Τεχνολογικό Εκπαιδευτικό Ίδρυμα
Τ.Ε.Ι. ΠΕΙΡΑΙΑ

Τμήμα Αυτοματισμού

Πτυχιακή Εργασία

Μελέτη κινηματικής μιας μαριονέτας με χρήση υπολογιστικής όρασης και μικροελεγκτή avr

ΤΕΙ
ΠΕΙΡΑΙΑ
Ιανουάριος
2012

Φοιτητής: Καρκασίνας Θεόδωρος

Α.Μ.: 34496

Επιβλέπων Καθηγητής:

Αλαφοδήμος Κωνσταντίνος

Επιβλέπων Καθηγητής:

Νικολάου Γρηγόριος

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Σκοπός πτυχιακής εργασίας

Στην παρακάτω εργασία αναπτύσσουμε έναν τρόπο αναγνώρισης χειρονομιών ανθρώπινου χεριού, μεταφοράς των δεδομένων μας σε μικροϋπολογιστικό σύστημα και αντιστοίχησης των χειρονομιών σε κινήσεις, τις οποίες πραγματοποιεί μια μαριονέτα με σχοινιά.

Η αναγνώριση χειρονομιών είναι ένα σύνθετο πρόβλημα της επιστήμης των υπολογιστών, το οποίο αποσκοπεί στη μετάφραση των ανθρώπινων χειρονομιών μέσω αλγορίθμων. Χειρονομίες μπορούν να θεωρηθούν κινήσεις του σώματος, του προσώπου ή συχνότερα οι κινήσεις των χεριών. Η συνήθης προσέγγιση για την λύση του προβλήματος περιλαμβάνει την χρήση κάμερας και αλγορίθμων της υπολογιστικής όρασης(computer vision).

Η αναγνώριση χειρονομιών μπορεί να θεωρηθεί ως η προσπάθεια του υπολογιστή να κατανοήσει τη γλώσσα του σώματος. Αυτή η κατανόηση μπορεί να δώσει την δυνατότητα στον χρήστη μιας συσκευής να αλληλεπιδρά με αυτήν χωρίς την χρήση μηχανικών μερών. Με αυτόν τον τρόπο συσκευές εισόδου, όπως το πληκτρολόγιο, το ποντίκι ή ακόμα και οι touch screens μπορούν να θεωρηθούν περιττές.

Η αναγνώριση χειρονομιών μπορεί να συνδυασθεί με την υπολογιστική όραση και την επεξεργασία εικόνας. Ο συνδυασμός αυτός επιφέρει καλύτερα και πιο στιβαρά αποτελέσματα στην πραγματοποίηση του αντικειμενικού μας σκοπού.

Στη περίπτωση μας, εφόσον αναγνωρίσουμε τις χειρονομίες μεταδίδουμε τις πληροφορίες ,μέσω του ασύρματου πρωτόκολλου επικοινωνίας Zigbee, σε μικροελεγκτή. Με αυτόν τον τρόπο δεν κάνουμε software manipulation,όπως συμβαίνει συνήθως, αλλά hardware manipulation.Το γεγονός αυτό διαφοροποιεί τον σκοπό μας στην χρήση της αναγνώρισης χειρονομιών.

Για να πετύχουμε τον στόχο μας χρησιμοποιούμε υπολογιστή,web camera,τον μικροελεγκτή της Atmel, AVR atmega16 και servo motors. Σε όλα τα παραπάνω θα αναφερθούμε αναλυτικότερα στη συνέχεια.

ΚΕΦΑΛΑΙΟ 1°
Γενική Παρουσίαση

Στο κεφάλαιο αυτό αναφερόμαστε στα εργαλεία που χρησιμοποιήσαμε για να πραγματοποιήσουμε την πτυχιακή εργασία. Αυτά είναι το περιβάλλον ανάπτυξης της εργασίας, IDE, βιβλιοθήκες, μικροεπεξεργαστής, Zigbee και servo motors.

1.1 Περιβάλλον εργασίας

Το λειτουργικό σύστημα πάνω στο οποίο αναπτύσσουμε την εργασία μας είναι το Microsoft Windows 7 32-bit. Το γραφικό περιβάλλον επικοινωνίας χρήστη-υπολογιστή(GUI) αλλά και το γεγονός ότι είναι το πιο δημοφιλές λειτουργικό σύστημα το κάνει κατάλληλο για την περίπτωση μας.
Τον αλγόριθμό μας τον αναπτύσσουμε στο περιβάλλον ολοκληρωμένης ανάπτυξης (IDE) του Microsoft Visual Studio 2010. Η γλώσσα προγραμματισμού που χρησιμοποιούμε είναι η C++, η οποία υποστηρίζεται από το IDE με το οποίο εργαζόμαστε.

1.2 Βιβλιοθήκες

Για την ανάπτυξη του κώδικα μας χειριζόμαστε την βιβλιοθήκη της OpenCV2.2 .
Η OpenCV (Open Source Computer Vision Library) που έχει αναπτυχθεί από την Intel και υποστηρίζεται από την Willow Garage , είναι βιβλιοθήκη η οποία περιέχει συναρτήσεις που στοχεύουν κυρίως στην επεξεργασία εικόνας σε πραγματικό χρόνο. Είναι βιβλιοθήκη τύπου ανοικτού κώδικα και μπορεί να χρησιμοποιηθεί από κάθε χρήστη χωρίς περιορισμούς.
Οι εφαρμογές της OpenCV εκτείνονται σε περιοχές όπως:

- Αναγνώριση προσώπου
- Αναγνώριση χειρονομιών
- Αλληλεπίδραση ανθρώπου-υπολογιστή
- Εντοπισμό κίνησης
- Τεχνητά νευρωτικά δίκτυα κ.α.

Η βιβλιοθήκη στην έκδοση 2.2 είναι γραμμένη σε γλώσσα προγραμματισμού C++ και μπορεί να τρέξει σε λειτουργικό σύστημα Microsoft Windows 7, γεγονός που την καθίστα ιδανική επιλογή.

Επιπλέον απαραίτητη είναι και η βιβλιοθήκη cvBlobsLib, η οποία βρίσκει blobs σε μια δυαδική εικόνα και πραγματοποιεί εξαγωγή χαρακτηριστικών από την εικόνα, απαραίτητα για την αναγνώριση χειρονομιών. Έχει αναπτυχθεί χρησιμοποιώντας το Microsoft Visual Studio C++(6.0) και εξαιτίας αυτού κάνουμε μικρές μετατροπές στην βιβλιοθήκη για να είναι συμβατή με το Microsoft Visual Studio 2010.

1.3 Μικροελεγκτής

Στην παρούσα εργασία δουλεύουμε με τον μικροελεγκτή της Atmel, AVR atmega16. Πρόκειται για μικροελεγκτή αρχιτεκτονικής registered-based.
Αυτό σημαίνει ότι πριν την πραγματοποίηση μιας διεργασίας , ο μικροελεγκτής φορτώνει όλα τα απαραίτητα δεδομένα στην κεντρική μονάδα επεξεργασίας. Το αποτέλεσμα της διεργασίας αποθηκεύεται επίσης σε κάποιον καταχωρητή. Ένα σετ εντολών, βασισμένο στην ιδέα RISC(Reduced Instruction Set Computer), συνδυάζεται με την register-based αρχιτεκτονική.

RISC είναι στρατηγική σχεδιασμού CPU, η οποία έχει σκοπό να απλουστεύσει πολύπλοκα προγράμματα και να βελτιώσει την απόδοση του επεξεργαστή. Ένας RISC επεξεργαστής είναι εξοπλισμένος με ένα συμπλήρωμα απλών και αποτελεσματικών λειτουργιών. Περίπλοκες εντολές δημιουργούνται από αυτές τις απλές λειτουργίες. Το αποτέλεσμα είναι η αποδοτικότερη εκτέλεση των προγραμμάτων. Το AVR atmega16 είναι εξοπλισμένο με 131 RISC –τύπου εντολές, οι περισσότερες των οποίων μπορούν να εκτελεστούν σε έναν απλό κύκλο ρολογιού.

Το AVR atmega16 έχει 32 γενικού σκοπού 8-bit καταχωρητές σε συνδυασμό με την λογική αριθμητική μονάδα στην CPU. Επίσης, ο επεξεργαστής είναι σχεδιασμένος σε Harvard architecture μορφή που σε συνδυασμό με το σετ εντολών τύπου RISC, επιτρέπουν στον μικροεπεξεργαστή να ολοκληρώνει εντολές της γλώσσας assembly σε έναν κύκλο ρολογιού. Ο atmega16 μπορεί να εκτελεί 16 εκατομμύρια εντολές το δευτερόλεπτο όταν λειτουργεί σε ταχύτητα ρολογιού των 16 MHz.

1.3.1 *Atmel AVR ATmega16 in 40 pin DIP*

1.3.2 *AVR Atmega16 Pinout Diagram*

Έχουμε την δυνατότητα να παρέχουμε τροφοδοσία στον AVR Atmega16 μέσω των pins VCC,GND,AVCC,AREF. Επίσης με το reset pin μπορούμε να κάνουμε reset τον επεξεργαστή. Επιπλέον έχουμε τέσσερις 8-bit πόρτες (PA0-PA7,PB0-PB7,PC0-PC7,PD0-PD7) με τις οποίες μπορούμε να επικοινωνήσουμε με το περιβάλλον. Αυτές οι πόρτες μπορούν να χρησιμοποιούνται σαν γενικού σκοπού, ψηφιακές πόρτες εισόδου-εξόδου ή για εναλλακτικές λειτουργίες. Ο Atmega16 περιέχει επίσης σύστημα χρονισμού, μετατροπέα αναλογικού σε ψηφιακό σήμα (ADC), σύστημα διακοπών, στοιχεία μνήμης και σύστημα επικοινωνίας .

Ο μικροεπεξεργαστής αποτελείται από μνήμη flash EEPROM, η οποία χρησιμοποιείται για να αποθηκεύουμε προγράμματα. Τα περιεχόμενα της μνήμης διατηρούνται όταν σβήσουμε τον μικροελεγκτή και το μέγεθος της μνήμης είναι 16 Kbytes. Επίσης διαθέτει στατική μνήμη τυχαίας προσπέλασης (SRAM), η οποία σε αντίθεση με την flash EEPROM χάνει τα δεδομένα της όταν σβήσει ο μικροεπεξεργαστής. Μπορούμε να γράψουμε και να διαβάσουνε από την SRAM κατά την διάρκεια εκτέλεσης του προγράμματος. Το μέγεθος της μνήμης είναι 1120 bytes. Κατά την εκτέλεση των προγραμμάτων η RAM χρησιμοποιείται για να αποθηκεύουμε global μεταβλητές, για παροχή τοποθεσίας για την στοίβα και για την υποστήριξη δυναμικής τοποθέτησης μεταβλητών στην μνήμη. Ακόμα υπάρχει και η byte-addressable EEPROM μνήμη στην οποία αποθηκεύουμε μόνιμα μεταβλητές και τις ανακαλούμε κατά την διάρκεια εκτέλεσης ενός προγράμματος. Το μέγεθος της είναι 512 bytes και εφαρμογές στις οποίες μπορούμε να χρησιμοποιήσουμε αυτήν την μνήμη είναι η αποθήκευση παραμέτρων του συστήματος, ηλεκτρονικές κλειδαριές, αυτόματες πόρτες γκαράζ κ.α.

Σημαντικό ρόλο στην επιλογή του συγκεκριμένου μικροελεγκτή έπαιξαν και τα περιφερειακά του χαρακτηριστικά. Αυτά τα χαρακτηριστικά επιτρέπουν στον μικροελεγκτή να εκτελέσει περίπλοκες διεργασίες.

Η ταχύτητα με την οποία εκτελείται ένα πρόγραμμα καθορίζεται από το ρολόι του μικροεπεξεργαστή. Ο avr atmega 16 μπορεί να χρονισθεί εσωτερικά ή εξωτερικά. Μπορούμε να επιλέξουμε συχνότητες λειτουργίας ρολογιού στα 1,2,4 ή 8 MHz. Σε περίπτωση που θέλουμε μεγαλύτερο εύρος επιλογής συχνοτήτων μπορούμε να χρησιμοποιήσουμε μια εξωτερική πηγή χρονισμού, όπως ο ταλαντωτής κρυστάλλου. Επιπλέον είναι εξοπλισμένος με χρονιστές (timers), που επιτρέπουν στον χρήστη να παράγει ακριβή σήματα εξόδου, να βρίσκει τα χαρακτηριστικά ενός ψηφιακού σήματος εισόδου ή να μετράει εξωτερικά συμβάντα. Συγκεκριμένα έχει δύο 8-bit timer/counter και ένα 16-bit counter.

Εξοπλισμένος με τέσσερα Pulse Width Modulated (PWM) κανάλια δίνει την δυνατότητα να ελέγχει ο χρήστης την θέση σέρβο κινητήρων και την ταχύτητα σε DC κινητήρες.

Ο μικροελεγκτής πετυχαίνει την σειριακή επικοινωνία μέσω διαφορετικών υποσυστημάτων όπως, το Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) για full-duplex επικοινωνία μεταξύ πομπού και δέκτη, Two Wire Serial Interface (TWI) για σύνδεση συσκευών μεταξύ τους και το υποσύστημα

Serial Peripheral Interface(SPI) που πομπός και δέκτης μοιράζονται την ίδια πηγή ρολογιού κατά την χρήση αυτού του συστήματος επικοινωνίας.

Το σύστημα διακοπών του μικροεπεξεργαστή περιλαμβάνει 21 πηγές διακοπής. Τρείς από τις οποίες αναφέρονται σε εξωτερικές πηγές.

