονίσουμε τα αποτελέσματα και γι' αυτό χρειαζόμαστε αυτόν τον πίνακα. Επίσης δίνουμε στην συνάρτηση το όνομα και τον τύπο των εικόνων του το οποίο θέλουμε να επεξεργαστούμε καθώς και το πλήθος αυτών των εικόνων.

char MainImgName[] = Το κυρίως όνομα των εικόνων.

int NumberOfImgs= Το πλήθος των εικόνων.

char ImgType[]= Ο τύπος των αρχείων.

10) ShowSkinMovementOnImgManual

void ShowSkinMovementOnImgManual(IplImage* Coords,char MainImgName[],char ImgType[],bool SaveFile)

Η συνάρτηση ShowSkinMovementOnImgManual χρησιμοποιείται προκειμένου να δείξουμε τα αποτελέσματα της cvProcessImgSeq (και γενικότερα του εντοπισμού κίνησης του δέρματος) αλλά και αν επιθυμούμε να τα αποθηκεύσουμε. Τα αποτελέσματα είναι ο εντοπισμός σε κάθε frame του πού βρίσκεται το Κεφάλι το Αριστερό και το Δεξί χέρι. Η απεικόνιση των αποτελεσμάτων γίνεται πάνω σε μια GrayScale εικόνα της αρχικής όπου πάνω στο κέντρο του κάθε μέλους έχει δημιουργηθεί ένα γράμμα το οποίο προσδιορίζει την θέση αυτού του μέλους και έχουμε την δυνατότητα με αυτό τον τρόπο να διαπιστώσουμε αν ο εντοπισμός δερματικής κίνησης που έχει γίνει είναι σωστός βλέποντας στην πράξη αν όντος το Κεφάλι συμπίπτει με τις αντίστοιχες συντεταγμένες για το κεφάλι που δίνει το πρόγραμμα και αντίστοιχα τα χέρια.

IplImage* Coords = Πίνακας που περιέχει τις συντεταγμένες κεφαλιού και χεριών απο προηγούμενη εκτέλεση της συνάρτησης (αρχική τιμή είναι ο πίνακας γεμάτος με άσους)

char MainImgName[] = Το κυρίως όνομα των εικόνων.

char ImgType[]= Ο τύπος των αρχείων.

bool SaveFile = οταν είναι true έχουμε αποθήκευση των αποτελεσμάτων.

11) Bring3digitsBack

void Bring3digitsBack(int InputNumber,char* C1,char* C2,char* C3)

Η συνάρτηση αυτή δέχεται σαν όρισμα έναν ακέραιο αριθμό μικρότερο του χίλια και επιστρέφει σε 3 χαρακτήρες τα 3 ψηφία του αριθμού. Η συνάρτηση αυτή χρησιμοποιείται οπουδήποτε θέλω να έχω μια αρίθμηση των εικόνων ώστε να τα επεξεργαστώ από την αρχή μέχρι το τέλος η μέχρι κάποιον αριθμό που ορίζει ο χρήστης. Η αντίστοιχη διαδικασία μπορεί να γίνει και για μεγαλύτερους του χίλια αριθμούς αλλά δεν το έχω υλοποιήσει μιας και θεώρησα ότι χίλια εικόνων είναι ένας ικανοποιητικός αριθμό. Χρησιμοποιείται από τις συναρτήσεις ShowSkinMovementOnImgManual και cvProcessImgSeq μιας και αυτές οι συναρτήσεις επεξεργάζονται μια σειρά από εικόνων και απαιτείται να μπορούν να τα επεξεργάζονται με την κατάδηλη αρίθμηση.

int InputNumber = αριθμός ο οποίος σπάει σε 3 ψηφία

char* C1 = πρώτο ψηφίο.

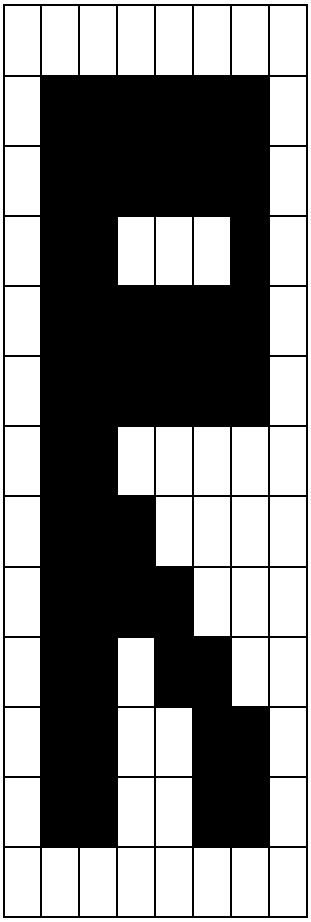
char* C2 = δεύτερο ψηφίο.

char* C3 = τρίτο ψηφίο.

12) DrawR

IplImage* DrawR(IplImage* Source,int x,int y,int colour)

Η συνάρτηση DrawR ζωγραφίζει σε μία υπάρχουσα εικόνα ένα R του οποίου η πάνω αριστερή γωνία έχει συνταγμένες αυτές που δίνονται στα ορίσματα της συνάρτησης.Το R αυτό υποδηλώνει την θέση του δεξιού χεριού. Η μορφή του R είναι η εξής:



IplImage* Source = Η εικόνα πάνω στην οποία θα σχεδιαστεί το R

int x = συνταγμένη του κέντρου του δεξιού χεριού.

int y = συνταγμένη του κέντρου του δεξιού χεριού.

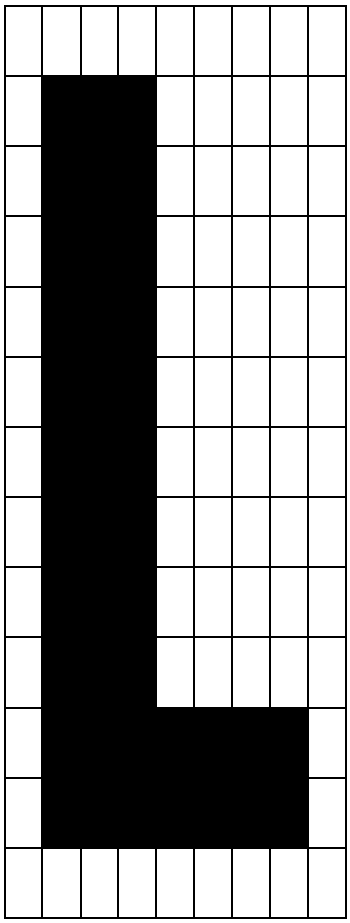
int colour = επιλογή του χρώματος.

13) DrawL

IplImage* DrawL(IplImage* Source,int x,int y,int colour)

Η συνάρτηση DrawL ζωγραφίζει σε μία υπάρχουσα εικόνα ένα L του οποίου η πάνω αριστερή γωνία έχει συνταγμένες αυτές που δίνονται στα

ορίσματα της συνάρτησης.Το L αυτό υποδηλώνει την θέση του αριστερού χεριού.Η μορφή του L είναι η εξής:



IplImage* Source = Η εικόνα πάνω στην οποία θα σχεδιαστεί το L

int x = συνταγμένη του κέντρου του αριστερού χεριού.

int y = συνταγμένη του κέντρου του αριστερού χεριού.

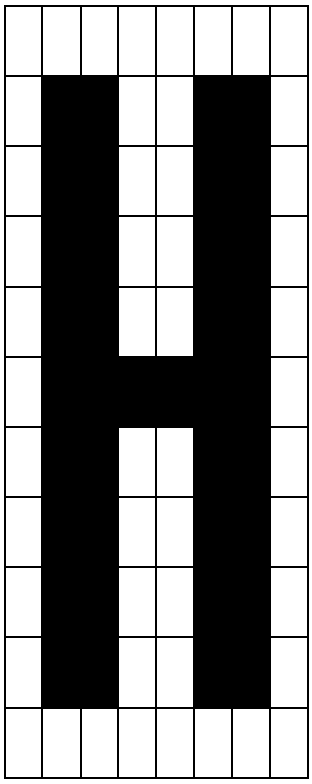
int colour = επιλογή του χρώματος.

14) DrawH

IplImage* DrawH(IplImage* Source,int x,int y,int colour)

Η συνάρτηση DrawH ζωγραφίζει σε μία υπάρχουσα εικόνα ένα Η του οποίου η πάνω αριστερή γωνία έχει συνταγμένες αυτές που δίνονται στα ορίσματα

της συνάρτησης.Το Η αυτό υποδηλώνει την θέση του κεφαλιού. Η μορφή του Η είναι η εξής:



IplImage* Source = Η εικόνα πάνω στην οποία θα σχεδιαστεί το Η

int x = συνταγμένη του κέντρου του κεφαλιού.

int y = συνταγμένη του κέντρου του κεφαλιού.

int colour = επιλογή του χρώματος.

15) cvTakeRowFromImage

IplImage* cvTakeRowFromImage(IplImage* Source,int x)

Η συνάρτηση cvTakeRowFromImage επιστρέφει έναν μονοδιάστατο πίνακα ο οποίος περιέχει μια γραμμή από τον δισδιάστατο πίνακα που έχει δοθεί ως είσοδος στην συνάρτηση. Τον αριθμό της γραμμής τον καθορίζει η άλλη μεταβλητή. Χρησιμοποιείται αρκετά αυτή η συνάρτηση στο πρόγραμμα μου μιας και το OpenCV

δεν παρέχει κάποια έτοιμη συνάρτηση που να κάνει την αντίστοιχη δουλειά. Η επιλογή μιας γραμμής από κάποιον πίνακα είναι μια πολύ βασική διεργασία στην επεξεργασία εικόνας

IplImage* Source = Η εικόνα από την οποία παίρνουμε την γραμμή.

int x = ο αριθμός της γραμμής.

16) cvMakeRowImage

IplImage* cvMakeRowImage(IplImage* Source,int i,int j)

Η συνάρτηση cvMakeRowImage παίρνει έναν μονοδιάστατο πίνακα και τον μετατρέπει σε δισδιάστατο με διαστάσεις αυτές που του έχουν δοθεί. Το OpenCV δεν παρέχει κάποια έτοιμη συνάρτηση που να κάνει την αντίστοιχη δουλειά. Η συνάρτηση αυτή χρησιμοποιείται πολύ από το πρόγραμμα μου μιας και λόγο των πολλαπλών γινόμενο πινάκων επιβάλλεται να μετατρέπω δισδιάστατους πίνακες σε μονοδιάστατους και αντίστροφα.

IplImage* Source = Μονοδιάστατος πίνακας προς μετατροπή.

int i= πλήθος γραμμών.

int j= πλήθος στηλών.

17) cvAddColumToImage

IplImage* cvAddColumToImage(IplImage* Source,IplImage* Addition)

Η συνάρτηση cvAddColumToImage προσθέτει σε έναν υπάρχοντα πίνακα μια επιπλέον στήλη. Η συνάρτηση αυτή έχει και άλλες λειτουργιές. Δεν συνδέει απαραίτητα μια μόνο στήλη. Μπορεί να συνδέσει και δύο πίνακες που απλά έχουν ίδιο αριθμό γραμμών και ο αριθμός στηλών μας είναι αδιάφορος. Το OpenCV δεν παρέχει κάποια έτοιμη συνάρτηση που να κάνει την αντίστοιχη διεργασία. Η συνάρτηση ελέγχει τους δύο πίνακες προκείμενου να πληρούν τις προϋποθέσεις ώστε να μπορέσουν να συνδεθούν. Σε περίπτωση που δεν πληρούνται οι προϋποθέσεις η συνάρτηση επιστρέφει δείκτη NULL

IplImage* Source = ο πίνακας στο δεξί μέρος του οποίου θα μπει ο πίνακας στήλη.

IplImage* Addition = ο πίνακας που θα προστεθεί στο δεξί μέρος του άλλου.

18) cvMakeImageCol

IplImage* cvMakeImageCol(IplImage* Source)

Η συνάρτηση cvMakeImageCol μετατρέπει έναν δισδιάστατο πίνακα σε μία στήλη όπου ουσιαστικά έχουμε την προσάρτηση της κάθε στήλης στην επόμενη. Το OpenCV δεν παρέχει κάποια έτοιμη συνάρτηση που να κάνει την αντίστοιχη διεργασία. Η συνάρτηση αυτή χρησιμοποιείται πολύ από το πρόγραμμα μου μιας και λόγο των πολλαπλών γινόμενο πινάκων επιβάλλεται να μετατρέπω δισδιάστατους πίνακες σε μονοδιάστατους και αντίστροφα.

IplImage* Source = ο δισδιάστατος πίνακα της εικόνας που θα μετατραπεί σε στήλη.