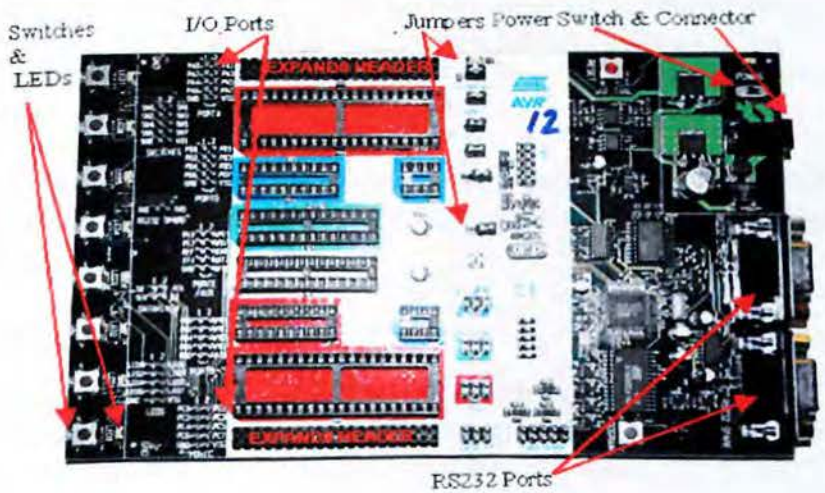
Όσον αφορά την κατανάλωση ρεύματος, ο atmega 16 μπορεί να λειτουργεί από 2.7 έως 5.5 VDC και από 4.5 έως 5.5 VDC. Ανάλογα με την συχνότητα ρολογιού επιλέγουμε και την κατάλληλη τάση τροφοδοσίας ώστε η κατανάλωση ρεύματος να είναι μειωμένη.

STK 500

Σημαντικό στοιχείο στην εργασία μας αποτελεί και η χρήση του αναπτυξιακού stk 500 της Atmel για τον Avr ελεγκτή μας. Επιτρέπει στον χρήστη να έχει εύκολη πρόσβαση στο σύστημα εισόδου εξόδου του μικροελεγκτή που χρησιμοποιεί.

Το αναπτυξιακό προσφέρει :

- AVR Studio Interface.
- RS232 θύρα επικοινωνίας με H/Y για τον προγραμματισμό και έλεγχο.
- Γέφυρα ανόρθωσης 10V - 15V πάνω στη πλακέτα.
- Sockets για 8-, 20-, 28- και 40-pin AVR controller.
- Συριακό (ISP) on-board προγραμματισμό και παράλληλο High Voltage για όλη την οικογένεια AVR
- 8 push button και 8 led γενικής χρήσης
- pin για όλες τις πόρτες για την ευκολότερη διαχείριση του μικροελεγκτή.
- On-board 2 megabit Data Flash για μόνιμη αποθήκευση δεδομένων.
- On-board μεταβλητός ταλαντωτής .
- Δεύτερη RS232 πόρτα γενικής χρήσεως .
- Κονέκτορες επέκτασης για Plug-in Modules και πρότυπες πλακέτες.



1.3.3 *STK500 development board*

WinAVR

Υπάρχουν πολλοί τρόποι που μπορούμε να γράψουμε, να μεταγλωττίσουμε και να φορτώσουμε ένα πρόγραμμα στον μικροελεγκτή μας, αλλά εμείς θα χρησιμοποιήσουμε το λογισμικό που λέγεται WinAVR. Δουλεύει στην πλατφόρμα των Windows 7 και περιέχει τον μεταγλωττιστή GNU GCC για C και C++ γλώσσες προγραμματισμοί οποίος αναφέρεται σαν avr-gcc. Για τους λόγους αυτούς επιλέγουμε το λογισμικό WinAVR.

Συνήθως οι μικροελεγκτές προγραμματίζονται σε γλώσσα assembly. Η assembly χρησιμοποιεί μόνο το βασικό σετ εντολών ενός μικροελεγκτή. Ενώ αυτό μπορεί να παράγει γρήγορους και αποτελεσματικούς κώδικες, περιορίζεται από το γεγονός ότι κάθε επεξεργαστής έχει το δικό του σετ εντολών. Εξαιτίας αυτού θεωρείται μη πρακτική γλώσσα. Η γλώσσα C από την άλλη, εμφανίζεται συχνά στη βιομηχανία και εφαρμόζεται σε διαφορετικές πλατφόρμες. Με αυτήν την γλώσσα μπορούμε να προγραμματίσουμε σχεδόν κάθε μικροελεγκτή, με την προϋπόθεση ότι έχουμε έναν compiler, στην περίπτωση μας τον GNU GCC, ο οποίος μπορεί να μεταφράσει τον κώδικα C σε assembly για τον ελεγκτή μας.

1.4 Zigbee

Το ZigBee βασίζεται στο πρότυπο μετάδοσης IEEE 802.15.4. Λειτουργεί στις ελεύθερες ζώνες συχνοτήτων 2,4 GHz, 915 MHz και 868 MHz. Ο ρυθμός μεταφοράς δεδομένων είναι 250 kbps, 40 kbps και 20 kbps για κάθε ζώνη συχνοτήτων αντίστοιχα. Λόγω του χαμηλού ρυθμού μετάδοσης των δεδομένων δεν προορίζεται για υψηλής ποιότητας μετάδοση φωνής ή εικόνας, άλλα ενδείκνυται περισσότερο για σήματα με χαμηλότερη ποσότητα πληροφορίας όπως για παράδειγμα οι μετρήσεις ενός αισθητηρίου. Για την αποφυγή διενέξεων εντός του ασύρματου δικτύου, το ZigBee διαθέτει έλεγχο πρόσβασης CSMA-CA.

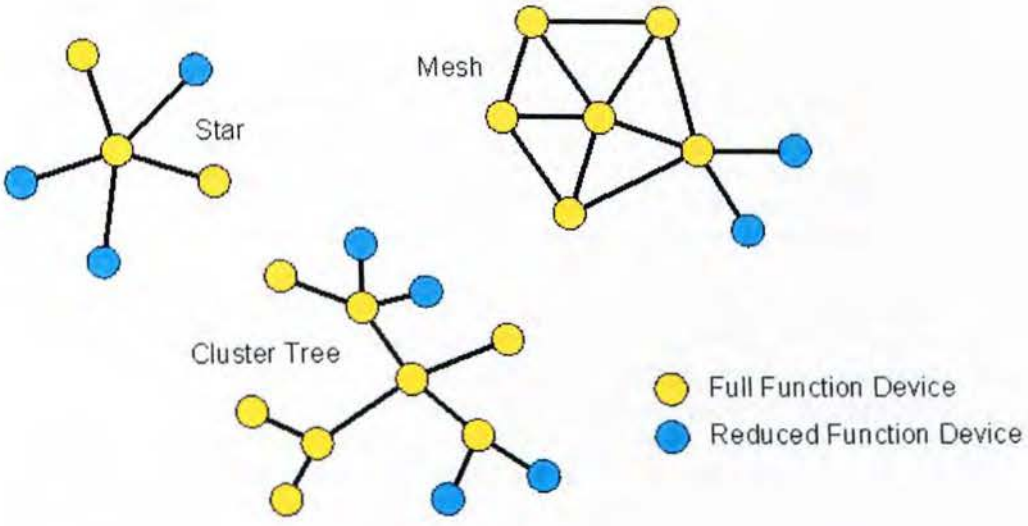
Το ZigBee είναι παρόμοιο με το Bluetooth και το WPAN, ένα ασύρματο προσωπικό δίκτυο (Wireless Personal Area Network). Το δίκτυο είναι πάντοτε διαθέσιμο για το χρήστη. Ωστόσο, το ZigBee έχει χαμηλότερους ρυθμούς μεταφοράς δεδομένων από ότι το Bluetooth.

Οι συσκευές ZigBee είναι οικονομικότερες από τις μονάδες Bluetooth. Τις περισσότερες φορές μεταδίδουν μόνο μικρούς όγκους δεδομένων και καταναλώνουν μικρές ποσότητες ενέργειας.

Δικτύωση

Οι συσκευές ZigBee διακρίνονται σε συσκευές με πλήρεις λειτουργίες FFD (Full Function Device) και σε συσκευές με περιορισμένες λειτουργίες RFD (Reduced Function Device). Τα στοιχεία που έχουμε σε ένα ZigBee δίκτυο είναι ο κεντρικός διαχειριστής (PAN Coordinator), ο απλός διαχειριστής ή δρομολογητής (Router) και η τερματική συσκευή (End Device). Τα PAN Coordinator και Router θα πρέπει να είναι οπωσδήποτε FFD ενώ το End Device μπορεί να είναι οτιδήποτε (συνήθως RFD).

Οι τοπολογίες δικτύωσης είναι αστέρα, δέντρο και πλέγμα. Η τοπολογία αστέρα περιλαμβάνει ένα κεντρικό διαχειριστή και πλήθος στοιχείων τα οποία επικοινωνούν αποκλειστικά μαζί του. Στην τοπολογία δέντρου και πλέγματος τα FFD στοιχεία έχουν τη δυνατότητα να επικοινωνούν το ένα με το άλλο, ενώ τα RFD αλληλεπιδρούν μόνο με το κοντινότερο σε αυτά FFD στοιχείο.



1.4.1 *Τοπολογίες αστέρα (star), δέντρου(tree) και πλέγματος (mesh)*

Παρακάτω παραθέτουμε πίνακα με τα σημαντικότερα χαρακτηριστικά των κυριότερων τεσσάρων προτύπων ασύρματης επικοινωνίας

Πρότυπο	Συχνότητα Λειτουργίας (GHz)	Ρυθμός Μετάδοσης δεδομένων (Mbps)	Τυπική Εμβέλεια (m)	Κατανάλωση Ισχύος	Τύπος Δικτύου
Wi – Fi	2.4 , 5	11 - 54	100	Υψηλή	WLAN
WiMAX	2-11 , 10-66	έως 72	50000	Υψηλή	WMAN
Bluetooth	2.4	1 - 3	10	Χαμηλή	WPAN-MR
ZigBee	0.868 , 0.915 2.4	0.02 , 0.04, 0.25	100	Πολύ χαμηλή	WPAN-LR

1.4.2 *Πίνακας χαρακτηριστικών ασύρματων δικτύων*

Λόγο της χαμηλής κατανάλωσης ισχύος, του ρυθμού μετάδοσης δεδομένων, της εμβέλειας αλλά και του χαμηλού κόστους επιλέγουμε το πρότυπο ZigBee για την πτυχιακή μας εργασία.

1.5 Servo Motors

Για να γίνουν κατανοητοί οι λόγοι για τους οποίους επιλέξαμε servo motors σε αυτήν την εργασία θα κάνουμε μια σύντομη αναφορά στους άλλους δύο τύπους κινητήρων που θα μπορούσαμε να χρησιμοποιήσουμε, δηλαδή τους βηματικούς και τους DC κινητήρες.

Βηματικοί κινητήρες

Οι βηματικοί κινητήρες παρέχουν ακρίβεια θέσης χωρίς την χρήση συστήματος ανάδρασης θέσης. Η ιδιότητα αυτή κρατάει το κόστος της εφαρμογής, στην οποία χρησιμοποιούνται, χαμηλά. Επιπλέον η χρησιμοποίηση τους περιλαμβάνει περιπτώσεις κατά τις οποίες το περιβάλλον της εφαρμογής δυσκολεύει την χρήση αισθητήρων ή τις περιπτώσεις στις οποίες η ρύθμιση ενός σερβοκινητήρα είναι δύσκολη ή απρόσφορη. Οι βηματικοί κινητήρες δεν χρησιμοποιούν ψήκτρες και για αυτό οι ανάγκες συντήρησης είναι μειωμένες.

Ο βηματικός κινητήρας είναι ένας σύγχρονος κινητήρας εναλλασσόμενου ρεύματος, σχεδιασμένος να λειτουργεί χρησιμοποιώντας ψηφιακή διέγερση σε κάθε ένα από τα τυλίγματα του. Για να κρατήσει μια σταθερή θέση ο κινητήρας χρειάζονται σταθερές τάσεις. Η κίνηση ενός ασύγχρονου κινητήρα ακολουθεί την κίνηση της φάσης της ηλεκτρικής διέγερσης. Υπό κανονικές συνθήκες, ένας σύγχρονος βηματικός κινητήρας θα ακολουθεί μια ταχύτητα που θα καθορίζεται από την συχνότητα του οδηγητικού ηλεκτρικού σήματος. Ο όρος σύγχρονος κινητήρας συνήθως υποδηλώνει μια συνεχή, ομαλή κίνηση. Ο βηματικός κινητήρας προσπαθεί ,πάντα, να παρακολουθεί την φάση του ηλεκτρικού σήματος που τον οδηγεί, αλλά πλέον τα σήματα είναι ψηφιακά και μπορούν να διατηρούν την ίδια φάση για μεγάλο χρονικό διάστημα. Κατά την διάρκεια αυτών των διαστημάτων ο βηματικός κρατάει σταθερή θέση, αντιδρώντας με αντίθετη ροπή στις πιθανές ροπές που τείνουν να τον μετακινήσουν από τη θέση αυτή.

Όταν μεταβάλλεται το ηλεκτρικό σήμα που οδηγεί τον κινητήρα και κάνει ένα απότομο άλμα σε μια νέα γωνία φάσης, ο κινητήρας προσπαθεί να ακολουθήσει. Ο κινητήρας θα μετακινηθεί σε αυτήν την νέα θέση μέχρι να αλλάξει το σήμα διέγερσης. Εάν η περίοδος της μεταβολής της διέγερσης κρατηθεί σταθερή, ο κινητήρας θα κινείται με σταθερή μέση ταχύτητα. Παρόλα αυτά η ταχύτητα κατά την διάρκεια ενός βήματος θα μπορούσε να αλλάζει σημαντικά κατά συνέπεια είναι δύσκολο να λειτουργήσει ένας βηματικός κινητήρας σε κατάσταση σταθερής ταχύτητας αλλά μπορεί να λειτουργεί σε κατάσταση σταθερού ρυθμού βημάτων. Επιπλέον μειονεκτήματα είναι οι ταλαντώσεις που προκαλούνται από την σπασμωδική κίνηση του κινητήρα , η μεγάλη κατανάλωση ρεύματος και οι υψηλές θερμοκρασίες. Επίσης αν το φυσικό σύστημα έχει μικρή απόσβεση , μπορεί να υπάρχουν κρίσιμες ταχύτητες στις οποίες δεν μπορεί να λειτουργεί ο κινητήρας για να μην προκληθεί ζημιά στο σύστημα.

DC κινητήρες

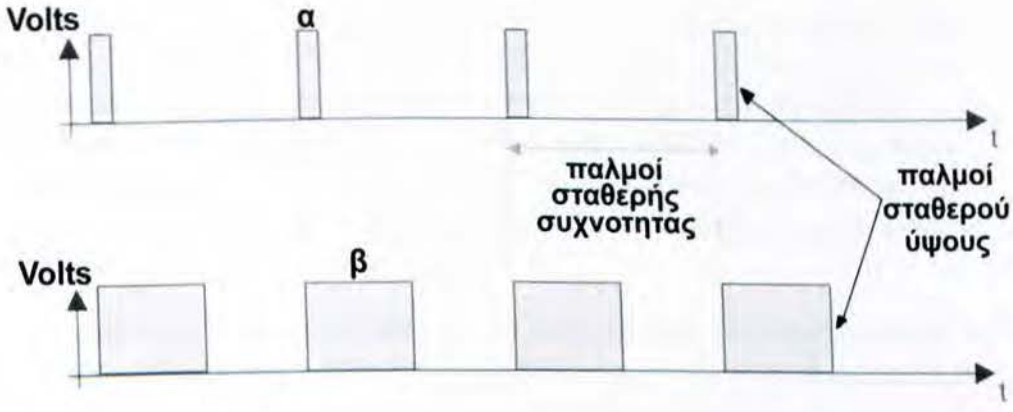
Οι κινητήρες συνεχούς ρεύματος είναι αρκετά απλοί. Όταν εφαρμοστεί τάση στον κινητήρα, αυτός αρχίζει να περιστρέφεται. Όσο μεγαλύτερη η τάση τόσο γρηγορότερα περιστρέφεται. Οι περισσότεροι κινητήρες συνεχούς ρεύματος είναι αρκετά γρήγοροι φτάνοντας μέχρι τις 5000 στροφές ανά λεπτό (RPM). Η ιδιότητα αυτή έκανε τους dc κινητήρες ιδιαίτερα δημοφιλής αλλά όταν κυριάρχησε η μέθοδος διανομής ηλεκτρικής ισχύος μέσω

εναλλασσόμενου ρεύματος, η εφαρμογή των dc κινητήρων μειώθηκε. Σε αρκετές εφαρμογές αντικαταστήθηκε, με επιτυχία, από τους βηματικούς κινητήρες.

Servo κινητήρες

Ο σερβοκινητήρας είναι ηλεκτρικός κινητήρας συνεχούς ρεύματος εφοδιασμένος με αισθητήρα προσδιορισμού της θέσης του άξονα περιστροφής του. Ο σερβοκινητήρας συνήθως συνδυάζεται με κιβώτιο υποβιβασμού της σχέσης μετάδοσης της κίνησης. Ο σερβοκινητήρας χρησιμοποιεί την μέθοδο ελέγχου κλειστού βρόχου για να βρεθεί στην επιθυμητή κατάσταση (θέση, ταχύτητα κ.α.).
Εφαρμόζοντας τάση λειτουργίας στους σερβοκινητήρες αυτοί περιστρέφονται με ταχύτητα η οποία καθορίζεται από τις προδιαγραφές τους. Η τεχνική του PWM (Pulse Width Modulation) χρησιμοποιείται για να μεταβάλλουμε την ταχύτητα περιστροφής του σερβοκινητήρα. Ακολουθώντας αυτόν τον τρόπο ο σέρβο οδηγείται με παλμούς σταθερής συχνότητας και ύψους και όχι με σταθερή τάση. Η διάρκεια των παλμών καθορίζει την ταχύτητα περιστροφής του κινητήρα.

Στο παρακάτω σχήμα βλέπουμε δύο σήματα οδήγησης του σερβοκινητήρα μας, με διαφορετικό πλάτος παλμού το καθένα.



1.5.1 *Ο κινητήρας περιστρέφεται με ταχύτητα U στην περίπτωση α.*
*Ο κινητήρας περιστρέφεται με ταχύτητα 5U στην περίπτωσης β, εάν β=5*α*

Ο εύκολος έλεγχος του σερβοκινητήρα από τον μικροελεγκτή, ο έλεγχος βρόχου με ανάδραση που διαθέτει, η αξιοπιστία, η ομαλή λειτουργία του και ο ακριβής έλεγχος της ταχύτητας περιστροφής του, καθιστούν τον σερβοκινητήρα ιδανική περίπτωση για την εργασία μας.

ΚΕΦΑΛΑΙΟ 2^{ο}
Αναγνώριση Χειρού

2.1 Περιγραφή

Σε αυτό το κεφάλαιο θα ασχοληθούμε με το πρόβλημα της αναγνώρισης χειρονομιών. Απαντάμε σε ερωτήσεις όπως: ¨Τι είναι η υπολογιστική όραση; ¨ . Παρουσιάζουμε τα βήματα που ακολουθούμε για την αντιμετώπιση του προβλήματος[1] όπως, σύλληψη εικόνας, επεξεργασία, χωρισμός εικόνας σε τμήματα, ανίχνευση περιγράμματος του επιθυμητού αντικειμένου, εξαγωγή χαρακτηριστικών, αναγνώριση και ερμηνεία χειρονομιών. Όλα τα παραπάνω έχουν σκοπό το καλύτερο δυνατό αποτέλεσμα της εφαρμογής μας.

2.2 Υπολογιστική όραση

Η υπολογιστική όραση είναι ένας δυναμικός τομέας της επιστήμης των υπολογιστών, με σκοπό την δημιουργία συστημάτων για την ανάκτηση πληροφοριών από εικόνες που συλλαμβάνουν κάμερες. Τέτοιες πληροφορίες μπορούν να οδηγούν το σύστημα μας να λαμβάνει αποφάσεις ή να αναπαριστά διαφορετικά τις εικόνες. Η λήψη αποφάσεων αναφέρετε στην οπτική αναγνώριση, πώς να γνωρίζουμε ,δηλαδή, τι είναι κάποιο αντικείμενο σε μια εικόνα- λειτουργία αυτονόητη για το ανθρώπινο οπτικό σύστημα. Η καινούργια παρουσίαση της εικόνας στοχεύει στην επεξεργασία και στην αλλαγή της πραγματικής εικόνας που μας παρέχει η κάμερα, εργασία απαραίτητη για την ανάπτυξη πληθώρας εφαρμογών της υπολογιστικής όρασης.

Πριν συνεχίσουμε στο θέμα μας, μπορούμε να κάνουμε μια μικρή αναφορά στον τρόπο με τον οποίο αντιλαμβάνεται ο ανθρώπινος εγκέφαλος ένα οπτικό σήμα. Ο εγκέφαλος μας διαχωρίζει ένα οπτικό σήμα σε πολλά κανάλια τα οποία παρέχουν διαφορετικού είδους πληροφορίες σε αυτόν. Ο εγκέφαλος χρησιμοποιεί μηχανισμούς οι οποίοι αναγνωρίζουν τα τμήματα της εικόνας που θεωρούνται σημαντικά και πρέπει να εξεταστούν και παραβλέπει άλλα τμήματα της. Η τελική εικόνα που βλέπει ο άνθρωπος δεν εξαρτάται από την πραγματική εικόνα αλλά από την προσωπική του εμπειρία, τι έχει μάθει στο παρελθόν, από την φυσική δομή και κατάσταση των ματιών και του εγκεφάλου. Κύρια αισθητήρια εισόδου μπορούν να θεωρηθούν τα μάτια , τα οποία ελέγχουν μηχανικά την φωτεινότητα αλλά και άλλες αισθήσεις για να σχηματίσουμε την τελική εικόνα.

Σε αντίθεση με το ανθρώπινο οπτικό σύστημα, στο σύστημα μηχανικής όρασης ο υπολογιστής το μόνο που λαμβάνει είναι πλέγμα από αριθμούς. Η δικιά μας δουλειά είναι να μετατρέψουμε αυτό το πλέγμα αριθμών σε αντίληψη. Στην παρακάτω εικόνα μπορούμε να δούμε τι ακριβώς ¨βλέπει¨ ένας υπολογιστής.

---

[1] F.Perales and M. Absolo , Vision por Ordenador

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

2.2.1 *Για τον υπολογιστή, ο πλαϊνός καθρέφτης του αμαξιού είναι ένα πλέγμα αριθμών*

Η μετατροπή αυτού του πλέγματος αριθμών σε ¨πλαϊνό καθρέπτη¨ δεν είναι εύκολο να γίνει, για την ακρίβεια, ίσως είναι αδύνατον. Πρόκειται για μια 2D εικόνα ενός τρισδιάστατου κόσμου και δεν υπάρχει τρόπος να κατασκευάσουμε το τρισδιάστατο σήμα. Αυτή η δυσδιάστατη εικόνα μπορεί να αντιστοιχεί σε οποιαδήποτε τρισδιάστατη σκηνή.

Ακόμη ένα πρόβλημα το οποίο πρέπει να αντιμετωπισθεί είναι ο θόρυβος στα δεδομένα που σχηματίζουν την εικόνα. Η διάβρωση της πληροφορίας μας μπορεί να προέρχεται από διάφορα αίτια. Ο καιρός , η φωτεινότητα, η αντανακλάσεις, οι κινήσεις είναι μερικές από τις ατέλειες που οφείλονται σε μεταβολές του περιβάλλοντος. Επιπλέον προβλήματα μηχανικής φύσεως που πρέπει να αντιμετωπίσουμε αφορούν, ελαττώματα στον φακό της κάμερας, λανθασμένες ρυθμίσεις, ηλεκτρικούς ή ηλεκτρονικούς θορύβους στα αισθητήρια της κάμερας, καθώς και η θαμπάδα κίνησης (motion blur) και τα σφάλματα συμπίεσης της ληφθείσας εικόνας.

Όλα τα παραπάνω αποτελούν προκλήσεις που πρέπει να αντιμετωπισθούν όταν γίνεται ανάπτυξη εφαρμογών της υπολογιστικής όρασης. Ο σαφής καθορισμός του θέματος της εφαρμογής μας καθώς και η απομάκρυνση του θορύβου διαδραματίζουν καθοριστικό ρόλο στο τελικό αποτέλεσμα.

2.3 Επεξεργασία εικόνας

Απαραίτητο για να επεξεργαστούμε μια εικόνα είναι να συλλάβουμε μια εικόνα. Αυτό το καταφέρνουμε χρησιμοποιώντας την Logitech C270 webcam. Η κάμερα μας λαμβάνει τις αναλογικές εικόνες σαν μια διαδοχή εικόνων (βίντεο) και τα ηλεκτρονικά κυκλώματα frame grabbers την μετατρέπουν σε ψηφιακή μορφή. Την ψηφιακή εικόνα που λαμβάνουμε την επεξεργαζόμαστε, δηλαδή χρησιμοποιούμε τεχνικές ανάλυσης, ενίσχυσης, συμπίεσης και ανασυγκρότησης. Η επεξεργασία γίνεται με την βοήθεια της βιβλιοθήκης της OpenCV. Στη συνέχεια γίνεται λεπτομερής αναφορά στις συναρτήσεις που χειριζόμαστε για να επεξεργαστούμε το αντικείμενο μας.

Ψηφιακή Εικόνα

Η ασπρόμαυρη εικόνα αποτελεί μια δισδιάστατη συνάρτηση του χώρου x και y. Περιγράφεται μαθηματικά ως f(x,y). Αυτό δείχνει ότι στα σημεία της επιφάνειας της εικόνας που είναι άσπρα, η τιμή της f είναι μεγάλη, ενώ στα σημεία της επιφάνειας της εικόνας που είναι μαύρα, η τιμή της f είναι μικρή. Δηλαδή η f αντιστοιχεί στην αμαύρωση της εικόνας σε κάθε θέση (x,y).

Για να έχουμε ψηφιακή επεξεργασία εικόνας πρέπει να μετατρέψουμε την αναλογική εικόνα σε ψηφιακή. Προκειμένου να πετύχουμε κάτι τέτοιο πρέπει να λάβουμε αρκετά πυκνά δείγματα ώστε να προλάβουμε τις εναλλαγές στην αμαύρωση της εικόνας. Μετά τα δείγματα πρέπει να κβαντιστούν σε πεπερασμένο αριθμό σταθμών. Συνήθως χρησιμοποιούνται 256 στάθμες, στη στάθμη 0 αντιστοιχεί το μαύρο ενώ στη στάθμη 255 το άσπρο. Με αυτόν τον τρόπο ένα byte είναι ικανό να περιγράψει την τιμή της αμαύρωσης ενός pixel. Η ψηφιακή εικόνα παριστάνεται ως η κβαντισμένη σε πλάτος συνάρτηση fq(n1,n2).