19) cvImReconstruct

void cvImReconstruct(IplImage* A,IplImage* B,IplImage* Destination)

Η συνάρτηση cvImReconstruct επιτελεί μια πολύ σημαντική διεργασία και είναι ένα από τα πιο βασικά σημεία στον εντοπισμό κίνησης δέρματος. Δέχεται δύο πίνακες τον Α και τον Β. Οι πίνακες είναι λογικοί(με άσους και μηδενικά). Η συνάρτηση δημιουργεί έναν νέο πίνακα ο οποίος είναι και αυτός λογικός και έχει ένα μόνο στις περιοχές όπου υπάρχουν στον Α και σε αυτές του Β που η τομή τους με μια περιοχή του Α δεν δίνει κενό σύνολο. Η σημαντικότητα της συνάρτησης έγκειται στο ότι με αυτήν την λειτουργία που εκτελεί από έναν πίνακα που εντοπίζει την κίνηση και από έναν που εντοπίζει το δέρμα μας δίνει έναν που εντοπίζει το κινούμενο δέρμα πράγμα που είναι και το ζητούμενο του όλου προγράμματος.

IplImage* A = Η εικόνα με την βασική προτεραιότητα. Οι άσοι της εικόνας αυτής θα είναι σίγουρα και στην τελική.

IplImage* B = Η εικόνα της οποία οι άσοι θα υπάρχουν στην τελική εικόνα μόνο και μόνο αν υπάρχει τομή με κάποια περιοχή του πίνακα Α.

IplImage* Destination= Ο πίνακας στο οποίο αποθηκεύονται τα αποτελέσματα της συνάρτησης.

20) cvBwLabel

void cvBwLabel(IplImage* Source,IplImage* Destination)

Η συνάρτηση αυτή παίρνει έναν πίνακα με μηδέν και ένα και επιστρέφει έναν άλλο ο οποίος αριθμεί με διαφορετικό νούμερο κάθε απομονωμένη περιοχή. Η λειτουργία αυτή είναι πολύ βασική μιας και χρησιμοποιείται για την απομόνωση του κεφαλιού και των δύο χεριών από τον υπόλοιπο χώρο. Η αρίθμηση ξεκινάει από το δύο και πάει μέχρι το πλήθος τον αντικείμενων μείον ένα. Οι απομονωμένες περιοχές εντοπίζονται με την χρήση μιας αναδρομικής συνάρτησης η οποία ξεκινώντας από ένα pixel ελέγχει όλα τα γειτονικά του μέχρι να βρει μηδέν. Παράδειγμα χρήσης συνάρτησης:

1	1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	1	1	1
1	1	1	0	1	0	0	1	1	1
1	1	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0	1
1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0	0

IplImage* Source = Ο αρχικός πίνακας που αποτελείται από μηδέν και ένα.

IplImage* Destination = Ο πίνακας που περιέχει αριθμημένες τις απομονωμένες περιοχές.

2	2	0	3	3	3	0	0	0	4
2	2	0	0	3	0	0	0	0	4
0	0	0	0	3	0	0	0	0	4
0	0	0	0	3	0	0	4	4	4
5	5	5	0	3	0	0	4	4	4
5	5	5	0	0	0	0	0	0	4
5	5	5	0	0	0	0	0	0	4
5	5	5	5	5	5	0	0	0	4
5	5	5	5	5	5	0	0	0	0
5	5	5	5	5	5	0	0	0	0

21) cvMakeMyCloseFriendLikeMe

void cvMakeMyCloseFriendLikeMe(IplImage* Destination,int x,int y,int Labeler)

Η συνάρτηση cvMakeMyCloseFriendLikeMe παίρνει έναν πίνακα με μηδέν και ένα και τις συντεταγμένες ενός σημείου του πίνακα που είναι ένα και κάνει όλα τα γειτονικά σημεία που έχουν ένα με την τιμή του Labeled που δίνεται στην συνάρτηση. Η συνάρτηση χρησιμοποιεί αναδρομή καλώντας τον εαυτό της για τα 8 γειτονικά σημεία ενός pixel. Τα γειτονικά σημεία είναι :

	(X -1, Y-1)	(X -1, Y)	(X-1 , Y+1)	
	(X , Y-1)	(X , Y)	(X , Y+1)	
	(X +1, Y-1)	(X +1, Y)	(X +1, Y+1)	

IplImage* Destination = εικόνα που περιέχει μηδέν και ένα της οποία τα στοιχεία προσπαθούμε να χρωματίσουμε

int x = σημείο του οποίου τους γείτονες θέλουμε να αλλάξουμε τιμή.

int y= σημείο του οποίου τους γείτονες θέλουμε να αλλάξουμε τιμή.

int Labeled= η τιμή που θα πάρει το σημείο και οι γείτονες του.

22) MakeRoiPOly

void MakeRoiPOly(IplImage* mask,int a[11][2])

Η συνάρτηση MakeRoiPOly εκτελεί μια πολύ σημαντική λειτουργία. Δεδομένης μιας εικόνας και ενός πίνακα ο οποίος περιέχει 10 συντεταγμένες σημείων δημιουργεί ένα δεκάγωνο πάνω στην εικόνα με βάση τις δέκα συντεταγμένες που έχει πάρει ως είσοδο. Τις τιμές αυτές του τετραγώνου τις κρατάει δημιουργώντας μια μάσκα η οποία έχει ένα μόνο στα σημεία του δεκάγωνου και μηδέν οπουδήποτε αλλού. Με την συνάρτηση αυτή ο χρήστης καλείται να δώσει την περιοχή μέσα στην οποία υπάρχει δέρμα και έτσι ο εντοπισμός δέρματος που θα κάνουμε θα είναι μεγάλης ακρίβειας. Το δεκάγωνο είναι ένας ικανοποιητικός χώρος ώστε να επιλεγεί μια κατάδηλη δερματική περιοχή σε μια εικόνα. Μεγαλύτερο πλήθος γωνιών είναι εφικτό αλλά δεν θα επιφέρει σημαντικές αλλαγές. Οι συντεταγμένες παίρνονται από την onmouse συνάρτηση η οποία θα αναλυθεί παρακάτω.

IplImage* mask = Μάσκα η οποία είναι ένα εκεί που έχει επιλέξει ο χρήστης ότι υπάρχει δέρμα

int a[11][2] = γωνίες δεκάγωνου.

23) MakeEveryKToG

void MakeEveryKToG(IplImage* Source,IplImage* Destination,int k,int g)

Η συνάρτηση MakeEveryKToG παίρνει σαν είσοδο έναν πίνακα και επιστρέφει έναν άλλο στον οποίον κάθε τιμή του αρχικού που είναι ίση με k στον νέο πίνακα έχει γίνει g. Η χρησιμότητα της συνάρτησης έγκειται στο γεγονός ότι το OpenCV δεν μπορεί να απεικονίσει λογικούς πίνακες. Έτσι χρησιμοποιώντας την συνάρτηση μετατρέπω τα 1 σε 255 και μπορώ να έχω την επιθυμητή απεικόνιση.Γενικότερα χρησιμοποιείται για διάφορες μετατροπές που πραγματοποιούνται

IplImage* Source = Πίνακας ο οποίος περιέχει στοιχεία που θέλουμε να αλλάξουμε.

IplImage* Destination = Πίνακας που περιέχει τα στοιχεία του Source εκτός από κάποια που έχει αλλάξει η συνάρτηση.

int k = Στοιχείο που θα αλλάξει.

int g = Στοιχείο που θα αντικαταστήσει το στοιχείο που αλλάζει.

24) on_mouse

void on_mouse(int event, int x, int y, int flags, void* param)

Η συνάρτηση on_mouse ουσιαστικά είναι ένας listener η οποία ενεργοποιείται οποτεδήποτε έχουμε μια αλλαγή στην κατάσταση του ποντικιού. Στην συγκεκριμένη περίπτωση ενεργοποιείται όταν πατιέται το αριστερό κουμπί του ποντικιού. Τις δέκα πρώτες φορές που πατιέται το κουμπί η συνάρτηση αποθηκεύει τις συντεταγμένες της εικόνας πάνω στην οποία βρίσκεται το ποντίκι. Με αυτό τον τρόπο ο χρήστης δίνει το δεκάγωνο που προαναφέρθηκε προκειμένου να καθορίσει μια δερματική περιοχή μέσω της οποίας θα κάνουμε τους διάφορους υπολογισμούς. Ο χρήστης δεν πρέπει να πατήσει το πλήκτρο του ποντικιού πριν του ζητηθεί να το κάνει γιατί θα έχουμε λανθασμένα αποτελέσματα. Η συνάρτηση αυτή χρησιμοποιεί τον public πίνακα RoiPoly και ο πίνακας αυτός χρησιμοποιείται μετά από την συνάρτηση που επιλέγει την δερματική μάσκα.

int event = μεταβλητή η οποία καθορίζει την λειτουργία που εκτέλεσε το ποντίκι στη προκειμένη περίπτωση επιλέγουμε την CV_EVENT_LBUTTONDOWN

int x = συντεταγμένη σημείου πάνω στο οποίο βρίσκεται το ποντίκι.

int y x = συντεταγμένη σημείου πάνω στο οποίο βρίσκεται το ποντίκι.

int flags = μεταβλητή που παίρνει τιμές κατά την διάρκεια της εκτέλεσης.

void* param = μεταβλητή που παίρνει τιμές κατά την διάρκεια της εκτέλεσης.

25) cvMeanM

void cvMeanM(IplImage* src, IplImage* dst)

Η συνάρτηση cvMeanM παίρνει έναν δισδιάστατο πίνακα και επιστρέφει έναν μονοδιάστατο πίνακα με τόσα στοιχεία όσες η στήλες του δισδιάστατου και κάθε στοιχείο του είναι το μέσο όρο των στοιχείων της αντίστοιχης στήλης του δισδιάστατου πίνακα.

IplImage* src = δισδιάστατος πίνακας του οποίου θέλουμε να υπολογίσουμε το μέσο όρο των στηλών του.

IplImage* dst= μονοδιάστατος πίνακας που περιέχει το μέσο όρο κάθε στήλης.

Κ Ε Φ Α Λ Α Ι Ο 7

Κώδικας προγράμματος

7.1 Κώδικας Συναρτήσεων

cvNormal01

```
IplImage* cvNormal01(IplImage* x,IplImage* y,double mx,double my,double sx,double sy){  //x=Cr y=Cb  Destination=z
        CvSize kopsia1;
        IplImage* V; //{-0.9981,-0.0611,0.0611,-0.9981};
        kopsia1.height=2;
        kopsia1.width=2;
        V=cvCreateImage(kopsia1,IPL_DEPTH_32F ,1);
        cvSetReal2D(V,0,0,-0.9981);
        cvSetReal2D(V,0,1,-0.0611);
        cvSetReal2D(V,1,0,0.0611);
        cvSetReal2D(V,1,1,-0.9981);

        int I,J;
        I=x->height;
        J=x->width;

        IplImage* Z;
        IplImage* Zana;
```

```
IplImage* Cb;
IplImage* Cr;
Cr=cvMakeImageCol(x);
Cb=cvMakeImageCol(y);
Z=cvAddColumToImage(Cr,Cb);

kopsia1.height= Z->width;
kopsia1.width=Z->height;
Zana=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvTranspose(Z,Zana);
Z=Zana;

IplImage* M1NF;
M1NF=cvOnes(I*J,1);
IplImage* M1;
kopsia1.height= M1NF->height;
kopsia1.width=M1NF->width;
M1=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvConvertScale(M1NF,M1,mx);

IplImage* M2NF;
M2NF=cvOnes(I*J,1);
IplImage* M2;
kopsia1.height= M2NF->height;
kopsia1.width=M2NF->width;
M2=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvConvertScale(M2NF,M2,my);

IplImage* L1NF;
L1NF=cvOnes(I*J,1);
IplImage* L1;
kopsia1.height= L1NF->height;
kopsia1.width=L1NF->width;
L1=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvConvertScale(L1NF,L1,sx);

IplImage* L2NF;
L2NF=cvOnes(I*J,1);
IplImage* L2;
kopsia1.height= L2NF->height;
kopsia1.width=L2NF->width;
L2=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvConvertScale(L2NF,L2,sy);
```

```
IplImage* M;
M=cvAddColumToImage(M1,M2);
kopsia1.height= M->width;
kopsia1.width=M->height;
Zana=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvTranspose(M,Zana);
//cvReleaseImage(&M);
M=Zana;


IplImage* L;
L=cvAddColumToImage(L1,L2);
kopsia1.height= L->width;
kopsia1.width=L->height;
Zana=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvTranspose(L,Zana);
//cvReleaseImage(&L);
L=Zana;


IplImage* TImg;
kopsia1.height= Z->height;
kopsia1.width=Z->width;
TImg=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvSub(Z,M,TImg);//TImg=Z-M
kopsia1.height= V->width;
kopsia1.width=V->height;
Zana=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvTranspose(V,Zana);
//TImg=Z-M kai Zana=V' theloume tora Z2=Zana*TImg
IplImage* Z2;
kopsia1.height= Zana->height;
kopsia1.width=TImg->width;
Z2=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
//pollaplasiasmos pinakon
cvGEMM(Zana,TImg,1,NULL,0,Z2);

IplImage* Z2Tetragonismeno;
kopsia1.height= Z2->height;
kopsia1.width=Z2->width;
Z2Tetragonismeno=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvMul(Z2,Z2,Z2Tetragonismeno);

IplImage* D;
kopsia1.height= Z2Tetragonismeno->height;
kopsia1.width=Z2Tetragonismeno->width;
```

```
        D=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

        cvDiv(Z2Tetragonismeno,L,D);

        cvConvertScale(D,D,-0.5);

        kopsia1.height= D->height;
        kopsia1.width=D->width;
        TImg=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
        cvExp(D,TImg);

        IplImage* Znormal01;
        Znormal01=cvMakeRowImage(TImg,I,J);

        cvReleaseImage(&Z);
        cvReleaseImage(&Z2);
        cvReleaseImage(&Zana);
        cvReleaseImage(&TImg);
        cvReleaseImage(&D);
        cvReleaseImage(&L2NF);
        cvReleaseImage(&L1NF);
        cvReleaseImage(&M2NF);
        cvReleaseImage(&M1NF);
        cvReleaseImage(&L2);
        cvReleaseImage(&L1);
        cvReleaseImage(&M2);
        cvReleaseImage(&M1);
        cvReleaseImage(&M);
        cvReleaseImage(&L);
        cvReleaseImage(&Z2Tetragonismeno);
        cvReleaseImage(&Cb);
        cvReleaseImage(&Cr);
        cvReleaseImage(&V);
        return Znormal01;
}//telos cvNormal01
```