Οι n1 και n2 αποτελούν διακριτές χωρικές μεταβλητές που αντιστοιχούν στις συνεχείς χωρικές μεταβλητές x και y. Ο υπολογιστής αντιλαμβάνεται μια εικόνα ως ένα πλέγμα αριθμών διαστάσεων MxN (M μέγιστη τιμή n1 και N μέγιστη τιμή n2) , καθένας από τους οποίους εκπροσωπεί ένα εικονοστοιχείο (pixel). Το αποτέλεσμα του πολλαπλασιασμού M*N είναι ο αριθμός των pixel μιας εικόνας. Όσα περισσότερα pixels τόσο πιο ευκρινής είναι η εικόνα. Η ευκρίνεια συνήθως μετριέται σε πλήθος εικονοστοιχείων ανά μονάδα επιφάνειας και εξαρτάται από τις διαστάσεις και το αριθμό των αποχρώσεων μιας εικόνας. Η πιο απλή μορφή εικόνας, που η πληροφορία παρουσιάζεται μόνο από ένα bit, είναι η δυαδική. Σε αυτή τη μορφή το 0 παριστάνει το μαύρο και το 1 το άσπρο, κάνοντας την απόκτηση πληροφοριών απλό γεγονός. Επίσης ο υπολογιστικός χρόνος είναι χαμηλός και λαμβάνονται πληροφορίες όπως το εμβαδόν και η θέση αντικειμένων. Συχνά είναι μορφή στην οποία επιδιώκουμε να μετατρέψουμε άλλες μορφές εικόνων, όπως τις έγχρωμες. Στις έγχρωμες εικόνες σε κάθε θέση της (n1,n2) η f έχει τρείς τιμές, οι οποίες αντιστοιχούν στις τιμές των χρωμάτων του χρωματικού χώρου που χρησιμοποιούμε.

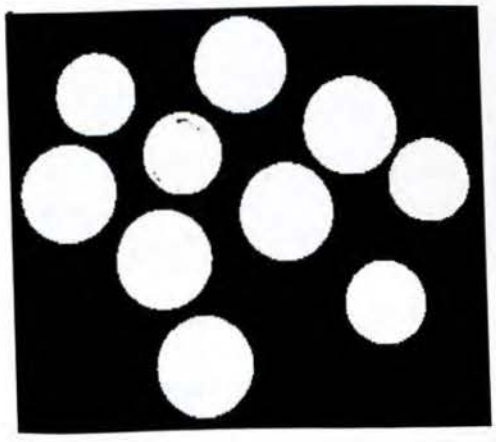
Επιπλέον, σε περίπτωση ακολουθίας εικόνων (βίντεο) η συνάρτηση f έχει την μεταβλητή του χρόνου, t. Κάθε εικόνα απέχει από την άλλη μικρό χρονικό διάστημα και περιγράφεται από την συνάρτηση f(n1,n2,n3) , διακριτή και ως προς το χρόνο (t->n3).

Κατωφλίωση (Thresholding)

Ένα κατώφλι όταν εφαρμόζεται σε μια εικόνα μπορεί να διαχωρίζει αντικείμενα από το φόντο(background) και το προσκήνιο(foreground). Συνήθως πρέπει η εικόνα να είναι σε κλίμακα του γκρι(grayscale) για να έχουμε ικανοποιητικά αποτελέσματα στην εφαρμογή της μεθόδου κατωφλίωσης. Σε μια grayscaled εικόνα οι τιμές κυμαίνονται μεταξύ του μηδέν και του ένα, γεγονός που απλουστεύει την διαδικασία κατωφλίωσης. Παρακάτω παρουσιάζουμε μια grayscaled εικόνα και το ιστόγραμμα της. Το ιστόγραμμα είναι η γραφική αναπαράσταση της εικόνας, ο οριζόντιος άξονας αναφέρεται στη φωτεινότητα και ο κατακόρυφος στο πλήθος τω εικονοστοιχείων.

2.3.1 *Grayscaled εικόνα και το ιστόγραμμα που αντιστοιχεί*

Το ιστόγραμμα εμφανίζει τις τιμές του φόντου και τις τιμές του προσκηνίου, οι οποίες παριστάνονται ως δύο διαφορετικές κορυφές. Η φωτεινότητα είναι ίση με 255 και ο συνολικός αριθμός τω pixel είναι 196608. Εάν ορίσουμε τιμή κατωφλίωσης το T = 90, το σημείο ανάμεσα στις δύο κορυφές, το αποτέλεσμα είναι η παρακάτω δυαδική εικόνα.



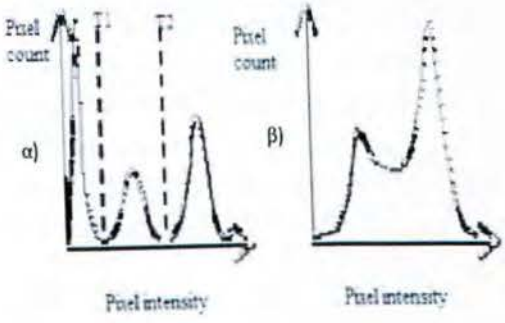
Ισχύει: φόντο, εάν $f(x,y) \leq T$ αλλιώς προσκήνιο

Το φόντο έχει χρώμα μαύρο (0) ενώ το προσκήνιο έχει χρώμα άσπρο (1).

2.3.2 *Αποτέλεσμα Thresholding*

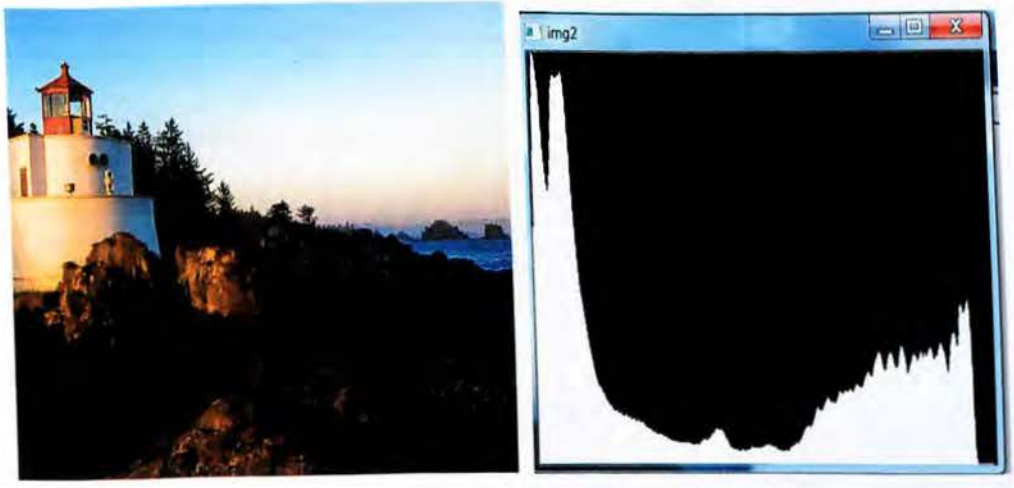
Η μέθοδος κατωφλίωσης που περιγράψαμε είναι η απλούστερη περίπτωση, όπου τα όρια των κορυφών είναι ευδιάκριτα. Ονομάζεται μέθοδος του απόλυτου κατωφλίου.

Συνήθως οι εικόνες που πρέπει να επεξεργαστούμε δεν έχουν το ιστόγραμμα της μορφής που είδαμε παραπάνω και ο καθορισμός του thresholding με την μέθοδο του απόλυτου κατωφλίου δεν δίνει ικανοποιητικά αποτελέσματα. Για τον λόγο αυτό χρησιμοποιούμε άλλες τεχνικές κατωφλίωσης. Στο σχήμα α της παρακάτω εικόνας υποθέτουμε ότι τα pixel του foreground βρίσκονται στη μεσαία κορυφή, άρα για να τα εμφανίσουμε πρέπει να εφαρμόσουμε δύο τιμές κατωφλίωσης, την T1 και την T2, και ισχύει: pixel προσκηνίου εάν $T1 < f(x,y) \leq T2$. Στη β περίπτωση οι κορυφές επικαλύπτονται και είναι σχεδόν αδύνατον να πετύχει η μέθοδος κατωφλίου.



2.3.3 *Ιστογράμματα δύο διαφορετικών εικόνων*

Παρακάτω παρουσιάζουμε ένα παράδειγμα που παρατηρείται το φαινόμενο της επικάλυψης. Παρατηρούμε ότι εξαιτίας της φωτεινότητας, τα εικονοστοιχεία του background και του foreground δεν διακρίνονται ευκρινώς.



2.3.4 *Ιστόγραμμα εικόνας με ανομοιογενή φωτισμό*

Το αποτέλεσμα της ανομοιογένειας του φωτισμού είναι η δυαδική εικόνα που λαμβάνουμε –από οποιαδήποτε μέθοδο κατωφλίου- να μην είναι ικανοποιητική.

Ο τρόπος με τον οποίο πραγματοποιούμε την κατωφλίωση στην εργασία μας είναι ο εξής: Επιλέγουμε το χρωματικό χώρο στον οποίο θα δουλέψουμε. Οι χρωματικοί χώροι έχουν

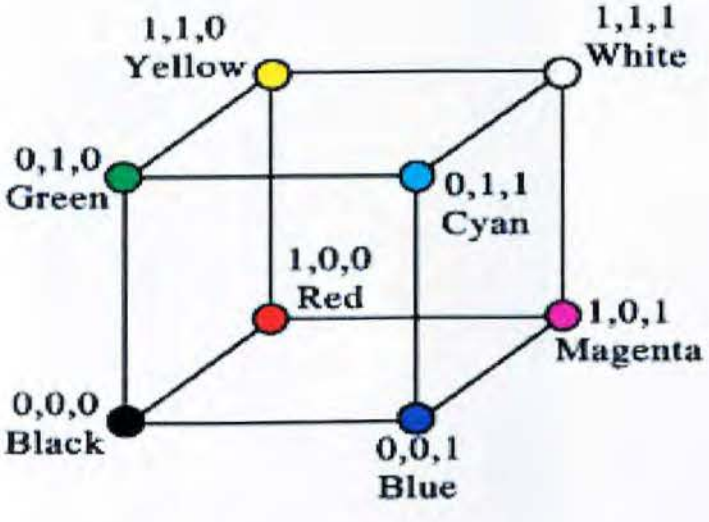
τρία κανάλια χρωμάτων, για τον λόγο αυτό προσπαθούμε να εφαρμόσουμε την τεχνική της κατωφλίωσης σε κάθε κανάλι ξεχωριστά. Οι τιμές κατωφλίωσης είναι δύο, η ανώτερη και η κατώτερη, τα εικονοστοιχεία που βρίσκονται μεταξύ των δύο τιμών παρουσιάζονται ως άσπρο χρώμα στη τελική δυαδική εικόνα, ενώ τα υπόλοιπα pixel σαν μαύρο χρώμα. Η τελική εικόνα είναι δυαδική και περιέχει το αποτέλεσμα της κατωφλίωσης και των τριών καναλιών.

Χρωματικός Χώρος (Color space)

Οι τεχνικές για την ανάπτυξη αλγορίθμων εντοπισμού του χρώματος του δέρματος χρησιμοποιούνται ευρέως για την αναγνώριση μερών του ανθρώπινου σώματος.
Ο χρωματικός χώρος στον οποίο θα αναπτύξουμε τον αλγόριθμο αναγνώρισης είναι σημαντικός για να πετύχουμε τα επιθυμητά αποτελέσματα. Ο αλγόριθμος αναγνώρισης του χεριού πρέπει να ξεχωρίζει τα pixels που έχουν το χρώμα του χεριού από τα υπόλοιπα pixels της εικόνας.
Παρουσιάζουμε τα color spaces RGB, YCrCb , συγκρίνοντας τα αποτελέσματα της αναγνώρισης σε καθένα από τα παραπάνω color spaces.

- **RGB**

Το RGB είναι χρωματικό πρότυπο που συνδυάζει τρία χρώματα -κόκκινο, πράσινο, μπλε- για να αναπαράγει τα υπόλοιπα. Επειδή χρησιμοποιούνται τρία χαρακτηριστικά χρώματα, το RGB μοντέλο παρουσιάζεται ως κύβος με εύρος τιμών από 0 εως 1.



2.3.5 *κύβος RGB μοντέλου*

Ο αριθμός των pixels που χρειάζεται για να αναπαραχθεί μια εικόνα στο RGB μοντέλο ονομάζεται βάθος pixel (pixel depth) και μπορεί να είναι 8-bit ανά κανάλι χρώματος (255,0,0) ,δηλαδή μπορεί να αναπαριστά πάνω από 16 εκατομμύρια χρώματα.

- **Αναγνώριση δέρματος στο RGB μοντέλο**

Παρακάτω βλέπουμε τα αποτελέσματα του αλγόριθμου αναγνώρισης του χρώματος του δέρματος στο μοντέλο RGB. Ο τρόπος που δουλεύει ο αλγόριθμος είναι απλός, καθορίζει αν κάποιο pixel ανήκει στο χρώμα του δέρματος. Αυτό το πετυχαίνει ακλουθώντας κανόνες οι οποίοι έχουν καθοριστεί εμπειρικά[2]. Οι τιμές των συστατικών εικόνων RGB προέρχονται από αυτές τις εμπειρικές παρατηρήσεις και το εύρος του κυμαίνεται από:

R [95,221]
G [40,169]
B [20,139]
οπου RGB[0,255].

2.3.2 *Αρχική rgb εικόνα*

2.3.6 *εμφάνιση σε δυαδική μορφή της αναγνώρισης δέρματος rgb εικόνας*

[2] Journal of Multimedia, VOL. 6, NO. 2

YCrCb

Το YCrCb πρότυπο είναι ένα μη γραμμικό, κωδικοποιημένο RGB σήμα το οποίο χρησιμεύει στη συμπίεση εικόνας. Δηλαδή είναι ψηφιακό μοντέλο χρώματος και έχει την δυνατότητα να πραγματοποιήσει ψηφιακή επεξεργασία βίντεο. Το στοιχείο Y αναφέρεται στην φωτεινότητα (Luminance) ενώ τα στοιχεία Cr και Cb στις πληροφορίες χρώματος (Chrominance). Το Cr είναι η χρωματική διαφορά ως προς το κόκκινο και το Cb είναι η χρωματική διαφορά ως προς το μπλε.

$$Y = 0.299R+0.578G+0.114B$$ [3]

$$Cr=R-Y$$

$$Cb=B-Y$$

Στην παρακάτω εικόνα παρατηρούμε το στοιχειό της φωτεινότητας Y μετά την μετατροπή.

2.3.7 Το σήμα φωτεινότητας Y (δεξιά)

Μετασχηματίζοντας το RGB σήμα σε ένα σήμα φωτεινότητας και σε δύο σήματα τα οποία μεταφέρουν όλες τις πληροφορίες για το χρώμα, μειώνουμε τον χρόνο μετάδοσης της εικόνας. Επίσης μειώνεται και το εύρος ζώνης καθώς η ακρίβεια μετάδοσης και η παράσταση των χρωματικών σημάτων είναι μικρότερη.

[3] Color From Grey by Optimized Color Ordering, Simon Frazer University

Η μετατροπή του RGB χρωματικού χώρου στον YCbCr γίνεται βάσει της παρακάτω σχέσης:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

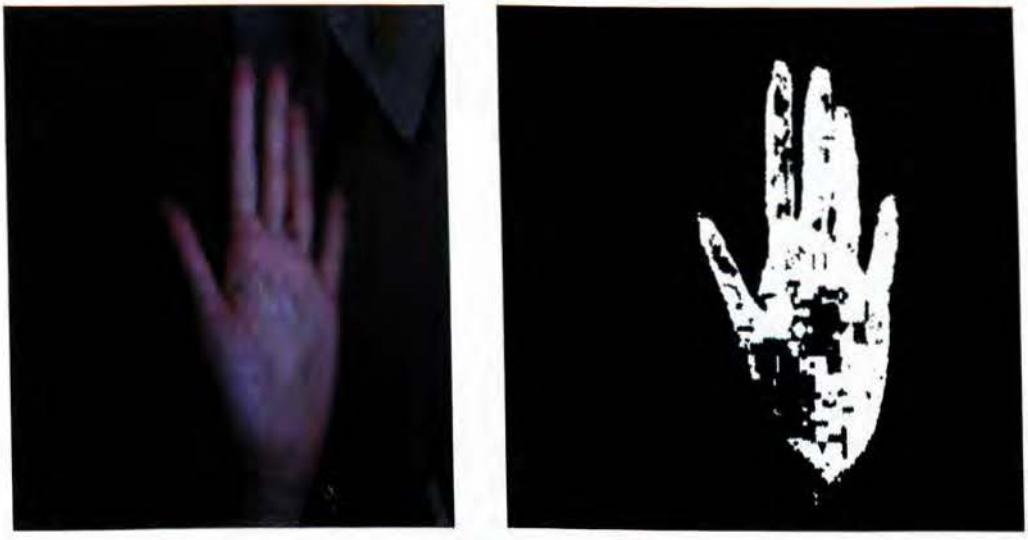
Στην παραπάνω σχέση οι RGB τιμές θεωρείται ότι βρίσκονται στο διάστημα [0,1]. Με τις προϋποθέσεις αυτές η μεταβλητή Y κυμαίνεται στο διάστημα [16,235] και οι Cb, Cr στο διάστημα [16,240].

- **Αναγνώριση δέρματος στο YCrCb μοντέλο**

Παρακάτω βλέπουμε τα αποτελέσματα του αλγόριθμου αναγνώρισης του χρώματος του δέρματος στο μοντέλο YCrCb. Για να το πετύχουμε αυτό πρέπει να μετατρέψουμε το RGB χώρο χρώματος που βρισκόμαστε σε χώρο YCrCb. Αυτό γίνεται με την χρήση της συνάρτησης της OpenCV, η οποία είναι η cvCvtColor (src,dst,colorspace). Εφόσον βρισκόμαστε –πλέον- στο καινούργιο color space καθορίζουμε αν κάποιο pixel ανήκει στο χρώμα του δέρματος. Αυτό το πετυχαίνουμε ακλουθώντας κανόνες οι οποίοι έχουν καθοριστεί εμπειρικά[4].
Οι τιμές των συστατικών εικόνων YCrCb προέρχονται από αυτές τις εμπειρικές παρατηρήσεις και το εύρος του κυμαίνεται από:

Y [60,250]
Cr [135,170]
Cb [90,135]
οπου YCrCb [0,255].

[4] Journal of Multimedia, VOL. 6, NO. 2



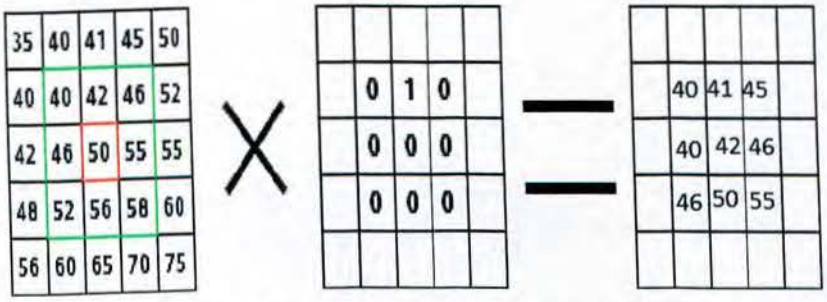
2.3.8 Αποτέλεσμα (δεξιά) αναγνώρισης χρώματος του δέρματος

Συμπεράσματα: Έχουμε την ίδια αρχική εικόνα και παρατηρούμε το αποτέλεσμα που έχει ο αλγόριθμος αναγνώρισης στα δύο διαφορετικά color spaces που παρουσιάσαμε.
Τα αποτελέσματα του αλγορίθμου στον RGB χρωματικό χώρο είναι άμεσα, καθώς ολόκληρη η πληροφορία για το χρώμα παρέχεται από την μορφή της εικόνας και δεν είναι αναγκαίο να μεταβούμε σε άλλο χρωματικό χώρο.
Το πρόβλημα που παρατηρούμε είναι η μειωμένη απόδοση του αλγορίθμου μας στις συνθήκες φωτεινότητας της αρχικής εικόνας, γεγονός το οποίο δεν καθιστά ικανή την χρήση του RGB χρωματικού χώρου στη συνέχεια της εργασίας μας.
Ο αλγόριθμος δείχνει να αποδίδει καλύτερα στο χρωματικό χώρο YCrCb. Τα αποτελέσματα σε εφαρμογές πραγματικού χρόνου (βίντεο) είναι ικανοποιητικά. Ο διαχωρισμός των σημάτων φωτεινότητας από τα σήματα που περιέχουν τις χρωματικές πληροφορίες δίνει σταθερότητα στα αποτελέσματα σε διαφορετικές συνθήκες φωτεινότητας.
Λαμβάνοντας υπόψη τα παραπάνω αναπτύσσουμε τον αλγόριθμο μας σε YCrCb χρωματικό χώρο.

Συνέλιξη

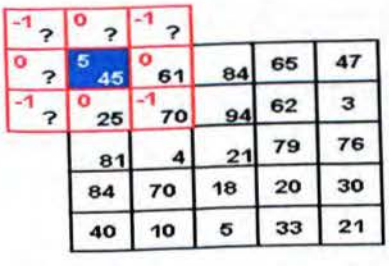
Η έννοια της συνέλιξης(convolution) συναντάται συχνά στην επεξεργασία σήματος και εικόνας. Στην ανάλυση εικόνας μας ενδιαφέρει, κυρίως, η δισδιάστατη συνέλιξη. Χαρακτηριστικό της είναι ο πυρήνας συνέλιξης (Kernel), ο οποίος εφαρμόζεται σε μια kxk περιοχή της εικόνας και προσδιορίζει την τιμή του κεντρικού pixel της περιοχής. Ο πυρήνας εφαρμόζεται σε κάθε ένα pixel της εικόνας. Μεταβάλλοντας το μέγεθος του πυρήνα μπορούμε να εκτελέσουμε διαφορετικές διεργασίες, όπως να τονίσουμε ή να

εξομαλύνουμε τα χαρακτηριστικά μιας εικόνας.
Οι πυρήνες στη πλειοψηφία τους είναι τετραγωνικοί πίνακες, kx=ky=k, όπου k είναι ένας μονός αριθμός. Η γενική μορφή της δισδιάστατης διακριτής συνέλιξης μπορεί να αποδοθεί από τον τύπο: $\mathrm{Out}(i,j)=\sum_{m=0}^{M-1}\sum_{n-1}^{N-1}\mathrm{In}(m,n)\mathrm{Mask}(i-m,j-n)$, όπου In είναι η εικόνα εισόδου, Mask η μάσκα συνέλιξης, και Out η εικόνα εξόδου. Η εικόνα έχει μέγεθος NxM και η μάσκα είναι σημαντικά μικρότερη από την εικόνα NxM. Ένα τυπικό μέγεθος της μάσκας είναι 3x3, αλλά μπορούμε να χρησιμοποιήσουμε μεγαλύτερες μάσκες συνέλιξης εάν το επιθυμούμε.
Κατά την εφαρμογή της συνέλιξης ακολουθούνται τρία στάδια. Στο πρώτο στάδιο, ο πυρήνας συνέλιξης τοποθετείται στην εικόνα με τρόπο ώστε το κεντρικό pixel του πύρινα να αντιστοιχεί στη θέση κάθε pixel της εικόνας. Στο δεύτερο στάδιο, οι τιμές του πυρήνα πολλαπλασιάζονται με τις τιμές των pixel που καλύπτουν. Στο τελικό στάδιο αντικαθίσταται η τιμή του pixel της εικόνας με το άθροισμα των τιμών του δεύτερου σταδίου. Παρακάτω βλέπουμε ένα παράδειγμα της εφαρμογής της συνέλιξης. Αριστερά είναι οι τιμές των pixels της εικόνας και δεξιά ο 3x3 πυρήνας συνέλιξης. Το πράσινο χρώμα στην εικόνα αντιστοιχεί στο πυρήνα ο οποίος εφαρμόζεται στην εικόνα και το κόκκινο είναι το κεντρικό pixel. Πολλαπλασιάζουμε κάθε τιμή των pixels της εικόνας με τις τιμές του πυρήνα και μετά το άθροισμα τοποθετείται στο κεντρικό pixel. Ο πυρήνας εφαρμόζεται σε όλα τα pixels της εικόνας.



2.3.9 *Εφαρμογή πυρήνα συνέλιξης στα pixels μιας εικόνας*

Από τον πίνακα μπορούμε να καταλάβουμε ότι η τελική εικόνα θα είναι πιο ¨λεία¨ από την αρχική της μορφή. Αυτό γιατί το άθροισμα των στοιχείων του πίνακα συνέλιξης είναι μονάδα, εάν ήταν μηδέν τότε η τελική εικόνα θα έδινε έμφαση στη εύρεση των ακμών. Επίσης παρατηρούμε ότι δεν είναι ολοκληρωμένος ο πίνακας της τελικής εικόνας από στοιχεία, γιατί αλλοιώνονται τα pixel των ορίων της αρχικής εικόνας όταν εφαρμόζουμε τη μέθοδο της συνέλιξης. Το βαθμό αλλοίωσης το καθορίζει το μέγεθος του πίνακα. Το φαινόμενο αυτό ονομάζεται border effect και συμβαίνει όταν το κεντρικό pixel του πυρήνα επικαλύπτει κάποιο από τα pixel που βρίσκονται στα άκρα.