cvRGB2YCbCrChanelY

```
IplImage* cvRGB2YCbCrChanelY(IplImage* X){
        CvSize kopsia1;
        IplImage* TAB;
        kopsia1.height=3;
        kopsia1.width=3;
        TAB=cvCreateImage(kopsia1,IPL_DEPTH_32F ,1);
        cvSetReal2D(TAB,0,0,0.299);
        cvSetReal2D(TAB,0,1,0.587);
        cvSetReal2D(TAB,0,2,0.114);
```

```
cvSetReal2D(TAB,1,0,-0.169);
cvSetReal2D(TAB,1,1,-0.331);
cvSetReal2D(TAB,1,2,0.5);
cvSetReal2D(TAB,2,0,0.5);
cvSetReal2D(TAB,2,1,-0.419);
cvSetReal2D(TAB,2,2,-0.081);

IplImage* X1;
IplImage* X2;
IplImage* X3;
IplImage* R;
IplImage* G;
IplImage* B;
kopsia1.height=X->height;
kopsia1.width=X->width;
X1=cvCreateImage(kopsia1,X->depth,1);
X2=cvCreateImage(kopsia1,X->depth,1);
X3=cvCreateImage(kopsia1,X->depth,1);
R=cvCreateImage(kopsia1,TAB->depth,1);//h    IPL_DEPTH_32F    na    to
dokimaso
G=cvCreateImage(kopsia1,TAB->depth,1);
B=cvCreateImage(kopsia1,TAB->depth,1);

cvSplit(X,X1,X2,X3,NULL);

cvConvertScale(X1,R);
cvConvertScale(X2,G);
cvConvertScale(X3,B);

IplImage* Rc;
IplImage* Gc;
IplImage* Bc;

Rc=cvMakeImageCol(R);
Gc=cvMakeImageCol(G);
Bc=cvMakeImageCol(B);

IplImage* RGBc;
RGBc=cvAddColumToImage(Rc,Gc);
RGBc=cvAddColumToImage(RGBc,Bc);

IplImage* RGBcT;
kopsia1.height=RGBc->width;
kopsia1.width=RGBc->height;
RGBcT=cvCreateImage(kopsia1,RGBc->depth,1);
cvTranspose(RGBc,RGBcT);  //RGBcT=[r(:) g(:) b(:)]'
```

```
IplImage* TAB1;//TAB1=tab(1,:)
IplImage* TAB2;//TAB1=tab(2,:)
IplImage* TAB3;//TAB1=tab(3,:)
TAB1=cvTakeRowFromImage(TAB,0);
TAB2=cvTakeRowFromImage(TAB,1);
TAB3=cvTakeRowFromImage(TAB,2);

IplImage* Yy;
kopsia1.height=TAB1->height;
kopsia1.width=RGBcT->width;
Yy=cvCreateImage(kopsia1,TAB->depth,1);
cvGEMM(TAB1,RGBcT,1,NULL,0,Yy);

double Ymin,Ymax;
CvPoint YminCor,YmaxCor;
cvMinMaxLoc(Yy,&Ymin,&Ymax,&YminCor,&YmaxCor);

IplImage* YySub;
kopsia1.height=Yy->height;
kopsia1.width=Yy->width;
YySub=cvCreateImage(kopsia1,Yy->depth,1);
CvScalar meiosh;
meiosh.val[0]=Ymin;
cvSubS(Yy,meiosh,YySub);//y=(y-min(min(y)))

IplImage* YySubDiv;
kopsia1.height=Yy->height;
kopsia1.width=Yy->width;
YySubDiv=cvCreateImage(kopsia1,Yy->depth,1);
if(Ymax-Ymin>0)       cvConvertScale(YySub,YySubDiv,1.0/(Ymax-Ymin));
//YySubDiv=(y-min(min(y)))/(max(max(y))-min(min(y)))

cvReleaseImage(&TAB);
cvReleaseImage(&X1);
cvReleaseImage(&X2);
cvReleaseImage(&X3);
cvReleaseImage(&R);
cvReleaseImage(&G);
cvReleaseImage(&B);
cvReleaseImage(&TAB1);
cvReleaseImage(&TAB2);
cvReleaseImage(&TAB3);
cvReleaseImage(&Rc);
cvReleaseImage(&Gc);
cvReleaseImage(&Bc);
```

```
        cvReleaseImage(&RGBc);
        cvReleaseImage(&RGBcT);
        cvReleaseImage(&Yy);
        cvReleaseImage(&YySub);
        IplImage* Y;
        Y=cvMakeRowImage(YySubDiv,X->height,X->width);
        cvReleaseImage(&YySubDiv);
        return Y;
}//telos cvRGB2YCbCrChanelY
```

cvRGB2YCbCrChanelCr

```
IplImage* cvRGB2YCbCrChanelCr(IplImage* X){
        CvSize kopsia1;
        IplImage* TAB;
        kopsia1.height=3;
        kopsia1.width=3;
        TAB=cvCreateImage(kopsia1,IPL_DEPTH_32F ,1);
        cvSetReal2D(TAB,0,0,0.299);
        cvSetReal2D(TAB,0,1,0.587);
        cvSetReal2D(TAB,0,2,0.114);
        cvSetReal2D(TAB,1,0,-0.169);
        cvSetReal2D(TAB,1,1,-0.331);
        cvSetReal2D(TAB,1,2,0.5);
        cvSetReal2D(TAB,2,0,0.5);
        cvSetReal2D(TAB,2,1,-0.419);
        cvSetReal2D(TAB,2,2,-0.081);
        IplImage* X1;
        IplImage* X2;
        IplImage* X3;
        IplImage* R;
        IplImage* G;
        IplImage* B;

        kopsia1.height=X->height;
        kopsia1.width=X->width;
        X1=cvCreateImage(kopsia1,X->depth,1);
        X2=cvCreateImage(kopsia1,X->depth,1);
        X3=cvCreateImage(kopsia1,X->depth,1);
        R=cvCreateImage(kopsia1,TAB->depth,1);//h    IPL_DEPTH_32F    na    to
        dokimaso
        G=cvCreateImage(kopsia1,TAB->depth,1);
        B=cvCreateImage(kopsia1,TAB->depth,1);

        cvSplit(X,X1,X2,X3,NULL);
```

```
cvConvertScale(X1,R);
cvConvertScale(X2,G);
cvConvertScale(X3,B);

IplImage* Rc;
IplImage* Gc;
IplImage* Bc;

Rc=cvMakeImageCol(R);
Gc=cvMakeImageCol(G);
Bc=cvMakeImageCol(B);

IplImage* RGBc;
RGBc=cvAddColumToImage(Rc,Gc);
RGBc=cvAddColumToImage(RGBc,Bc);

IplImage* RGBcT;
kopsia1.height=RGBc->width;
kopsia1.width=RGBc->height;
RGBcT=cvCreateImage(kopsia1,RGBc->depth,1);
cvTranspose(RGBc,RGBcT);  //RGBcT=[r(:) g(:) b(:)]'

IplImage* TAB3;//TAB1=tab(3,:)
TAB3=cvTakeRowFromImage(TAB,1);

IplImage* CR;
kopsia1.height=TAB3->height;
kopsia1.width=RGBcT->width;
CR=cvCreateImage(kopsia1,TAB->depth,1);
cvGEMM(TAB3,RGBcT,1,NULL,0,CR);

IplImage* Cr;
Cr=cvMakeRowImage(CR,X->height,X->width);

cvReleaseImage(&TAB);
cvReleaseImage(&X1);
cvReleaseImage(&X2);
cvReleaseImage(&X3);
cvReleaseImage(&R);
cvReleaseImage(&G);
cvReleaseImage(&B);
cvReleaseImage(&TAB3);
cvReleaseImage(&Rc);
cvReleaseImage(&Gc);
cvReleaseImage(&Bc);
cvReleaseImage(&RGBc);
```

```
	cvReleaseImage(&RGBcT);
	cvReleaseImage(&CR);

	return Cr;
}//telos  cvRGB2YCbCrChanelCr
```

cvRGB2YCbCrChanelCb

```
IplImage* cvRGB2YCbCrChanelCb(IplImage* X){
	CvSize kopsia1;
	IplImage* TAB;
	kopsia1.height=3;
	kopsia1.width=3;
	TAB=cvCreateImage(kopsia1,IPL_DEPTH_32F ,1);
	cvSetReal2D(TAB,0,0,0.299);
	cvSetReal2D(TAB,0,1,0.587);
	cvSetReal2D(TAB,0,2,0.114);
	cvSetReal2D(TAB,1,0,-0.169);
	cvSetReal2D(TAB,1,1,-0.331);
	cvSetReal2D(TAB,1,2,0.5);
	cvSetReal2D(TAB,2,0,0.5);
	cvSetReal2D(TAB,2,1,-0.419);
	cvSetReal2D(TAB,2,2,-0.081);
	IplImage* X1;
	IplImage* X2;
	IplImage* X3;
	IplImage* R;
	IplImage* G;
	IplImage* B;

	kopsia1.height=X->height;
	kopsia1.width=X->width;
	X1=cvCreateImage(kopsia1,X->depth,1);
	X2=cvCreateImage(kopsia1,X->depth,1);
	X3=cvCreateImage(kopsia1,X->depth,1);
	R=cvCreateImage(kopsia1,TAB->depth,1);//h    IPL_DEPTH_32F    na    to
	dokimaso
	G=cvCreateImage(kopsia1,TAB->depth,1);
	B=cvCreateImage(kopsia1,TAB->depth,1);

	cvSplit(X,X1,X2,X3,NULL);

	cvConvertScale(X1,R);
	cvConvertScale(X2,G);
	cvConvertScale(X3,B);
```

```
IplImage* Rc;
IplImage* Gc;
IplImage* Bc;

Rc=cvMakeImageCol(R);
Gc=cvMakeImageCol(G);
Bc=cvMakeImageCol(B);

IplImage* RGBc;
RGBc=cvAddColumToImage(Rc,Gc);
RGBc=cvAddColumToImage(RGBc,Bc);

IplImage* RGBcT;
kopsia1.height=RGBc->width;
kopsia1.width=RGBc->height;
RGBcT=cvCreateImage(kopsia1,RGBc->depth,1);
cvTranspose(RGBc,RGBcT);  //RGBcT=[r(:) g(:) b(:)]'

IplImage* TAB2;//TAB1=tab(2,:)
TAB2=cvTakeRowFromImage(TAB,2);

IplImage* CB;
kopsia1.height=TAB2->height;
kopsia1.width=RGBcT->width;
CB=cvCreateImage(kopsia1,TAB->depth,1);
cvGEMM(TAB2,RGBcT,1,NULL,0,CB);

IplImage* Cb;
Cb=cvMakeRowImage(CB,X->height,X->width);


cvReleaseImage(&TAB);
cvReleaseImage(&X1);
cvReleaseImage(&X2);
cvReleaseImage(&X3);
cvReleaseImage(&R);
cvReleaseImage(&G);
cvReleaseImage(&B);
cvReleaseImage(&TAB2);
cvReleaseImage(&Rc);
cvReleaseImage(&Gc);
cvReleaseImage(&Bc);
cvReleaseImage(&RGBc);
cvReleaseImage(&RGBcT);
cvReleaseImage(&CB);
```

```
	return Cb;
}//telos  cvRGB2YCbCbChanelCb
```

cvGetSkinProps

```
IplImage* cvGetSkinProps(IplImage* X,double* zhue,double* zsat,double* zval,double* mx,double* my,double* mhue,double* msat,double* mval){
	CvSize kopsia1;
	IplImage* Y;
	IplImage* Cb;
	IplImage* Cr;
	Y=cvRGB2YCbCrChanelY(X);
	Cb=cvRGB2YCbCrChanelCb(X);
	Cr=cvRGB2YCbCrChanelCr(X);

	IplImage* CbF;
	IplImage* CrF;
	kopsia1.height=Cb->height;
	kopsia1.width=Cb->width;
	CbF=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
	CrF=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

	cvConvertScale(Cb,CbF,1.0/255);
	cvConvertScale(Cr,CrF,1.0/255);

	//roipoly
	ci=0;
	cvNamedWindow( "Polygon Selection", 1 );
	cvShowImage( "Polygon Selection", X);
	printf("Give a 10 cornered polygon by cliking 10 times on the photo indicating the polygon corners!!!\n");
	cvSetMouseCallback( "Polygon Selection", on_mouse, 0 );
	cvWaitKey(); //Press a key to continue in order to give time to user to give cordinates

	IplImage* RoipolyMask;
	kopsia1.height=X->height;
	kopsia1.width=X->width;
	RoipolyMask=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

	MakeRoiPOly(RoipolyMask,RoiPoly);

	IplImage* RoipolyMaskOnes;
	kopsia1.height=RoipolyMask->height;
	kopsia1.width=RoipolyMask->width;
	RoipolyMaskOnes=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
```

```
MakeImage255To1(RoipolyMask,RoipolyMaskOnes);

IplImage* RoipolyMaskOnesF;
kopsia1.height=RoipolyMaskOnes->height;
kopsia1.width=RoipolyMaskOnes->width;
RoipolyMaskOnesF=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvConvertScale(RoipolyMaskOnes,RoipolyMaskOnesF);

int nonzerocounter=0;
for(int i1=0;i1<RoipolyMaskOnes->height;i1++)
                for(int j1=0;j1<RoipolyMaskOnes->width;j1++){
                        if(cvGetReal2D(RoipolyMaskOnes,i1,j1)>0){
                                nonzerocounter++;
                        }
                }


IplImage* CrMAsk;
kopsia1.height=CrF->height;
kopsia1.width=CrF->width;
CrMAsk=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvMul(CrF,RoipolyMaskOnesF,CrMAsk);

IplImage* CrMAskForMean;
kopsia1.height=nonzerocounter;
kopsia1.width=1;
CrMAskForMean=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);


IplImage* CbMAsk;
kopsia1.height=CbF->height;
kopsia1.width=CbF->width;
CbMAsk=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvMul(CbF,RoipolyMaskOnesF,CbMAsk);

IplImage* CbMAskForMean;
kopsia1.height=nonzerocounter;
kopsia1.width=1;
CbMAskForMean=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

nonzerocounter=0;
for(int i11=0;i11<RoipolyMaskOnes->height;i11++)
                for(int j11=0;j11<RoipolyMaskOnes->width;j11++){
                        if(cvGetReal2D(RoipolyMaskOnes,i11,j11)>0){
```

```
        cvSetReal2D(CrMAskForMean,nonzerocounter,0,cvGetReal2D(CrMAsk,i11,
        j11));
        cvSetReal2D(CbMAskForMean,nonzerocounter,0,cvGetReal2D(CbMAsk,i11
        ,j11));
                                        nonzerocounter++;
                                }
                        }


        CvScalar mean2x;
        mean2x=cvAvg(CrMAskForMean);
        *mx=mean2x.val[0];

        CvScalar mean2y;
        mean2y=cvAvg(CbMAskForMean);
        *my=mean2y.val[0];
        double sx;
        double sy;
        sx=0.005;
        sy=0.005;

        IplImage* Znor;
        Znor=cvNormal01(CrF,CbF,*mx,*my,sx,sy);

        //arxh Release
        cvReleaseImage(&CbMAskForMean);
        cvReleaseImage(&CbMAsk);
        cvReleaseImage(&CrMAskForMean);
        cvReleaseImage(&CrMAsk);
        cvReleaseImage(&RoipolyMaskOnesF);
        cvReleaseImage(&RoipolyMaskOnes);
        cvReleaseImage(&RoipolyMask);
        cvReleaseImage(&CrF);
        cvReleaseImage(&CbF);
        cvReleaseImage(&Cr);
        cvReleaseImage(&Cb);
        cvReleaseImage(&Y);
        //telos Release

        return Znor;
}//telos cvGetSkinProps
```

Cvobjxy

```
void cvobjxy(IplImage* mask,int* xmean,int* ymean){ //thn maska na thn exo ftiaksei na einai me asous kai mhden prin kaleso thn sunartish
	IplImage* X;
	IplImage* Y;
	CvSize kopsia1;
	IplImage* Mask8U;

	kopsia1.height=mask->height;
	kopsia1.width=mask->width;
	Mask8U=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
	cvConvertScale(mask,Mask8U);

	int nonzerocounter=0;
	for(int i1=0;i1<Mask8U->height;i1++)
			for(int j1=0;j1<Mask8U->width;j1++){
				if(cvGetReal2D(Mask8U,i1,j1)>0){
					nonzerocounter++;
				}
			}

	kopsia1.height=nonzerocounter;
	kopsia1.width=1;
	X=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
	Y=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

	nonzerocounter=0;
	for(int i11=0;i11<Mask8U->height;i11++)
			for(int j11=0;j11<Mask8U->width;j11++){
				if(cvGetReal2D(Mask8U,i11,j11)==1){
					cvSetReal2D(X,nonzerocounter,0,i11);
					cvSetReal2D(Y,nonzerocounter,0,j11);
					nonzerocounter++;
				}
			}

	CvScalar meanX;
	meanX=cvAvg(X);
	*xmean=cvRound(meanX.val[0]);

	CvScalar meanY;
	meanY=cvAvg(Y);
	*ymean=cvRound(meanY.val[0]);
	//arxh Release
	cvReleaseImage(&X);
	cvReleaseImage(&Y);
	cvReleaseImage(&Mask8U);
```

```
	//telos Release
}// telos tou cvobjxy
```

cvFindMovingSkin

```
void cvFindMovingSkin(IplImage* ImgA,IplImage* ImgB,double mx,double my,IplImage* ms,IplImage* cmout,IplImage* mmout){
	IplImage* chY;
	IplImage* chCr;
	IplImage* chCb;

	chY=cvRGB2YCbCrChanelY(ImgA);
	chCr=cvRGB2YCbCrChanelCr(ImgA);
	chCb=cvRGB2YCbCrChanelCb(ImgA);

	IplImage* CrDB;
	IplImage* CbDB;

	CvSize kopsia1;
	kopsia1.height=chY->height;
	kopsia1.width=chY->width;
	CrDB=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
	CbDB=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

	cvConvertScale(chCr,CrDB,1.0/255);
	cvConvertScale(chCb,CbDB,1.0/255);

	int r,c;
	r=chY->height;
	c=chY->width;

	IplImage* Z3;
	Z3=cvNormal01(CrDB,CbDB,mx,my,0.001,0.001);

	IplImage* cm1;
	kopsia1.height=ImgA->height;
	kopsia1.width=ImgA->width;
	cm1=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

	int i2,j2;
	for(i2=0;i2<ImgA->height;i2++)
			for(j2=0;j2<ImgA->width;j2++){
				if(cvGetReal2D(Z3,i2,j2)>0.8){

					cvSetReal2D(cm1,i2,j2,1);
```

```
                        }
                        else
                        {
                                cvSetReal2D(cm1,i2,j2,0);
                        }
                }//telos j2

IplImage* ImgAGrayscale;
IplImage* ImgBGrayscale;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
ImgAGrayscale=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
ImgBGrayscale=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

cvCvtColor(ImgA,ImgAGrayscale,CV_RGB2GRAY);
cvCvtColor(ImgB,ImgBGrayscale,CV_RGB2GRAY);

IplImage* ImgAGrayscaleDB;
IplImage* ImgBGrayscaleDB;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
ImgAGrayscaleDB=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
ImgBGrayscaleDB=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

cvConvertScale(ImgAGrayscale,ImgAGrayscaleDB,1.0/255);
cvConvertScale(ImgBGrayscale,ImgBGrayscaleDB,1.0/255);

IplImage* Dif;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
Dif=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

cvAbsDiff(ImgAGrayscaleDB,ImgBGrayscaleDB,Dif);

IplImage* mm1;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
mm1=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

for(i2=0;i2<ImgA->height;i2++)
                for(j2=0;j2<ImgA->width;j2++){
                        if(cvGetReal2D(Dif,i2,j2)>0.11){

                                cvSetReal2D(mm1,i2,j2,1);
                        }
                        else
```

```
                {
                        cvSetReal2D(mm1,i2,j2,0);
                }
        }//telos j2

int disksize;
disksize=cvRound(c/150);

IplImage* mm2;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
mm2=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

IplConvKernel* maskastreldisk;
maskastreldisk=cvCreateStructuringElementEx(disksize,disk
size,NULL,NULL,CV_SHAPE_ELLIPSE,NULL);
cvMorphologyEx(mm1,mm2,NULL,maskastreldisk,CV_MO P_CLOSE);

IplImage* mm3;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
mm3=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

cvMorphologyEx(mm2,mm3,NULL,maskastreldisk,CV_MO   P_OPEN);


IplImage* cm2;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
cm2=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
cvMorphologyEx(cm1,cm2,NULL,maskastreldisk,CV_MOP_CLOSE);

IplImage* dt;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
dt=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvDistTransform(cm2,dt);

int dtsize;
dtsize=cvRound(c/100);


IplImage* dt_markers;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
dt_markers=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
```

```
for(i2=0;i2<ImgA->height;i2++)
		for(j2=0;j2<ImgA->width;j2++){
			if(cvGetReal2D(dt,i2,j2)>dtsize){

				cvSetReal2D(dt_markers,i2,j2,1);
			}
			else
			{
				cvSetReal2D(dt_markers,i2,j2,0);
			}
		}//telos j2

IplImage* cm3;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
cm3=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

IplImage* dtPOS;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
dtPOS=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

	for(i2=0;i2<ImgA->height;i2++)
		for(j2=0;j2<ImgA->width;j2++){
			if(cvGetReal2D(dt,i2,j2)>0){

				cvSetReal2D(dtPOS,i2,j2,1);
			}
			else
			{
				cvSetReal2D(dtPOS,i2,j2,0);
			}
		}//telos j2

IplImage* dt_markersPOS;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
dt_markersPOS=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

for(i2=0;i2<ImgA->height;i2++)
		for(j2=0;j2<ImgA->width;j2++){
			if(cvGetReal2D(dt_markers,i2,j2)>0){

				cvSetReal2D(dt_markersPOS,i2,j2,1);
			}
```

```
                        else
                        {
                                cvSetReal2D(dt_markersPOS,i2,j2,0);
                        }
                }//telos j2

cvImReconstruct(dt_markersPOS,dtPOS,cm3);


IplImage* movingSkinMarkers;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
movingSkinMarkers=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

IplImage* cm3POS;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
cm3POS=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

for(i2=0;i2<ImgA->height;i2++)
                for(j2=0;j2<ImgA->width;j2++){
                        if(cvGetReal2D(cm3,i2,j2)>0){

                                cvSetReal2D(cm3POS,i2,j2,1);
                        }
                        else
                        {
                                cvSetReal2D(cm3POS,i2,j2,0);
                        }
                }//telos j2

cvMul(cm3POS,mm3,movingSkinMarkers);

IplImage* msTemp;
kopsia1.height=ImgA->height;
kopsia1.width=ImgA->width;
msTemp=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

cvImReconstruct(movingSkinMarkers,cm3POS,msTemp);

for(i2=0;i2<ImgA->height;i2++)
                for(j2=0;j2<ImgA->width;j2++){

cvSetReal2D(ms,i2,j2,cvGetReal2D(msTemp,i2,j2));
                }//telos j2
for(i2=0;i2<ImgA->height;i2++)
```

```
                        for(j2=0;j2<ImgA->width;j2++){

        cvSetReal2D(mmout,i2,j2,cvGetReal2D(mm3,i2,j2));
                        }//telos j2

        for(i2=0;i2<ImgA->height;i2++)
                        for(j2=0;j2<ImgA->width;j2++){

        cvSetReal2D(cmout,i2,j2,cvGetReal2D(cm3,i2,j2));
                        }//telos j2
        //edo bgazo otidhpote axrhsto apo mnhmh
        cvReleaseImage(&msTemp);
        cvReleaseImage(&cm3POS);
        cvReleaseImage(&movingSkinMarkers);
        cvReleaseImage(&dt_markersPOS);
        cvReleaseImage(&cm3);
        cvReleaseImage(&dt_markers);
        cvReleaseImage(&dt);
        cvReleaseImage(&cm2);
        cvReleaseImage(&mm3);
        cvReleaseImage(&mm2);
        cvReleaseImage(&mm1);
        cvReleaseImage(&Dif);
        cvReleaseImage(&ImgBGrayscaleDB);
        cvReleaseImage(&ImgAGrayscaleDB);
        cvReleaseImage(&ImgAGrayscale);
        cvReleaseImage(&ImgBGrayscale);
        cvReleaseImage(&cm1);
        cvReleaseImage(&Z3);
        cvReleaseImage(&CrDB);
        cvReleaseImage(&CbDB);
        cvReleaseImage(&chY);
        cvReleaseImage(&chCr);
        cvReleaseImage(&chCb);
        //Telos Release
}//telos cvFindMovingSkin
```