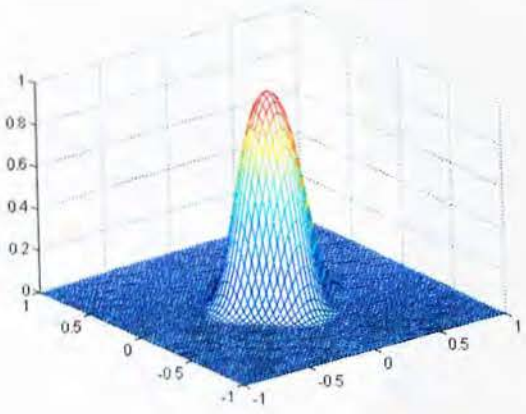
2.3.10 *φαινόμενο border effect*

- **Λείανση εικόνας**

Η λείανση (smoothing) είναι διαδικασία που χρησιμοποιείται συχνά στην επεξεργασία εικόνας. Πραγματοποιείται για την μείωση θορύβου και ελαττωμάτων της εικόνας. Είναι, επίσης, σημαντική όταν θέλουμε να μειώσουμε την ανάλυση της εικόνας με ομαλοποιημένο τρόπο. Η OpenCV μας προσφέρει πέντε διαφορετικές λειτουργίες λείανσης στη συνάρτηση cvSmooth. Από αυτές τις επιλογές εμείς θα εργαστούμε με την γκαουσιανή ομαλοποίηση (CV_GAUSSIAN).
Το φίλτρο Gauss είναι γραμμικό, βαθυπερατό και στις δύο διαστάσεις και ορίζεται με την συνάρτηση:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

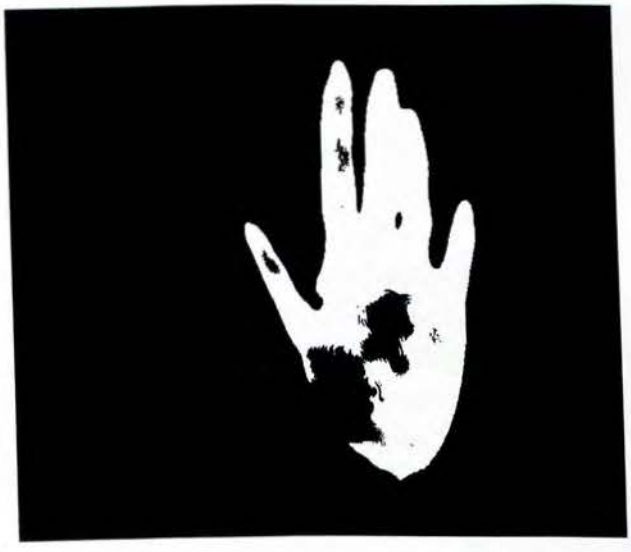
όπου, σ είναι η σταθερή απόκλιση της γκαουσιανής διανομής, x είναι η απόσταση από την αρχή του οριζόντιου άξονα και y η απόσταση από την αρχή του κάθετου άξονα. Το μεγαλύτερο βάρος της διανομής βρίσκεται στο κέντρο και υπάρχει ομαλή εξασθένηση όσο απομακρυνόμαστε από αυτό.



2.3.11 Gaussian διανομή

Στο φάσμα μιας εικόνας οι υψηλές συχνότητες αντιπροσωπεύουν τις λεπτομέρειες των αντικειμένων και οι χαμηλές συχνότητες απεικονίζουν μεγάλες περιοχές ενός αντικειμένου. Οι ακμές -το τμήμα της εικόνας που γίνεται ο διαχωρισμός των αντικειμένων- θα έχουν μετά την εφαρμογή του φίλτρου πιο κοντινές τιμές με τα γειτονικά τους pixels και έτσι θα είναι δυσδιάκριτες.
Παρακάτω παρατηρούμε τα αποτελέσματα του φίλτρου.

2.3.12 *Αρχική εικόνα*



2.3.13 Αποτέλεσμα φιλτραρίσματος (σ_x=5.45, σ_y=5.45)*

Όπως παρατηρούμε από τις παραπάνω εικόνες το low pass φίλτρο μειώνει την τιμή των ακμών των αντικειμένων που απεικονίζονται στην εικόνα, με αποτέλεσμα τη λείανση της.

* $\sigma_x=(n_x/2)*0.3+0.8$ n: μέγεθος παραθύρου του φίλτρου
$\sigma_y=(n_y/2)*0.3+0.8$

- **Διαστολή και Συστολή**

Η διαδικασίες συστολής και διαστολής μιας εικόνας πραγματοποιούνται από μορφολογικά φίλτρα[5]. Αυτά μοιάζουν αρκετά με τα φίλτρα μέγιστου- ελάχιστου. Στα φίλτρα μέγιστου – ελάχιστου γίνεται επιλογή του μικρότερου ή του μεγαλύτερου δείγματος το οποίο επιλέγεται από N- δείγματα μιας περιοχής (παράθυρο) του σήματος εισόδου. Το μέγεθος του φίλτρου αντιστοιχεί στο μέγεθος του παραθύρου.
Η διαφορά στα μορφολογικά φίλτρα έγκειται στο γεγονός ότι χρησιμοποιούν ένα δομικό στοιχείο (structural element), δηλαδή ένα σήμα συνάρτηση, μήκους όσο και το μήκος του παραθύρου. Οι τιμές του στοιχείου προστίθενται σημείο προς σημείο στις τιμές σήματος του παραθύρου. Οι μέγιστες τιμές που προκύπτουν αποτελούν την διαστολή (dilation) του σήματος f(n) με δομικό στοιχείο g(n) και εκφράζεται από τον παρακάτω τύπο:

$f(n)\oplus g(n)=\max\{f(m)+g(m-n)\}$
$m\in D$ D: είναι πεδίο ορισμού της f(n)
$m-n\in G$ G:είναι πεδίο ορισμού της g(n)

Οι ελάχιστες τιμές που προκύπτουν αποτελούν τη συστολή (erosion) του σήματος f(n) με δομικό στοιχείο g(n) και εκφράζεται από τη παρακάτω συνάρτηση:
$f(n)\ominus g(n)=\min\{f(m)-g(m-n)\}$
$m\in D$ D: είναι πεδίο ορισμού της f(n)
$m-n\in G$ G:είναι πεδίο ορισμού της g(n)

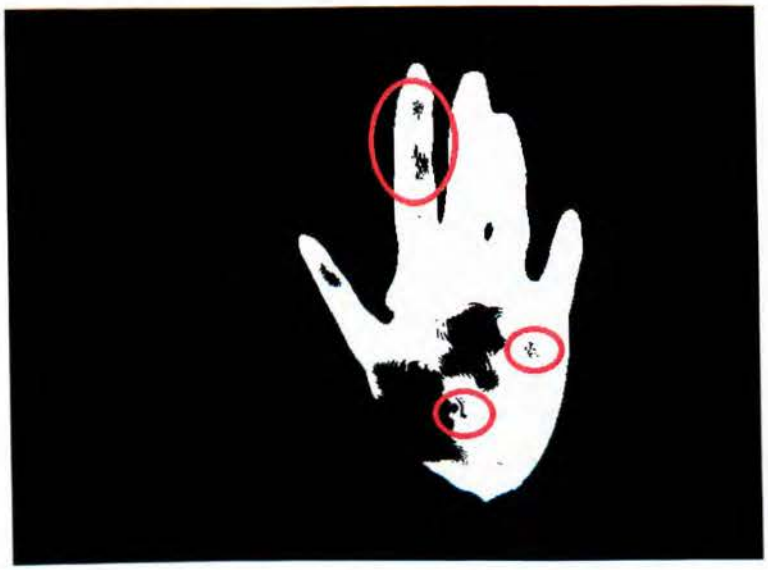
Για την δημιουργία ενός μορφολογικού φίλτρου χρειάζεται ο συνδυασμός των διαδικασιών της διαστολής και της συστολής. Τα αποτελέσματα αυτού του συνδυασμού αποτελούν την διαδικασία του κλεισίματος (closing) και την διαδικασία του ανοίγματος (opening). Όταν η συστολή ακλουθεί μετά την διαστολή έχουμε την πρώτη περίπτωση, ενώ όταν η διαστολή ακολουθεί μετά τη συστολή έχουμε την δεύτερη περίπτωση. Οι συναρτήσεις που εκφράζουν τις διαδικασίες είναι οι παρακάτω:

$f^g=(f\oplus g)\ominus g$ διαδικασία κλεισίματος
$f^g=(f\ominus g)\oplus g$ διαδικασία ανοίγματος

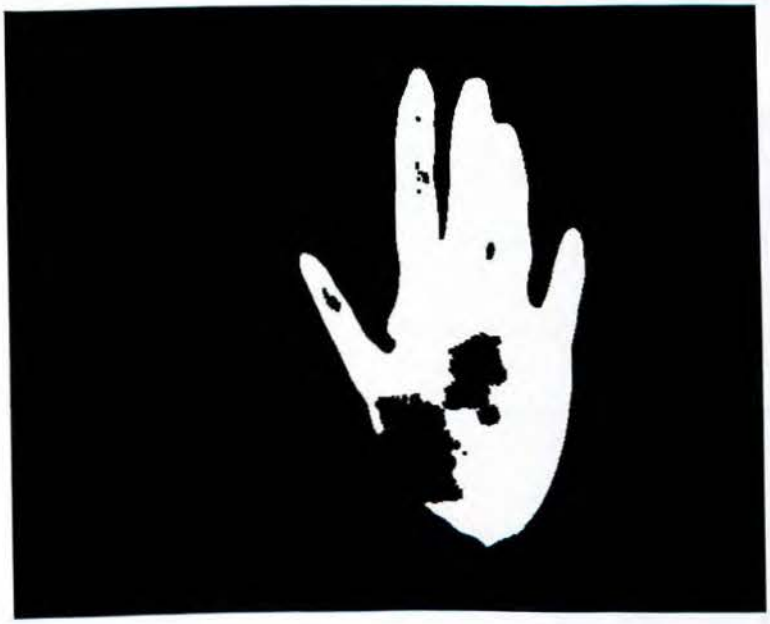
Η πρακτική αξία βρίσκεται στο γεγονός ότι με την διαδικασία του closing εξαφανίζονται οι εσοχές του σήματος στις οποίες δεν μπορεί να εισέλθει το structural element. Η διαδικασία του opening βοηθάει στην εξάλειψη των εξοχών του σήματος στο οποίο δεν μπορεί να εισέλθει το structural element.

[5] Α. ΣΚΟΔΡΑΣ & Β. ΑΝΑΣΤΑΣΟΠΟΥΛΟΣ: 'ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΟΣ ΚΑΙ ΕΙΚΟΝΑΣ'

Στην εργασία μας χρησιμοποιούμε την διαδικασία του closing για να μειώσουμε μικρά ¨μαύρα¨ τμήματα της εικόνας μας. Χρησιμοποιούμε τις συναρτήσεις cvDilate() και cvErode() τις οποίες μας παρέχει η βιβλιοθήκη της OpenCV .
Τα αποτελέσματα φαίνονται στη δεύτερη εικόνα:



2.3.14 *Αρχική εικόνα, στους κύκλους παρατηρούμε τις εσοχές*



2.3.15 *Αποτέλεσμα διαδικασίας κλεισίματος, οι εσοχές έχουν μειωθεί ή εξαφανισθεί*

2.4 Μετατροπή εικόνας

Ακλουθώντας τα προηγούμενα βήματα φτάσαμε στο σημείο διαχωρισμού του χεριού από το περιβάλλον του, τη λεγόμενη τμηματοποίηση (segmentation). Η διαδικασία αυτή είναι προϋπόθεση για να συνεχίσουμε στα επόμενα στάδια, τον προσδιορισμό ακμών και την αναγνώριση του περιγράμματος.
Στη συνέχεια θα δούμε τον τελεστή Canny που χειριζόμαστε για να προσδιορίσουμε τις ακμές. Στην περίπτωση μας ο προσδιορισμός των ακμών είναι τυπικά και η ανίχνευση του περιγράμματος του αντικειμένου που μας ενδιαφέρει. Απαραίτητες για τον ουσιαστικό προσδιορισμό του περιγράμματος είναι η χρήση συναρτήσεων της OpenCV, που μας επιτρέπουν να εξάγουμε σημαντικά χαρακτηριστικά από το αντικείμενο της εικόνας.

- **Προσδιορισμός Ακμών**

Οι διαφορετικές εντάσεις της φωτεινότητας ορίζουν το σύνορο μεταξύ δύο διαφορετικών τμημάτων μιας εικόνας, το σύνορο αυτό θεωρείτε ακμή μιας εικόνας. Οι ακμές είναι βασικό χαρακτηριστικό της εικόνας και περιέχουν πληροφορίες απαραίτητες για τον προσδιορισμό των αντικειμένων και για την πραγματοποίηση διεργασιών όπως η ανάλυση και η επεξεργασία.
Παρά το γεγονός της αυξημένης σημασίας των ακμών για την ανάλυση μιας εικόνας δεν υπάρχει ευρέως αποδεκτός μαθηματικός καθορισμός. Για τον προσδιορισμό τους θα χρησιμοποιήσουμε την μέθοδο Canny, οπότε θα θεωρήσουμε την ακμή σαν τοπική μεταβολή της φωτεινότητας.
Η μέθοδος Canny αναπτύχθηκε το 1986, είναι γρήγορη, αξιόπιστη και χρησιμοποιεί αλγόριθμο με αρκετά στάδια για την ανίχνευση ακμών. Τα στάδια αυτά είναι τέσσερα και αναλύονται παρακάτω.
Πρώτο στάδιο είναι η μείωση του θορύβου της εικόνας που θέλουμε να επεξεργαστούμε. Φιλτράρουμε την εικόνα με το φίλτρο Gauss με τον τρόπο που αναλύσαμε στην προηγούμενη ενότητα. Για να μειωθεί ακόμα περισσότερο η έκθεση του αλγορίθμου στο θόρυβο, η μέθοδος Canny παρέχει και την τεχνική της κανονικοποίησης (normalization). Κανονικοποίηση είναι η διαδικασία αλλαγής του εύρους τιμών της έντασης των pixel. Γίνεται μετατροπή της εικόνας ανακατανέμοντας τις τιμές των pixels, ανάλογα με τις αρχικές τους τιμές. Αυξάνεται η αντίθεση και η εικόνα αποκτά πιο ¨κανονική¨ μορφή.
Επόμενο στάδιο του αλγορίθμου είναι ο υπολογισμός της κλίσης (gradient) κάθε pixel της εικόνας. Σκοπός είναι η αναζήτηση τοπικών μέγιστων της κλίσης. Αναζητούνται τοπικά μέγιστα της κλίσης γιατί αναπαριστούν την αλλαγή της έντασης και κατ' επέκταση το περίγραμμα ενός αντικειμένου. Το διάνυσμα κλίσης (gradient vector) για τα σημεία (x,y) αντιπροσωπεύει την υψηλότερη μεταβολή της έντασης ενός pixel. Η κλίση G που εφαρμόζεται σε δυσδιάστατη εικόνα f(x,y) είναι διάνυσμα με δύο στοιχεία:
$\nabla f(x,y) = [Gx\ Gy] = [\frac{\Delta f}{\Delta x}\ \frac{\Delta f}{\Delta y}]$. Το πρώτο στοιχείο-Gx- είναι η παράγωγος του f(x) και το δεύτερο-Gy- είναι η παράγωγος του f(y). Χαρακτηριστικά του διανύσματος κλίσης είναι το μέτρο και διεύθυνση.