cvProcSkinDetection

```
IplImage*  cvProcSkinDetection(IplImage*  ms,IplImage*  cmout,IplImage*
mmout,IplImage* prevcoords){
        CvSize kopsia1;
```

```
int i2,j2,iL;

IplImage* curcoords;
kopsia1.height=prevcoords->height;
kopsia1.width=prevcoords->width;
curcoords=cvCreateImage(kopsia1,prevcoords->depth,1);

IplImage* msPOS;
kopsia1.height=ms->height;
kopsia1.width=ms->width;
msPOS=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

bool flag1=false;
	for(i2=0;i2<ms->height;i2++)
		for(j2=0;j2<ms->width;j2++){
			if(cvGetReal2D(ms,i2,j2)>0){

				cvSetReal2D(msPOS,i2,j2,1);
				flag1=true;
			}
			else
			{
				cvSetReal2D(msPOS,i2,j2,0);
			}
		}//telos j2

if(flag1==false){
	for(i2=0;i2<ms->height;i2++)
		for(j2=0;j2<ms->width;j2++){

cvSetReal2D(ms,i2,j2,cvGetReal2D(cmout,i2,j2));
		}//telos j2
}


int ro,co;
ro=ms->height;
co=ms->width;

double unitsize;
unitsize=sqrt(ro*ro+co*co)/10;
//dokimh
//if(unitsize>90) printf("Malakas");
//printf("unitsize: %fl \n",unitsize);
//telos dokimhs
int h,r,l;
```

```
h=-1;
r=-1;
l=-1;

IplImage* bw;
kopsia1.height=ms->height;
kopsia1.width=ms->width;
bw=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
cvBwLabel(ms,bw);

int nor;
double maxtemp;
cvMinMaxLoc(bw,NULL,&maxtemp);
nor=cvRound(maxtemp);

if(nor==0){
	for(i2=0;i2<prevcoords->height;i2++)
		for(j2=0;j2<prevcoords->width;j2++){
cvSetReal2D(curcoords,i2,j2,cvGetReal2D(prevcoords,i2,j2));
		}//telos j2
		return curcoords;
}//telos if nor==0

IplImage* bweqI;
kopsia1.height=ms->height;
kopsia1.width=ms->width;
bweqI=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

IplImage* regpos;
kopsia1.height=nor+1;
kopsia1.width=2;
regpos=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);

for(iL=2;iL<nor+1;iL++){
	int X,Y;

	for(i2=0;i2<bw->height;i2++)
		for(j2=0;j2<bw->width;j2++){
			if(cvGetReal2D(bw,i2,j2)==iL){

				cvSetReal2D(bweqI,i2,j2,1);
			}
			else
			{
				cvSetReal2D(bweqI,i2,j2,0);
			}
```

```
                }//telos j2
        cvobjxy(bweqI,&X,&Y);
        cvSetReal2D(regpos,iL,0,X);
        cvSetReal2D(regpos,iL,1,Y);
                if(             cvGetReal2D(regpos,iL,0)<(ro/2)             &&
cvGetReal2D(regpos,iL,0)>100  &&  cvGetReal2D(regpos,iL,1)>2*co/5  &&
cvGetReal2D(regpos,iL,1)<3*(co/5)  ){
                h=iL;
        }
}//telos iL


//cvSave("regpos.xml",regpos); sosth
if(h!=-1){
        for(iL=2;iL<nor+1;iL++){
if(cvGetReal2D(regpos,iL,1)<cvGetReal2D(regpos,h,1)                     &&
cvGetReal2D(regpos,iL,0)>100 ) {
                        r=iL;
                }
                        if(cvGetReal2D(regpos,iL,1)>cvGetReal2D(regpos,h,1)
&& cvGetReal2D(regpos,iL,0)>100 ) {
                        l=iL;
                }
        }

}
else
{
        if(cvGetReal2D(prevcoords,4,0)!=-1){
                for(iL=2;iL<nor+1;iL++){
if(cvGetReal2D(regpos,iL,1)<cvGetReal2D(prevcoords,4,0)                 &&
cvGetReal2D(regpos,iL,0)>100 ) {
                                r=iL;
                        }

        if(cvGetReal2D(regpos,iL,1)>cvGetReal2D(prevcoords,4,0)         &&
cvGetReal2D(regpos,iL,0)>100 ) {
                                l=iL;
                        }
                }//telos iL
        }//telos if !=-1

}//telos if h!=-1

//tora bainoun oi elegxoi gia to h l r
```

```
	if( (r!=-1) && (cvGetReal2D(prevcoords,1,0)!=-1) &&
(cvGetReal2D(prevcoords,2,0)!=-1)){ //o ypoloipos elegxos mpainei apo kato
giati thelei r!=-1
	if(abs(cvGetReal2D(regpos,r,0)-
cvGetReal2D(prevcoords,1,0))>unitsize || abs(cvGetReal2D(regpos,r,1)-
cvGetReal2D(prevcoords,2,0))>unitsize ){
		r=-1;
	}//telos regpros
}//telos (r!=-1)

if( (h!=-1) && (cvGetReal2D(prevcoords,3,0)!=-1) &&
(cvGetReal2D(prevcoords,4,0)!=-1)){
	if(abs(cvGetReal2D(regpos,h,0)-
cvGetReal2D(prevcoords,3,0))>unitsize || abs(cvGetReal2D(regpos,h,1)-
cvGetReal2D(prevcoords,4,0))>unitsize ){
		h=-1;
	}//telos regpros
}//telos (h!=-1)

if( (l!=-1) && (cvGetReal2D(prevcoords,5,0)!=-1) &&
(cvGetReal2D(prevcoords,6,0)!=-1)){
	if(abs(cvGetReal2D(regpos,l,0)-
cvGetReal2D(prevcoords,5,0))>unitsize || abs(cvGetReal2D(regpos,l,1)-
cvGetReal2D(prevcoords,6,0))>unitsize ){
		l=-1;
	}//telos regpros
}//telos (l!=-1)


if(r==-1){
	cvSetReal2D(curcoords,1,0,cvGetReal2D(prevcoords,1,0));
	cvSetReal2D(curcoords,2,0,cvGetReal2D(prevcoords,2,0));
}
else
{
	cvSetReal2D(curcoords,1,0,cvGetReal2D(regpos,r,0));
	cvSetReal2D(curcoords,2,0,cvGetReal2D(regpos,r,1));
}//telos r==-1

if(h==-1){
	cvSetReal2D(curcoords,3,0,cvGetReal2D(prevcoords,3,0));
	cvSetReal2D(curcoords,4,0,cvGetReal2D(prevcoords,4,0));
}
else
{
	cvSetReal2D(curcoords,3,0,cvGetReal2D(regpos,h,0));
```

```
                cvSetReal2D(curcoords,4,0,cvGetReal2D(regpos,h,1));
        }//telos h==-1

        if(l==-1){
                cvSetReal2D(curcoords,5,0,cvGetReal2D(prevcoords,5,0));
                cvSetReal2D(curcoords,6,0,cvGetReal2D(prevcoords,6,0));
        }
        else
        {
                cvSetReal2D(curcoords,5,0,cvGetReal2D(regpos,l,0));
                cvSetReal2D(curcoords,6,0,cvGetReal2D(regpos,l,1));
        }//telos l==-1

        //edo bgazo otidhpote axrhsto apo mnhmh
        cvReleaseImage(&bweqI);
        cvReleaseImage(&bw);
        cvReleaseImage(&msPOS);
        //Telos Release
        return curcoords;
}//telos cvProcSkinDetection
```

cvProcessImgSeq

```
IplImage* cvProcessImgSeq(char MainImgName[],int NumberOfImgs,char ImgType[]){
        CvSize kopsia1;
        IplImage* Img1;
        IplImage* Img2;
        char C1,C2,C3;
        int i2,j2;
        char ImgNumber[]="000";
        char enumeration[]="0123456789";
        char MainImgNameTmp[]="";
        strcpy(MainImgNameTmp,MainImgName);

        char ImgTypeTmp[] = ".bmp";
        strcpy(ImgTypeTmp,ImgType);

        char CurrentFileName[] = "";
        char CurrentFileNameTempDef[] = "";
        char FilenameForProps[] = "";

        IplImage* curcoords;
        kopsia1.height=7;
        kopsia1.width=NumberOfImgs-1;
        curcoords=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
```

```
IplImage* Tmpcoords;
kopsia1.height=7;
kopsia1.width=1;
Tmpcoords=cvCreateImage(kopsia1,IPL_DEPTH_32F,1);
cvSetReal2D(Tmpcoords,0,0,-1);
cvSetReal2D(Tmpcoords,1,0,-1);
cvSetReal2D(Tmpcoords,2,0,-1);
cvSetReal2D(Tmpcoords,3,0,-1);
cvSetReal2D(Tmpcoords,4,0,-1);
cvSetReal2D(Tmpcoords,5,0,-1);
cvSetReal2D(Tmpcoords,6,0,-1);

//edo na balo mia klish ths getskinprops oste na paroume ta mx kai my
double zhue;
double zsat;
double zval;
double mx;
double my;
double mhue;
double msat;
double mval;

strcpy(FilenameForProps,MainImgName);
strcat(FilenameForProps,"000");
strcat(FilenameForProps,ImgType);
IplImage* ImgForProps= cvLoadImage(FilenameForProps);
//cvNamedWindow( "ImgForProps", 1 );
//cvShowImage( "ImgForProps",ImgForProps);
IplImage* Getskin;
Getskin=cvGetSkinProps(ImgForProps,&zhue,&zsat,&zval,&mx,&my,&mhu
e,&msat,&mval);
printf("Mx and My are calculated mx=%fl my=%fl  \n",mx,my);
//telos eureseis mx my

        IplImage* ms;
        IplImage* cmout;
        IplImage* mmout;

        kopsia1.height=ImgForProps->height;
        kopsia1.width=ImgForProps->width;
        ms=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
        cmout=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);
        mmout=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

for(i2=0;i2<NumberOfImgs-1;i2++){
```

```
        Bring3digitsBack(i2,&C1,&C2,&C3);
        ImgNumber[0]=C1;
        ImgNumber[1]=C2;
        ImgNumber[2]=C3;
        strcpy(CurrentFileName,MainImgName);
        strcat(CurrentFileName,ImgNumber);
        strcat(CurrentFileName,ImgType);
        Img1= cvLoadImage(CurrentFileName);
        //cvNamedWindow( CurrentFileName, 1 );
        //cvShowImage( CurrentFileName,Img1);

        Bring3digitsBack(i2+1,&C1,&C2,&C3);
        ImgNumber[0]=C1;
        ImgNumber[1]=C2;
        ImgNumber[2]=C3;
        strcpy(CurrentFileName,MainImgName);
        strcat(CurrentFileName,ImgNumber);
        strcat(CurrentFileName,ImgType);
        Img2= cvLoadImage(CurrentFileName);
        //cvNamedWindow( CurrentFileName, 1 );
        //cvShowImage( CurrentFileName,Img2);



        //cvFindMovingSkin(Img1,Img2,0.05267,-
0.033367,ms,cmout,mmout);
        cvFindMovingSkin(Img1,Img2,mx,my,ms,cmout,mmout);

Tmpcoords=cvProcSkinDetection(ms,cmout,mmout,Tmpcoords);

        cvSetReal2D(curcoords,0,i2,-1);
        cvSetReal2D(curcoords,1,i2,cvGetReal2D(Tmpcoords,1,0));
        cvSetReal2D(curcoords,2,i2,cvGetReal2D(Tmpcoords,2,0));
        cvSetReal2D(curcoords,3,i2,cvGetReal2D(Tmpcoords,3,0));
        cvSetReal2D(curcoords,4,i2,cvGetReal2D(Tmpcoords,4,0));
        cvSetReal2D(curcoords,5,i2,cvGetReal2D(Tmpcoords,5,0));
        cvSetReal2D(curcoords,6,i2,cvGetReal2D(Tmpcoords,6,0));
        //printf("Right Hand:(%fl,%fl) Head:(%fl,%fl) Left Hand:(%fl,%fl)
Turn:%d
\n",cvGetReal2D(Tmpcoords,1,0),cvGetReal2D(Tmpcoords,2,0),cvGetReal2
D(Tmpcoords,3,0),cvGetReal2D(Tmpcoords,4,0),cvGetReal2D(Tmpcoords,5,
0),cvGetReal2D(Tmpcoords,6,0),i2);
        printf("Right Hand:(%d,%d) Head:(%d,%d) Left Hand:(%d,%d)
Turn:%d
\n",cvRound(cvGetReal2D(Tmpcoords,1,0)),cvRound(cvGetReal2D(Tmpcoo
rds,2,0)),cvRound(cvGetReal2D(Tmpcoords,3,0)),cvRound(cvGetReal2D(Tm
```

```
        pcoords,4,0)),cvRound(cvGetReal2D(Tmpcoords,5,0)),cvRound(cvGetReal2
        D(Tmpcoords,6,0)),i2);
                //edo mporo na balo merika release gia img1 kai img2
                cvReleaseImage(&ImgForProps);
                cvReleaseImage(&Getskin);
                cvReleaseImage(&Img1);
                cvReleaseImage(&Img2);
                //telos release
        }//telos i2
        printf("Movement Detection Completed  \n");
        return curcoords;
}//telos
```

ShowSkinMovementOnImgManual

```
void            ShowSkinMovementOnImgManual(IplImage*           Coords,char
MainImgName[],char ImgType[],bool SaveFile){
        CvSize kopsia1;
        char C1,C2,C3;
        //int i2,j2;
        char ImgNumber[]="000";
        char CurrentFileName[] = " ";
        char enumeration[]="0123456789";

        IplImage* Img1;
        IplImage* Img1Grayscale;

        for(int i2=0;i2<Coords->width;i2++){
                Bring3digitsBack(i2,&C1,&C2,&C3);
                char ImgNumber[]="000";
                ImgNumber[0]=C1;
                ImgNumber[1]=C2;
                ImgNumber[2]=C3;
                strcpy(CurrentFileName,MainImgName);
                strcat(CurrentFileName,ImgNumber);
                strcat(CurrentFileName,ImgType);
                Img1= cvLoadImage(CurrentFileName);

                kopsia1.height=Img1->height;
                kopsia1.width=Img1->width;
                Img1Grayscale=cvCreateImage(kopsia1,IPL_DEPTH_8U,1);

                cvCvtColor(Img1,Img1Grayscale,CV_RGB2GRAY);
                //tora na kano to zografisma
                //test malakias
```

```
		// printf("Coords (3,%d)=%d kai Coords (4,%d)=%d \n",i2,cvRound(cvGetReal2D(Coords,3,i2)),i2,cvRound(cvGetReal2D(Coords,4,i2)));
		//printf("i2=%d kai Coords->height=%d kai Coords->width=%d \n",i2,Coords->height,Coords->width);
	Img1=DrawH(Img1Grayscale,cvRound(cvGetReal2D(Coords,3,i2)),cvRound(cvGetReal2D(Coords,4,i2)),255);

	Img1=DrawL(Img1,cvRound(cvGetReal2D(Coords,1,i2)),cvRound(cvGetReal2D(Coords,2,i2)),255);

	Img1=DrawR(Img1,cvRound(cvGetReal2D(Coords,5,i2)),cvRound(cvGetReal2D(Coords,6,i2)),255);

		cvNamedWindow( "MovementPositioning", 1 );
		cvShowImage( "MovementPositioning",Img1);
		if(SaveFile=true){
			strcpy(CurrentFileName,MainImgName);
			strcat(CurrentFileName,ImgNumber);
			strcat(CurrentFileName,"Result");
			strcat(CurrentFileName,ImgType);
			cvSaveImage(CurrentFileName,Img1);
		}
		//edo bgazo otidhpote axrhsto apo mnhmh
		cvReleaseImage(&Img1);
		cvReleaseImage(&Img1Grayscale);
		//Telos Release
		cvWaitKey();
	}//telos i2
}//telos ShowSkinMovementOnImgManual
```

Bring3digitsBack

```
void Bring3digitsBack(int InputNumber,char* C1,char* C2,char* C3){
		char enumeration[]="0123456789";
		if(InputNumber<10){
			*C1=enumeration[0];
			*C2=enumeration[0];
			*C3=enumeration[InputNumber];
		}
		if(InputNumber<100){
			*C1=enumeration[0];
			*C2=enumeration[cvRound(InputNumber/10)];
```

```
                        *C3=enumeration[cvRound(InputNumber-
        cvRound(InputNumber/10)*10)];

                }
                if(InputNumber<1000){
                        *C1=enumeration[cvRound(InputNumber/100)];
                                *C2=enumeration[cvRound((InputNumber-
        cvRound(InputNumber/100)*100)/10)];
                        *C3=enumeration[cvRound(InputNumber-
        cvRound(InputNumber/100)*100-cvRound((InputNumber-
        cvRound(InputNumber/100)*100)/10)*10)];

                }
                if(InputNumber>1000){
                        printf("Smaller NumberOfImgs plz \n");
                }
}//telos Bring3digitsBack
```

DrawR

```
IplImage* DrawR(IplImage* Source,int x,int y,int colour){
                IplImage* ADDH;
                CvSize kopsia1;
                kopsia1.height=Source->height;
                kopsia1.width=Source->width;
                ADDH=cvCreateImage(kopsia1,Source->depth,1);
                for(int i2=0;i2<Source->height;i2++)
                        for(int j2=0;j2<Source->width;j2++){

        cvSetReal2D(ADDH,i2,j2,cvGetReal2D(Source,i2,j2));
                        }//telos j2
                if(x+10<Source->height && y+5<Source->width ){
                        cvSetReal2D(ADDH,x,y,colour);
                        cvSetReal2D(ADDH,x+1,y,colour);
                        cvSetReal2D(ADDH,x+2,y,colour);
                        cvSetReal2D(ADDH,x+3,y,colour);
                        cvSetReal2D(ADDH,x+4,y,colour);
                        cvSetReal2D(ADDH,x+5,y,colour);
                        cvSetReal2D(ADDH,x+6,y,colour);
                        cvSetReal2D(ADDH,x+7,y,colour);
                        cvSetReal2D(ADDH,x+8,y,colour);
                        cvSetReal2D(ADDH,x+9,y,colour);
                        cvSetReal2D(ADDH,x+10,y,colour);

                        cvSetReal2D(ADDH,x,y+1,colour);
```

```
            cvSetReal2D(ADDH,x+1,y+1,colour);
            cvSetReal2D(ADDH,x+2,y+1,colour);
            cvSetReal2D(ADDH,x+3,y+1,colour);
            cvSetReal2D(ADDH,x+4,y+1,colour);
            cvSetReal2D(ADDH,x+5,y+1,colour);
            cvSetReal2D(ADDH,x+6,y+1,colour);
            cvSetReal2D(ADDH,x+7,y+1,colour);
            cvSetReal2D(ADDH,x+8,y+1,colour);
            cvSetReal2D(ADDH,x+9,y+1,colour);
            cvSetReal2D(ADDH,x+10,y+1,colour);

            cvSetReal2D(ADDH,x,y+2,colour);
            cvSetReal2D(ADDH,x+1,y+2,colour);
            cvSetReal2D(ADDH,x+3,y+2,colour);
            cvSetReal2D(ADDH,x+4,y+2,colour);
            cvSetReal2D(ADDH,x+6,y+2,colour);
            cvSetReal2D(ADDH,x+7,y+2,colour);

            cvSetReal2D(ADDH,x,y+3,colour);
            cvSetReal2D(ADDH,x+1,y+3,colour);
            cvSetReal2D(ADDH,x+3,y+3,colour);
            cvSetReal2D(ADDH,x+4,y+3,colour);
            cvSetReal2D(ADDH,x+7,y+3,colour);
            cvSetReal2D(ADDH,x+8,y+3,colour);

            cvSetReal2D(ADDH,x,y+4,colour);
            cvSetReal2D(ADDH,x+1,y+4,colour);
            cvSetReal2D(ADDH,x+2,y+4,colour);
            cvSetReal2D(ADDH,x+3,y+4,colour);
            cvSetReal2D(ADDH,x+4,y+4,colour);
            cvSetReal2D(ADDH,x+8,y+4,colour);
            cvSetReal2D(ADDH,x+9,y+4,colour);
            cvSetReal2D(ADDH,x+10,y+4,colour);

            cvSetReal2D(ADDH,x,y+5,colour);
            cvSetReal2D(ADDH,x+1,y+5,colour);
            cvSetReal2D(ADDH,x+2,y+5,colour);
            cvSetReal2D(ADDH,x+3,y+5,colour);
            cvSetReal2D(ADDH,x+4,y+5,colour);
            cvSetReal2D(ADDH,x+9,y+5,colour);
            cvSetReal2D(ADDH,x+10,y+5,colour);
    }
    else
    {
            cvSetReal2D(ADDH,x,y,colour);
    }
```

```
		return ADDH;
}//DrawR
```

DrawL

```
IplImage* DrawL(IplImage* Source,int x,int y,int colour){
		IplImage* ADDH;
		CvSize kopsia1;
		kopsia1.height=Source->height;
		kopsia1.width=Source->width;
		ADDH=cvCreateImage(kopsia1,Source->depth,1);
		for(int i2=0;i2<Source->height;i2++)
			for(int j2=0;j2<Source->width;j2++){

	cvSetReal2D(ADDH,i2,j2,cvGetReal2D(Source,i2,j2));
			}//telos j2
		if(x+10<Source->height && y+5<Source->width ){
			cvSetReal2D(ADDH,x+1,y,colour);
			cvSetReal2D(ADDH,x+2,y,colour);
			cvSetReal2D(ADDH,x+3,y,colour);
			cvSetReal2D(ADDH,x+4,y,colour);
			cvSetReal2D(ADDH,x+5,y,colour);
			cvSetReal2D(ADDH,x+6,y,colour);
			cvSetReal2D(ADDH,x+7,y,colour);
			cvSetReal2D(ADDH,x+8,y,colour);
			cvSetReal2D(ADDH,x+9,y,colour);
			cvSetReal2D(ADDH,x+10,y,colour);

			cvSetReal2D(ADDH,x+1,y+1,colour);
			cvSetReal2D(ADDH,x+2,y+1,colour);
			cvSetReal2D(ADDH,x+3,y+1,colour);
			cvSetReal2D(ADDH,x+4,y+1,colour);
			cvSetReal2D(ADDH,x+5,y+1,colour);
			cvSetReal2D(ADDH,x+6,y+1,colour);
			cvSetReal2D(ADDH,x+7,y+1,colour);
			cvSetReal2D(ADDH,x+8,y+1,colour);
			cvSetReal2D(ADDH,x+9,y+1,colour);
			cvSetReal2D(ADDH,x+10,y+1,colour);

			cvSetReal2D(ADDH,x+1,y+2,colour);
			cvSetReal2D(ADDH,x+2,y+2,colour);
			cvSetReal2D(ADDH,x+3,y+2,colour);
			cvSetReal2D(ADDH,x+4,y+2,colour);
			cvSetReal2D(ADDH,x+5,y+2,colour);
			cvSetReal2D(ADDH,x+6,y+2,colour);
			cvSetReal2D(ADDH,x+7,y+2,colour);
```

```
                cvSetReal2D(ADDH,x+8,y+2,colour);
                cvSetReal2D(ADDH,x+9,y+2,colour);
                cvSetReal2D(ADDH,x+10,y+2,colour);

                cvSetReal2D(ADDH,x+8,y+3,colour);
                cvSetReal2D(ADDH,x+8,y+4,colour);
                cvSetReal2D(ADDH,x+8,y+5,colour);

                cvSetReal2D(ADDH,x+9,y+3,colour);
                cvSetReal2D(ADDH,x+9,y+4,colour);
                cvSetReal2D(ADDH,x+9,y+5,colour);

                cvSetReal2D(ADDH,x+10,y+3,colour);
                cvSetReal2D(ADDH,x+10,y+4,colour);
                cvSetReal2D(ADDH,x+10,y+5,colour);


        }
        else
        {
                cvSetReal2D(ADDH,x,y,colour);
        }
        return ADDH;
}//DrawL
```