Η διεύθυνση δείχνει την κατεύθυνση κατά την οποία η συνάρτηση f εμφανίζει την μεγαλύτερη μεταβολή στην τιμή έντασης ενός pixel. Δηλαδή δείχνει το προσανατολισμό των ακμών. Η διεύθυνση του gradient vector εκφράζεται ως γωνία.

arctan(Gy/Gx)

Το μέτρο δείχνει το εύρος της μέγιστης διακύμανσης της έντασης ενός pixel. Υπολογίζεται με τον παρακάτω τύπο: $\sqrt{({G_x}^2 + {G_y}^2)}$

Άρα μετά τον υπολογισμό της βαθμίδας κλίσης που αντιστοιχεί σε κάθε τμήμα της εικόνας μπορούμε να βρούμε αυτά τα pixel στα οποία παρατηρούμε απότομη αλλαγή της έντασης.

Οι παράγωγοι της συνάρτησης f μπορούν να υπολογιστούν από τις παρακάτω μάσκες συνέλιξης:

Gx

-1	1

Gy

-1
1

Το πρόβλημα στην εφαρμογή αυτής της μάσκας είναι ότι είναι αρκετά ευαίσθητη στο θόρυβο, επειδή λαμβάνει πληροφορίες μόνο από δύο γειτονικά pixel. Αντιμετωπίζουμε αυτό το πρόβλημα με την χρήση του τελεστή Sobel. Ο τελεστής αυτός είναι από του συχνότερα εφαρμόσιμους τελεστές για την αναγνώριση περιγράμματος. Ο τελεστής Sobel χρησιμοποιεί 3x3 μάσκες στις οποίες σημαντικό ρόλο παίζει το κεντρικό pixel. Ο τελεστής αυτός ανιχνεύει αποτελεσματικά διαγώνια περιγράμματα. Οι μάσκες συνέλιξης του τελεστή είναι οι παρακάτω:

Gx

-1	0	1
-2	0	2
-1	0	1

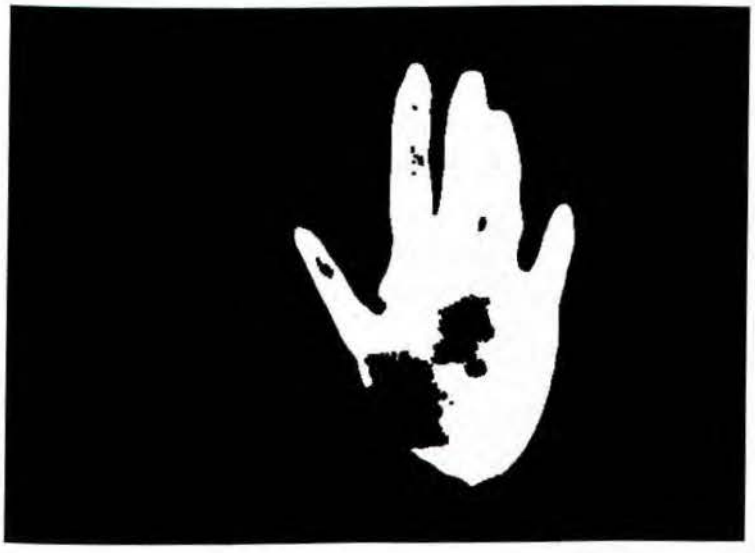
Gy

-1	-2	-1
0	0	0
1	2	1

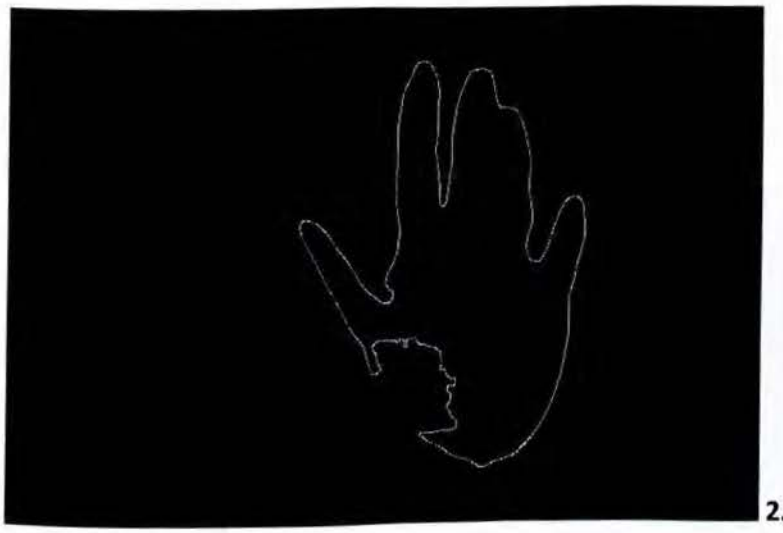
Άρα στο στάδιο αυτό του αλγορίθμου χρειαζόμαστε τον τελεστή Sobel για να υπολογίσουμε την διεύθυνση της κλίσης της έντασης κάθε pixel. Η γωνία της κατεύθυνσης μετά στρογγυλοποιείται σε μια από τις τέσσερις γωνίες, οριζόντια, κάθετη και δύο διαγώνιες. Στην ουσία αυτό που πετυχαίνουμε με τη χρήση του τελεστή Sobel είναι να πάρουμε την πρώτη παράγωγο των αξόνων X και Y της εικόνας.

Τρίτο στάδιο του αλγορίθμου που χρησιμοποιεί η μέθοδος Canny είναι η αναζήτηση για την εύρεση τοπικού μέγιστου της βαθμίδας κλίσης στη κατεύθυνση της κλίσης. Το αποτέλεσμα

μετά από αυτό τα στάδιο είναι ένα σύνολο ακμών με την μορφή δυαδικής εικόνας.
Το τελευταίο στάδιο αναφέρεται στην κατωφλίωση με υστέρηση (hysteresis thresholding). Η κατωφλίωση με υστέρηση είναι μέθοδος η οποία απαιτεί δύο τιμές κατωφλίου, την υψηλή και την χαμηλή. Εάν κάποιο pixel έχει τιμή μέτρου κλίσης μεγαλύτερη από την υψηλή τιμή κατωφλίου τότε αυτό γίνεται δεκτό σαν pixel ακμής. Αντίθετα εάν κάποιο pixel έχει τιμή μικρότερη του χαμηλού κατωφλίου, τότε απορρίπτετε. Εάν η τιμή της κλίσης ενός pixel βρίσκεται ανάμεσα στις τιμές των κατωφλίων , τότε το pixel γίνεται δεκτό μόνο εάν συνδέεται με pixel του οποίου η τιμή είναι μεγαλύτερη από το υψηλό κατώφλι.
Το στάδιο αυτό του αλγορίθμου μπορεί να θεωρηθεί ότι δουλεύει ενισχυτικά στα αποτελέσματα που πήραμε από το τρίτο στάδιο, καθώς αποβάλλει από το σύνολο των ακμών όσες είναι ψευδείς. Παρακάτω βλέπουμε το αποτέλεσμα του προσδιορισμού των ακμών με την μέθοδο Canny.



2.4.1 *Αρχική εικόνα*



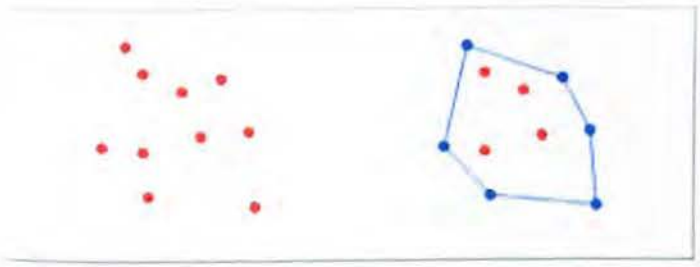
2.4.2 *Αποτέλεσμα Canny*

- **Επεξεργασία Περιγράμματος**

Στην προηγούμενη ενότητα αποκτήσαμε το περίγραμμα του χεριού, όμως πρέπει να αναλύσουμε τα αποτελέσματα της προηγούμενης εικόνας. Αυτός είναι ο ρόλος της επεξεργασίας περιγράμματος. Για να έχουμε τα καλύτερα αποτελέσματα δουλεύουμε με τις συναρτήσεις που μας παρέχει η OpenCV.
Σε αυτήν την φάση θα πάρουμε τις κατάλληλες πληροφορίες από το περίγραμμα για να καθορίσουμε τις χειρονομίες. Τα δομικά στοιχεία που θα εξάγουμε είναι η κυρτή κοιλότητα (convex hull) και η έλλειψη.

- **Κυρτό Πολύγωνο**

Η δομή αυτή είναι στοιχειώδης για να λάβουμε τις τιμές της κυρτότητας του πολυγώνου που περικλείει τις χειρονομίες. Η κυρτότητα υπολογίζεται από τον αλγόριθμο του Sklansky. Ο Sklansky πρότεινε την χρήση του 3-coins αλγορίθμου για την εύρεση ενός απλού πολυγώνου. Ο αλγόριθμος αυτός βρίσκει και σημειώνει την κάθε κορυφή από ένα πολύγωνο ακολουθώντας την φορά της κατεύθυνσης του ρολογιού. Μετά επεξεργάζεται μια ομάδα τριών κορυφών -πίσω ,κέντρο, μπροστά- και αντιλαμβάνεται αν η κίνηση του χεριού είναι προς τα αριστερά ή προς τα δεξιά. Στη συνέχεια μετράει πόσες κορυφές σημειωμένες ως κέντρο περιλαμβάνονται στην κίνηση προς τα δεξιά του χεριού.



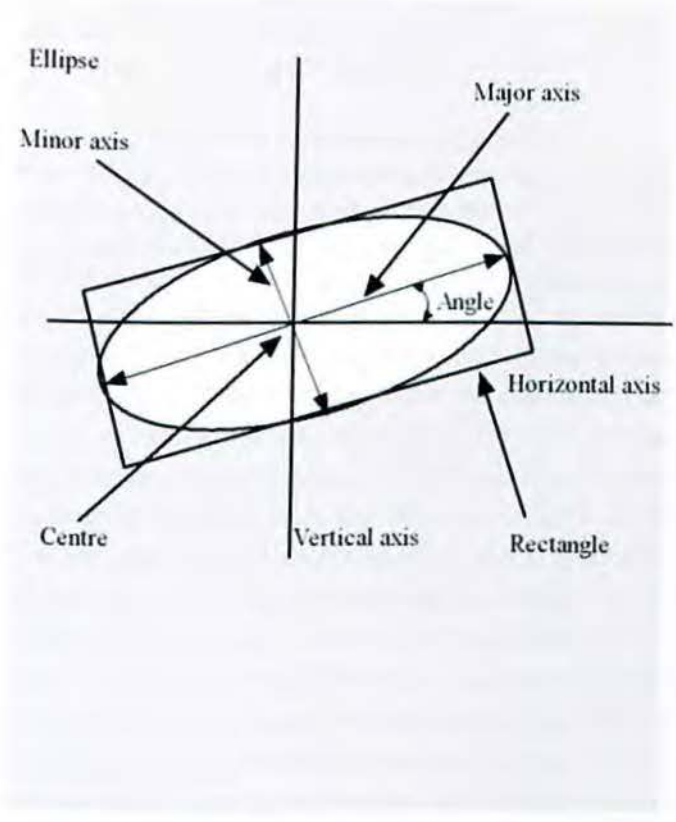
2.4.3 *Ένα κυρτό πολύγωνο από ένα σύνολο σημείων*

- **Έλλειψη**

Η έλλειψη συμπεριλαμβάνει όλα εκείνα τα σημεία που συνθέτουν το περίγραμμα του χεριού που πραγματοποιεί την χειρονομία. Η έλλειψη η οποία σχηματίζεται γύρω από το χέρι περιέχει πληροφορίες οι οποίες είναι απαραίτητες για τον καθορισμό των χειρονομιών.

Τα στοιχεία τα οποία εξάγουμε από την έλλειψη είναι η γωνία (Angle) και η αναλογία μεταξύ κυρίου άξονα (major axis) και μικρότερου άξονα (minor axis).

Στο παρακάτω σχήμα φαίνονται τα χαρακτηριστικά που εξάγουμε.



2.4.4 *Χαρακτηριστικά έλλειψης που εξάγουμε*

Εφαρμογή

Στο κεφάλαιο αναφερθήκαμε στις μεθόδους και τις τεχνικές τις οποίες ακολουθήσαμε για να πραγματοποιήσουμε την αναγνώριση του ανθρώπινου χεριού. Σε πρώτη φάση περιγράφουμε τεχνικές της επεξεργασίας εικόνας, όπως κατωφλίωση, τμηματοποίηση και φιλτράρισμα εικόνας. Σε δεύτερη φάση επεξηγούμε μεθόδους της υπολογιστικής όρασης, όπως την εύρεση ακμών με την μέθοδο Canny. Η συνέχεια της περιγραφής των τεχνικών της υπολογιστικής όρασης ,που χρησιμοποιήθηκαν κατά την ανάπτυξη της πτυχιακής εργασίας, περιλαμβάνονται στο επόμενο κεφάλαιο.
Λαμβάνουμε την εικόνα που είναι προς επεξεργασία. Στη συνέχεια φροντίζουμε να μεταφερθούμε στο χρωματικό πρότυπο YCrCb από το RGB. Αυτό συμβαίνει για να απομονώσουμε το στοιχείο της φωτεινότητας, το οποίο προκαλεί τα περισσότερα προβλήματα στις μεθόδους αναγνώρισης. Η μετατροπή από το χρωματικό πρότυπο RGB στο YCrCb γίνεται με τη βοήθεια της συνάρτησης cvCvtColor() της βιβλιοθήκης OpenCV. Οι σχέσεις που χειρίζεται η συνάρτηση για να πετύχει τον σκοπό είναι οι παρακάτω.