DrawH

```
IplImage* DrawH(IplImage* Source,int x,int y,int colour){
        IplImage* ADDH;
        CvSize kopsia1;
        kopsia1.height=Source->height;
        kopsia1.width=Source->width;
        ADDH=cvCreateImage(kopsia1,Source->depth,1);
        for(int i2=0;i2<Source->height;i2++)
                for(int j2=0;j2<Source->width;j2++){

    cvSetReal2D(ADDH,i2,j2,cvGetReal2D(Source,i2,j2));
                }//telos j2
        if(x+8<Source->height && y+5<Source->width ){
                cvSetReal2D(ADDH,x+1,y,colour);
                cvSetReal2D(ADDH,x+2,y,colour);
                cvSetReal2D(ADDH,x+3,y,colour);
                cvSetReal2D(ADDH,x+4,y,colour);
                cvSetReal2D(ADDH,x+5,y,colour);
                cvSetReal2D(ADDH,x+6,y,colour);
                cvSetReal2D(ADDH,x+7,y,colour);
```

```
                cvSetReal2D(ADDH,x+8,y,colour);

                cvSetReal2D(ADDH,x+1,y+1,colour);
                cvSetReal2D(ADDH,x+2,y+1,colour);
                cvSetReal2D(ADDH,x+3,y+1,colour);
                cvSetReal2D(ADDH,x+4,y+1,colour);
                cvSetReal2D(ADDH,x+5,y+1,colour);
                cvSetReal2D(ADDH,x+6,y+1,colour);
                cvSetReal2D(ADDH,x+7,y+1,colour);
                cvSetReal2D(ADDH,x+8,y+1,colour);

                cvSetReal2D(ADDH,x+4,y+2,colour);
                cvSetReal2D(ADDH,x+4,y+3,colour);

                cvSetReal2D(ADDH,x+1,y+4,colour);
                cvSetReal2D(ADDH,x+2,y+4,colour);
                cvSetReal2D(ADDH,x+3,y+4,colour);
                cvSetReal2D(ADDH,x+4,y+4,colour);
                cvSetReal2D(ADDH,x+5,y+4,colour);
                cvSetReal2D(ADDH,x+6,y+4,colour);
                cvSetReal2D(ADDH,x+7,y+4,colour);
                cvSetReal2D(ADDH,x+8,y+4,colour);

                cvSetReal2D(ADDH,x+1,y+5,colour);
                cvSetReal2D(ADDH,x+2,y+5,colour);
                cvSetReal2D(ADDH,x+3,y+5,colour);
                cvSetReal2D(ADDH,x+4,y+5,colour);
                cvSetReal2D(ADDH,x+5,y+5,colour);
                cvSetReal2D(ADDH,x+6,y+5,colour);
                cvSetReal2D(ADDH,x+7,y+5,colour);
                cvSetReal2D(ADDH,x+8,y+5,colour);

        }
        else
        {
                cvSetReal2D(ADDH,x,y,colour);
        }
        return ADDH;
}//DrawH
```

cvTakeRowFromImage

```
IplImage* cvTakeRowFromImage(IplImage* Source,int x){
        IplImage* RowFromImg;
        CvSize kopsia1;
```

```
		kopsia1.height=1;
		kopsia1.width=Source->width;
		RowFromImg=cvCreateImage(kopsia1,Source->depth,1);
		for(int j1=0;j1<Source->width;j1++){

	cvSetReal1D(RowFromImg,j1,cvGetReal2D(Source,x,j1));
			}//telos j1
		return RowFromImg;
}//cvTakeRowFromImage
```

cvMakeRowImage

```
IplImage* cvMakeRowImage(IplImage* Source,int i,int j){
		IplImage* NImage;
		CvSize kopsia1;
		kopsia1.height=i;
		kopsia1.width=j;
		NImage=cvCreateImage(kopsia1,Source->depth,1);
		for(int i1=0;i1<i;i1++)
			for(int j1=0;j1<j;j1++){
				//uchar*
		AImg=&CV_IMAGE_ELEM(NImage,uchar,i1,j1);

	//AImg[0]=CV_IMAGE_ELEM(Source,uchar,0,i1*j+j1);

	cvSetReal2D(NImage,i1,j1,cvGetReal1D(Source,i1*j+j1));
			}//telos j1
		return NImage;
}
```

cvAddColumToImage

```
IplImage* cvAddColumToImage(IplImage* Source,IplImage* Addition){  //an thelo
na kano [x1 x2 x3] kano 2 fores thn sunartish
	if(Source->height==Addition->height)
	{
		IplImage* AddCol;
		CvSize kopsia1;
		kopsia1.height= Source->height;
		kopsia1.width=(Source->width) + Addition->width;
		AddCol=cvCreateImage(kopsia1,Source->depth,1);
		for(int i1=0;i1<Source->height;i1++)
			for(int j1=0;j1<Source->width;j1++){
				//uchar*
		AImg=&CV_IMAGE_ELEM(Source,uchar,i1,j1);
```

```
                        //uchar*
                BImg=&CV_IMAGE_ELEM(AddCol,uchar,i1,j1);
                        //BImg[0]=AImg[0];

        cvSetReal2D(AddCol,i1,j1,cvGetReal2D(Source,i1,j1));
                        }//telos j1
                for(int i11=0;i11<Source->height;i11++)
                        for(int         j11=Source->width;j11<Source->width+Addition-
                >width;j11++){
                        //uchar*
                AImg=&CV_IMAGE_ELEM(Addition,uchar,i11,j11-Source->width);
                        //uchar*
                BImg=&CV_IMAGE_ELEM(AddCol,uchar,i11,j11);
                        //BImg[0]=AImg[0];
                cvSetReal2D(AddCol,i11,j11,cvGetReal2D(Addition,i11,j11-Source-
                >width));
                        }//telos j11
                return AddCol;
        }
}//telos cvAddColumToImage
```

cvMakeImageCol

```
IplImage* cvMakeImageCol(IplImage* Source){// to (:)
                IplImage* Col;
                CvSize kopsia1;
                kopsia1.height=(Source->height*Source->width);
                kopsia1.width=1;
                Col=cvCreateImage(kopsia1,Source->depth,1);
                for(int i1=0;i1<Source->height;i1++)
                        for(int j1=0;j1<Source->width;j1++){
                        //uchar*
                AImg=&CV_IMAGE_ELEM(Source,uchar,i1,j1);
                        //uchar*
                BImg=&CV_IMAGE_ELEM(Col,uchar,i1*Source->width+j1,0);
                        //BImg[0]=AImg[0];
                        cvSetReal1D(Col,i1*Source-
                >width+j1,cvGetReal2D(Source,i1,j1));
                        }
                return Col;
}//telos cvMakeImageCol
```

cvImReconstruct

```
void cvImReconstruct(IplImage* A,IplImage* B,IplImage* Destination){//A einai h dynath eikona bash ths opoias tha dextoume h oxi thn B kai prepei na einai kai oi 2 logikes
		int c1,c2;
		cvSetZero(Destination);

		CvSize kopsia1;
		IplImage* A1;
		kopsia1.height=A->height;
		kopsia1.width=A->width;
		A1=cvCreateImage(kopsia1,A->depth,1);
		cvSetZero(A1);

		IplImage* B1;
		kopsia1.height=B->height;
		kopsia1.width=B->width;
		B1=cvCreateImage(kopsia1,B->depth,1);
		cvSetZero(B1);

		cvBwLabel(A,A1);
		cvBwLabel(B,B1);

		for(int i2=0;i2<A1->height;i2++)
			for(int j2=0;j2<A1->width;j2++){
				if(CV_IMAGE_ELEM(A,uchar,i2,j2)==1)
					{
						uchar*
		RcS=&CV_IMAGE_ELEM(Destination,uchar,i2,j2);
						RcS[0]=cvRound(1);
					}
				}//telos j2

		for(int i1=0;i1<A1->height;i1++) //auto eksasfalizei oti ta koina B kai A tha emfanistoun
			for(int j1=0;j1<A1->width;j1++){
				if(CV_IMAGE_ELEM(A1,uchar,i1,j1)!=0 &&
		CV_IMAGE_ELEM(B1,uchar,i1,j1)!=0 )
				{

	c1=CV_IMAGE_ELEM(A1,uchar,i1,j1);

	c2=CV_IMAGE_ELEM(B1,uchar,i1,j1);

	for(int i2=0;i2<A1->height;i2++) //syblhrono
		for(int j2=0;j2<A1->width;j2++){
```

```
        if(CV_IMAGE_ELEM(A1,uchar,i2,j2)==c1                                    ||
        CV_IMAGE_ELEM(B1,uchar,i2,j2)==c2)
                                                                {
                                                                        uchar*
        RcS=&CV_IMAGE_ELEM(Destination,uchar,i2,j2);

        RcS[0]=cvRound(1);
                                                                }
                                                        }//telos j2

                                }//telos tou If koinou stoixeiou
                        }//telos j1
                //arxh Release
                cvReleaseImage(&A1);
                cvReleaseImage(&B1);
                //telos Release
}//telos cvImReconstruct
```

cvBwLabel

```
void cvBwLabel(IplImage* Source,IplImage* Destination){
        int Labeler=2;
        for(int i11=0;i11<Source->height;i11++)
                for(int j11=0;j11<Source->width;j11++){
                        uchar* bi2=&CV_IMAGE_ELEM(Source,uchar,i11,j11);
                        uchar* bi3=&CV_IMAGE_ELEM(Destination,uchar,i11,j11);
                        bi3[0]=bi2[0];
                }//telos j1
        for(int i1=0;i1<Source->height;i1++)
                for(int j1=0;j1<Source->width;j1++){
                        if(CV_IMAGE_ELEM(Destination,uchar,i1,j1)==1)
                        {

        cvMakeMyCloseFriendLikeMe(Destination,i1,j1,Labeler);
                                Labeler++;
                        }
                }//telos j1

}//telos cvBwLabel
```

cvMakeMyCloseFriendLikeMe

```
void cvMakeMyCloseFriendLikeMe(IplImage* Destination,int x,int y,int Labeler) {
//na do an bgainei ektos bound
                //if (y==100) printf("Problhma  (%d,%d) \n",x,y);
                fr1=&CV_IMAGE_ELEM(Destination,uchar,x,y);
```

```
		fr1[0]=cvRound(Labeler);
		//(x+1,y)@@@@@@@@@@@1
		if(x+1<Destination->height){
			if(CV_IMAGE_ELEM(Destination,uchar,x+1,y)==1){

	cvMakeMyCloseFriendLikeMe(Destination,x+1,y,Labeler);
			}
		}//telos if x+1,y

		//(x+1,y-1)@@@@@@@@@@2
		if(x+1<Destination->height && y-1>0){
		if(CV_IMAGE_ELEM(Destination,uchar,x+1,y-1)==1){
	cvMakeMyCloseFriendLikeMe(Destination,x+1,y-1,Labeler);
			}
		}//telos if x+1,y-1

		//(x+1,y+1)@@@@@@@@@@3
	if(x+1<Destination->height && y+1<Destination->width){
			if(CV_IMAGE_ELEM(Destination,uchar,x+1,y+1)==1){
	cvMakeMyCloseFriendLikeMe(Destination,x+1,y+1,Labeler);
			}
		}//telos if x+1,y+1

		//(x,y-1)@@@@@@@@@@4
		if(y-1>0){
		if(CV_IMAGE_ELEM(Destination,uchar,x,y-1)==1){
				cvMakeMyCloseFriendLikeMe(Destination,x,y-
1,Labeler);
			}
		}//telos if x,y-1

		//(x,y+1)@@@@@@@@@@5
		if(y+1<Destination->width){
			if(CV_IMAGE_ELEM(Destination,uchar,x,y+1)==1){

	cvMakeMyCloseFriendLikeMe(Destination,x,y+1,Labeler);
			}
		}//telos if x,y+1

		//(x-1,y)@@@@@@@@@@6
		if(x-1>0){
		if(CV_IMAGE_ELEM(Destination,uchar,x-1,y)==1){
			//	cvMakeMyCloseFriendLikeMe(Destination,x-
1,y,Labeler);
			}
		}//telos if x-1,y
```

```
		//(x-1,y-1)@@@@@@@@@@7
		if(x-1>0 && y-1>0){
	if(CV_IMAGE_ELEM(Destination,uchar,x-1,y-1)==1){
				//	cvMakeMyCloseFriendLikeMe(Destination,x-1,y-
1,Labeler);
				}
		}//telos if x-1,y-1

		//(x-1,y+1)@@@@@@@@@@8
		if(x-1>0 && y+1<Destination->width){
	if(CV_IMAGE_ELEM(Destination,uchar,x-1,y+1)==1){
	cvMakeMyCloseFriendLikeMe(Destination,x-1,y+1,Labeler);
				}
		}//telos if x-1,y+1
}//telos cvMakeMyCloseFriendLikeMe
```

MakeRoiPOly

```
void MakeRoiPOly(IplImage* mask,int a[11][2]){ //Na bgazo int pinaka an xreiastei
sto kato kato
	a[10][1]=a[0][1];
	a[10][0]=a[0][0];
	int fillY1,fillY2;
	int ci1,cj1,minC;
	int aT[8];
	int tempX,tempY;
	cvSetZero(mask);   //zero=mauro
	for(ci1=0;ci1<10;ci1++){
		uchar* roi1 =&CV_IMAGE_ELEM(mask,uchar,a[ci1][0],a[ci1][1]);
		roi1[0]=cvRound(255);
		tempX=a[ci1][0];
		tempY=a[ci1][1];
		while(tempX!=a[ci1+1][0] || tempY!=a[ci1+1][1])
		{
aT[0]=abs(a[ci1+1][0]-(tempX))+abs(a[ci1+1][1]-(tempY-1));
aT[1]=abs(a[ci1+1][0]-(tempX+1))+abs(a[ci1+1][1]-(tempY-1));
aT[2]=abs(a[ci1+1][0]-(tempX+1))+abs(a[ci1+1][1]-(tempY));
aT[3]=abs(a[ci1+1][0]-(tempX+1))+abs(a[ci1+1][1]-(tempY+1));
aT[4]=abs(a[ci1+1][0]-(tempX))+abs(a[ci1+1][1]-(tempY+1));
aT[5]=abs(a[ci1+1][0]-(tempX-1))+abs(a[ci1+1][1]-(tempY+1));
aT[6]=abs(a[ci1+1][0]-(tempX-1))+abs(a[ci1+1][1]-(tempY));
aT[7]=abs(a[ci1+1][0]-(tempX-1))+abs(a[ci1+1][1]-(tempY-1));
		minC=findmin(aT);
		if (minC==0) {tempX=tempX;tempY=tempY-1;}
		if (minC==1) {tempX=tempX+1;tempY=tempY-1;}
```

```
                if (minC==2) {tempX=tempX+1;tempY=tempY;}
                if (minC==3) {tempX=tempX+1;tempY=tempY+1;}
                if (minC==4) {tempX=tempX;tempY=tempY+1;}
                if (minC==5) {tempX=tempX-1;tempY=tempY+1;}
                if (minC==6) {tempX=tempX-1;tempY=tempY;}
                if (minC==7) {tempX=tempX-1;tempY=tempY-1;}
uchar* roi2 =&CV_IMAGE_ELEM(mask,uchar,tempX,tempY);
                roi2[0]=cvRound(255);
                }//telos while
        }//telos for

        //GEmisma tou polugonou gia etoimasia maskas
                //if(CV_IMAGE_ELEM(mask,uchar,tempX,tempY)==255)
printf("Anagnorizei to xroma mesa sthn sunartish o pousths \n");
        for(ci1=0;ci1<mask->height;ci1++){

                        minC=0;
                        for(cj1=0;cj1<mask->width;cj1++){
        if(CV_IMAGE_ELEM(mask,uchar,ci1,cj1)==255   &&   minC>=1){      //edo
        kanei mia blakeia na do ti einai
                                        fillY2=cj1;
                                        minC++;
                                }
        if(CV_IMAGE_ELEM(mask,uchar,ci1,cj1)==255 && minC==0){
                                        fillY1=cj1;
                                        fillY2=cj1;
                                        minC++;
                                }

                                        //printf("Kolhma                                %d,%d
        (%d,%d)\n",ci1,cj1,a[0][0],a[0][1]);
                        }//telos cj1

                        if(minC>=2){
                                for(cj1=fillY1;cj1<fillY2;cj1++){
                                        uchar*                                          roi2
        =&CV_IMAGE_ELEM(mask,uchar,ci1,cj1);
                                        roi2[0]=cvRound(255);

                                }
                        }

        }//telos ci1

}//telos MakeRoiPOly
```

MakeEveryKToG

```
void MakeEveryKToG(IplImage* Source,IplImage* Destination,int k,int g){
        for(int i1=0;i1<Source->height;i1++)
                for(int j1=0;j1<Source->width;j1++){
                        if(CV_IMAGE_ELEM(Source,uchar,i1,j1)==k)
                        {
                                uchar*
bi2=&CV_IMAGE_ELEM(Destination,uchar,i1,j1);
                                        bi2[0]=cvRound(g);  //255 aspro
                        }
                        else
                        {

                        }
                }//telos j1
                //printf("An diksei 414720=%d Paizei akoma h malakia \n",cj);
}//telos MakeImage255To1
```

on_mouse

```
void on_mouse( int event, int x, int y, int flags, void* param )
{


   if( event == CV_EVENT_LBUTTONDOWN )
   {
                if(ci>9)
                {
                        printf("Enough Cordinates Given\n");
                        return;
                }
      RoiPoly[ci][0]=y;
                RoiPoly[ci][1]=x;
                ci++;
                printf("Saved to file (%d,%d)\n",x,y);

   }
        return;
}
```

cvMeanM

```
void cvMeanM(IplImage* src, IplImage* dst){
                IplImage* tmpArr;
                //to width einai auto pou tha exoume sto telos!
```

```
                    //ara o tmpArr tha prepei na exei 1 width kai height src->heigth oste na
                    bgalei thn timh ths sthlhs
                    CvScalar stilavg;
                    CvSize kopsia1;
                    kopsia1.height=src->height;
                    kopsia1.width=1;
                    tmpArr=cvCreateImage(kopsia1,src->depth,1);
                    for(int j6=0;j6<src->width;j6++){
                            //cvSetZero(tmpArr);
                            for(int i6=0;i6<src->height;i6++){
uchar* ptrA =&CV_IMAGE_ELEM(tmpArr,uchar,i6,0);
uchar* ptrB =&CV_IMAGE_ELEM(src,uchar,i6,j6);
                            ptrA[0]=cvRound(ptrB[0]);
                            }//telos i6
                            stilavg=cvAvg(tmpArr);
                        uchar* ptr = &CV_IMAGE_ELEM(dst,uchar,0,j6);
                            ptr[0]=cvRound(stilavg.val[0]);
```

Κ Ε Φ Α Λ Α Ι Ο 8

Αποτελέσματα και παρατηρήσεις

8.1 Εικόνες αποτελεσμάτων

Παρακάτω μέσα από μια σειρά εικόνων φαίνονται τα αποτελέσματα του εντοπισμού δερματικής κίνησης

1)

2)

3)

4)

5)

6)

7)

8)

9)

10)

11)

12)

13)

14)

15)

16)

17)

18)

19)

20)

21)

22)

23)

24)

25)

26)

27)

28)

29)

30)

30-

49-60

8.2 Παρατηρήσεις-Συμπεράσματα

Στις παραπάνω εικόνες φαίνεται καθαρά ο εντοπισμός της δερματικής κίνησης. Το κεφάλι και τα χέρια εντοπίζονται σε κάθε φωτογραφία παρόλη την πολυπλοκότητα των κινήσεων του χρήστη. Το Η υποδηλώνει το κέντρο του κεφαλιού, το R το κέντρο του δεξιού χεριού και το L το κέντρο του αριστερού χεριού. Σε κάποιες περιπτώσεις υπάρχει το ενδεχόμενο να 'μπερδευτεί' το πρόγραμμα και μετά από κάποια κίνηση (ένωση των χεριών, μετακίνηση εκτός εικόνας, κίνηση του δεξιού χεριού στο αριστερό μέρος , κ.λ.π.) να θεωρεί το δεξί χέρι αριστερό και το αριστερό χέρι δεξί, μετά το τέλος της κίνησης όμως αναγνωρίζει τα χέρια πάλι όπως πριν την εμφάνιση της παράξενης κίνησης. Σε περιπτώσεις όπου κατά την διάρκεια του βίντεο εμφανίζονται φαινόμενα τα οποία δυσχεραίνουν τον εντοπισμό δέρματος ή κίνησης (φαινόμενα που αναφέρονται παραπάνω στα προβλήματα εντοπισμού δέρματος και κίνησης) το πρόγραμμα δίνει λανθασμένες θέσεις για το κεφάλι και τα χέρια μέχρι τη στιγμή που θα απομακρυνθεί η αιτία που προκάλεσε την ανεπιθύμητη αλλαγή. Από τα διάφορα πειράματα που έγιναν σε αρκετά βίντεο παρατηρήθηκε ότι οποτεδήποτε κάποιος παράγοντας επηρέαζε αρνητικά τα αποτελέσματα του προγράμματος, μόλις ο ανεπιθύμητος αυτός παράγοντας σταματούσε να δρα, το πρόγραμμα επανερχόταν στη ορθή λειτουργία του δίνοντας τις σωστές θέσεις του κεφαλιού και των χεριών σε κάθε χρονική στιγμή του βίντεο. Επίσης πολύ χρήσιμη και αποτελεσματική είναι η λειτουργία του προγράμματος να ζητάει από τον χρήστη να δηλώσει μια δερματική περιοχή σε μια εικόνα του βίντεο πριν αρχίσει η διαδικασία εντοπισμού δερματικής κίνησης. Με αυτόν τον τρόπο , όπως έχει προαναφερθεί , το πρόγραμμα ξέρει ακριβώς σε ποιες χρωματικες περιοχές να ψάξει προκυμένου να εντοπίσει τις δερματικές περιοχές, μέσα στο βίντεο, μειώνοντας κατά πολύ μια σειρά σφαλμάτων που θα μπορούσαμε να είχαμε εάν είχε εφαρμοσθεί κάποια άλλη μέθοδος. Ακόμα θα πρέπει να παρατηρηθεί ότι επειδή η προσπάθεια εντοπισμού δερματικής κίνησης έγινε με βάση τις υπάρχουσες μεθόδους εντοπισμού δέρματος και κίνησης (οι οποίες αναφέρονται αναλυτικά στις δημοσιεύσεις και έρευνες που έχουν αναλυθεί παραπάνω) έχει τα προβλήματα και τις αποκλίσεις απο το επιθυμητό αποτέλεσμα που οι μέθοδοι αυτές έχουν. Όπως έχει αναφερθεί, ο εντοπισμός δερματικής κίνησης δεν είναι τελειοποιημένος ακόμα, με αποτέλεσμα να απαιτούνται επιπλέον έρευνες προκυμένου να τελειοποιηθεί και να φτάσει την διακριτική ικανότητα του ανθρώπινου ματιού.

Β ι β λ ι ο γ ρ α φ ί α

1)www.intel.com/technology/computing/opencv/

2) http://opencvlibrary.sourceforge.net/

3) http://tech.groups.yahoo.com/group/OpenCV

4)http://cgi.cs.indiana.edu/~oleykin/website/OpenCVHelp/

5)http://opencvlibrary.sourceforge.net/Welcome#head-3af5a18ef4bb86ec39be0716a7232b84fe0705b4

6)http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow/

7)Microsoft Visual Studio 6.0 MSDN Library

8)Wu, Y. and Huang, T.S., "Hand modeling, analysis, and recognition for vision-based human computer interaction", *IEEE Signal Processing Magazine*, 18(3): 51-60, May 2001.

9) Whissel, C.M., The dictionary of affect in language, Emotion: Theory, Research and Experience: vol. 4, The Measurement of Emotions, R. Plutchik and H. Kellerman, Eds., New York: Academic, 1989.

10) Hartmann, B., Mancini, M. and Pelachaud, C., Implementing Expressive Gesture Synthesis for Embodied Conversational Agents. Gesture Workshop (2005) Vannes

11) Wallbott, H.G, Bodily expression of emotion. European Journal of Social Psychology, 28:879–896, 1998.

12) Harrigan, J.A., Listener's body movements and speaking turns. Communication Research, 12(2):233–250, 1985.

13) Gallaher, P., Individual differences in nonverbal behavior: Dimensions of style. Journal of Personality and Social Psychology 63 (1992)

14)de Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V. and De Carolis, B., From Greta's mind to her face: modeling the dynamics of affective states in a Conversational Embodied Agent. *International Journal of Human-Computer Studies*, 59, 81-118, 2003.

15) Maurizio Mancini, Bjoern Hartmann, Catherine Pelachaud, Non-verbal behaviors expressivity and their representation, PF-star report 3.

16) Martin, J.-C., Caridakis, G., Devillers, L., Karpouzis, K., Abrilian, S., Manual Annotation and Image Processing of Multimodal Emotional Behaviours in TV Interviews, accepted for publication to LREC06.

17)Vassilis Athitsos, Michael J. Swain, and Charles Frankel. Distinguishing photographs and graphics on the world wide web. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 10–17, San Juan, Puerto Rico, June 20 1997.

18)Qian Chen, Haiyuan Wu, and Masahiko Yachida. Face detection by fuzzy pattern matching. In *Proc. of Fifth Intl. Conf. on Computer Vision*, pages 591–596, Cambridge, MA, June 1995.

19) Symon D'Oyly Cotton and Ela Claridge. Do all human skin colors lie on a defined surface within LMS space? Technical Report CSR-96-01, School of Computer Science, Univ. of Birmingham, UK, Jan 1996.

20) David A. Forsyth and Margaret M. Fleck. Automatic detection of human nudes. *International Journal of Computer Vision*, 32(1):63–77, August 1999.

21) Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.

22) Yihong Gong and Masao Sakauchi. Detection of regions matching specified chromatic features. *Computer Vision and Image Understanding*, 61(2):263–269, March 1995.

23) Tony S. Jebara and Alex Pentland. Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 144–150, San Juan, Puerto Rico, June 17-19 1997.

24) Rick Kjeldsen and John Kender. Finding skin in color images. In *Proceedings of the International Conference on*
Automatic Face and Gesture Recognition, pages 312–317, Killington, VT, October 14-16 1996.

25) Michael Oren, Constantine Papageorgiou, Pawan Sinha, Edgar Osuna, and Tomaso Poggio. Pedestrian detection using
wavelet templates. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 193–199,
San Juan, Puerto Rico, June 17-19 1997.

26) J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman Publishers, Inc., 1993.

27) R. Redner and H. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Revew*, 26:195–239,
1994.

28) Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on*
Pattern Analysis and Machine Intelligence, 20(1):23–38, January 1998.

29) Michael J. Jones and James M. RehgStatistical. Color Models with Application to Skin Detection Cambridge Research Laboratory Compaq Computer Corporation Cambridge, MA 02142

30) Michael J. Jones and James M. RehgStatistical Color Models with Application to Skin Detection Cambridge Research Laboratory Compaq Computer Corporation Cambridge, MA 02142 Face Segmentation Using Skin-Color Map in Videophone Applications Douglas Chai, Student Member, IEEE, and King N. Ngan, Senior Member, IEEE