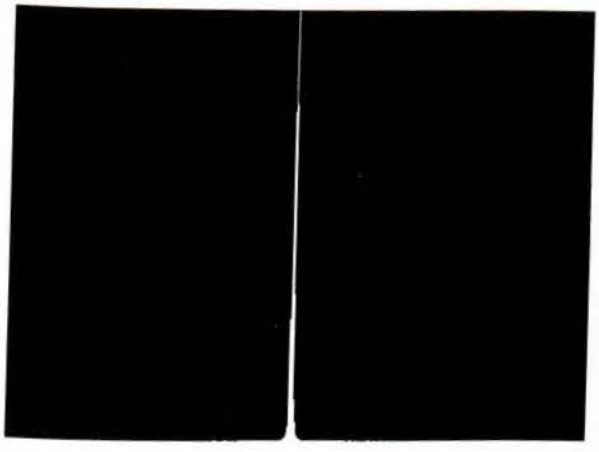
Y <- 0.299*R + 0.587*G + 0.114*B Μετατροπή από τον RGB χρωματικό χώρο στο YCrCb
Cr <- (R-Y)*0.713 + delta
Cb <- (B-Y)*0.564 + delta

R <- Y + 1.403*(Cr - delta) Μετατροπή από το YCrCb χρωματικό χώρο στο RGB
G <- Y - 0.344*(Cr - delta) - 0.714*(Cb - delta)
B <- Y + 1.773*(Cb - delta),

{ 128 for 8-bit εικόνες,
όπου delta = { 32768 for 16-bit εικόνες
{ 0.5 for floating-point εικόνες

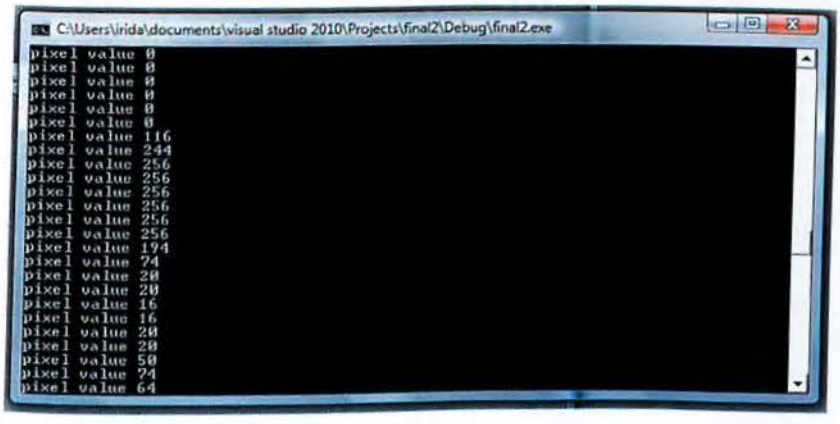
Y, Cr και Cb καλύπτουν ολόκληρο το εύρος τιμών.
Αφού βρισκόμαστε στο χρωματικό πρότυπο που επιθυμούμε, εφαρμόζουμε τη μέθοδο της κατωφλίωσης για να ξεχωρίσουμε το ανθρώπινο χέρι από το περιβάλλον του. Διαθέτουμε εικόνα βάθους 8-bit, 3 καναλιών σε χρωματικό πρότυπο YCrCb. Για τον λόγο αυτό ορίζουμε μέγιστη και ελάχιστη τιμή κατωφλίωσης για κάθε κανάλι. Οι τιμές αυτές είναι Y [60,250] Cr[135,170],Cb[90,135], οπου YCrCb [0,255]. Η τιμή κάθε εικονοστοιχείου που βρίσκεται μεταξύ των παραπάνω τιμών θα εμφανίζεται ως άσπρο χρώμα στη τελική δυαδική εικόνα. Σε αντίθετη περίπτωση θα εμφανίζεται ως μαύρο χρώμα, το αποτέλεσμα μπορούμε να το δούμε στη εικόνα 2.3.8. Στη συνέχεια λειαίνουμε την εικόνα μας εφαρμόζοντας το φίλτρο Gauss. Καθορίζουμε τις τιμές του παραθύρου του φίλτρου για τον οριζόντιο άξονα nx=3 και για τον κάθετο άξονα ny=3. Δηλαδή χρησιμοποιούμε μια μάσκα συνέλιξης 3x3. Με αυτό τον τρόπο φιλτράρουμε την εικόνα που επιθυμούμε και αποφεύγουμε το φαινόμενο αλλοίωσης των ακμών. Η σταθερή απόκλιση της γκαουσιανής διανομής (σ), έχει τιμές σ_x=5.45, σ_y=5.45 και δίνεται από την εφαρμογή της σχέσης $\sigma_x=(n_x/2)*0.3+0.8$, $\sigma_y=(n_y/2)*0.3+0.8$, n: μέγεθος παραθύρου του φίλτρου. Το φιλτραρισμένο αποτέλεσμα μπορούμε να το δούμε στην εικόνα 2.3.13.
Επόμενο βήμα είναι η διαδικασία του κλεισίματος , κατά την οποία ένας 3x3 πίνακας χρησιμοποιείται σαν δομικό στοιχείο. Στη περίπτωση μας ο πίνακας εφαρμόζεται μια φορά σε κάθε κανάλι χρώματος. Το τελικό αποτέλεσμα μπορούμε να το δούμε στην εικόνα 2.3.15. Η επιλογή του συγκεκριμένου πίνακα έγινε μετά από δοκιμές που πραγματοποιήσαμε. Ο αλγόριθμος Canny που χρησιμοποιούμε για να ανιχνεύσουμε τις ακμές της εικόνας υποστηρίζεται από τη συνάρτηση cvCanny(). Σε αυτή τη περίπτωση πρέπει να ορίσουμε τις τιμές του ανώτερου και του κατώτερου κατωφλίου. Οι τιμές που ορίζουμε για το κατώτερο κατώφλι είναι 129 και για το ανώτερο είναι 100. Εάν η τιμή του μέτρου κλίσης ενός pixel είναι μεγαλύτερη από την τιμή του κατώτερου κατωφλίου τότε αυτό το pixel είναι ακμή. Η τιμή του ανώτερου κατωφλίου είναι για να ελέγχει αν τα ¨ κεντρικά ¨ pixel είναι ακμές. Για να είναι ακμές πρέπει η τιμή του μέτρου κλίσης να είναι μεγαλύτερη από την τιμή του ανώτερου κατωφλίου. Για να κατανοήσουμε το λόγο για τον οποίο επιλέξαμε την τιμή 129 για κατώτερο όριο κατωφλίου πρέπει να εφαρμόσουμε τους πίνακες συνέλιξης του τελεστή Sobel στην αρχική μας εικόνα. Το αποτέλεσμα φαίνεται στην εικόνα 2.4.5, στη συνέχεια από το ιστόγραμμα της εικόνας, το οποίο φαίνεται στην 2.4.6, επιλέγουμε την τιμή του κατώτερου κατωφλίου. Η γραμμή που βλέπουμε είναι το σημείο εκείνο που παρατηρούμε έντονη μεταβολή της τιμής της έντασης των pixel της εικόνας .

2.4.5 *Εικόνα μετά την εφαρμογή του τελεστή Sobel (1η παράγωγος)*



2.4.6 Ιστόγραμμα της εικόνας, το σημείο αλλαγής της έντασης των pixel είναι η λευκή γραμμή

Ο οριζόντιος άξονας έχει τιμές από 0 εως 255, άρα η ιδανική τιμή για το lower threshold είναι το 129.
Η τιμή για το ανώτερο όριο κατωφλίωσης είναι 100. Όταν η τιμή του μέτρου κλίσης είναι μεγαλύτερη από το 100 τότε έχουμε ακμή.



2.4.7 *Παράδειγμα τιμών μέτρου κλίσης*

Κεφάλαιο 3°
Λήψη Χαρακτηριστικών

Στο προηγούμενο κεφάλαιο παρακολουθήσαμε τα βήματα τα οποία μας οδήγησαν στην αναγνώριση και απομόνωση του χεριού από το περιβάλλον του. Οι αλγόριθμοι που χρησιμοποιήσαμε μας παρέχουν και επιπρόσθετες πληροφορίες τις οποίες θα εκμεταλλευτούμε για να καθορίσουμε τις χειρονομίες. Δημιουργούμε ένα μηχανισμό σύγκρισης ο οποίος θα προσδιορίζει τη μοναδικότητα κάθε χειρονομίας. Η ιδιότητα αυτή δίνει ξεχωριστή έννοια σε κάθε χειρονομία και είναι το τελικό στάδιο της αναγνώρισης.

3.1 Ορισμός Χαρακτηριστικών

Χαρακτηριστικά είναι τα στοιχεία εκείνα που καθορίζουν την μοναδικότητα κάθε χειρονομίας. Αυτά τα οποία εξάγουμε πρέπει να είναι ανεξάρτητα του μεγέθους του χεριού που απεικονίζεται στο βίντεο που τραβάμε. Τέτοια χαρακτηριστικά είναι:

- **Γωνία:** είναι ανεξάρτητο μεγέθους χαρακτηριστικό που το εξάγουμε από την έλλειψη που ζωγραφίζουμε γύρω από το περίγραμμα του χεριού. Θέτει την κλίση του χεριού.

- **Αναλογία μεταξύ των αξόνων:** Είναι –και αυτό- χαρακτηριστικό το οποίο το λαμβάνουμε από το σχήμα της έλλειψης και αναφέρεται στην αναλογία μεταξύ κύριου και δευτερεύοντος άξονα της έλλειψης.

- **Αριθμός Κυρτών Σημείων:** Αφού έχουμε αποκτήσει το περίγραμμα του χεριού μπορούμε να σχηματίσουμε ένα πολύγωνο που περικλείει το περίγραμμα. Ο αριθμός των κυρτών σημείων του πολυγώνου είναι το χαρακτηριστικό το οποίο μας ενδιαφέρει.

- **Λόγος περιμέτρου προς εμβαδό:** Είναι ανεξάρτητο μεγέθους χαρακτηριστικό δηλαδή η τιμή του δεν αλλάζει ακόμα και όταν το χέρι κινείται. Στη συνέχεια θα γίνει εκτενέστερή αναφορά στο χαρακτηριστικό αυτό.

3.2 Λήψη Χαρακτηριστικών

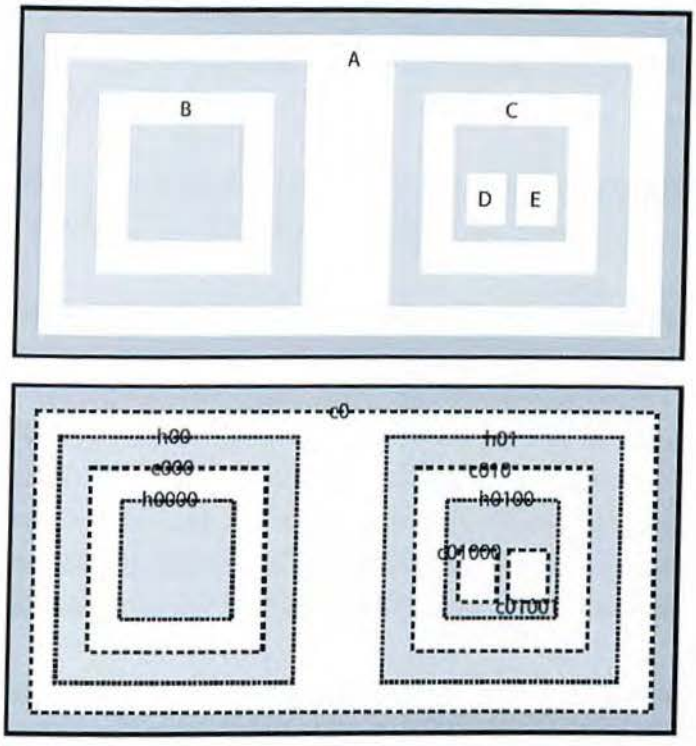
Η εξαγωγή των χαρακτηριστικών (feature extraction) είναι το στάδιο το οποίο θα μας δώσει τις απαραίτητες πληροφορίες για να ταξινομήσουμε μια χειρονομία. Για να γίνει αυτό πρέπει να έχουμε ένα αρχείο το οποίο θα περιέχει τις πληροφορίες των χαρακτηριστικών για κάθε διαφορετική χειρονομία.
Το πρώτο χαρακτηριστικό που εξάγουμε είναι η γωνία κλίσης του χεριού. Τα βήματα τα οποία ακολουθούμε για την εξαγωγή του χαρακτηριστικού της γωνίας είναι, η εύρεση του περιγράμματος του χεριού, ο σχεδιασμός του τετραγώνου που περικλείει το περίγραμμα και η εύρεση του ελαχίστου τετραγώνου που σχηματίζεται γύρω από το περίγραμμα του χεριού.

- **Εύρεση Περιγράμματος**

Αλγόριθμοι εύρεσης ακμών, όπως ο αλγόριθμος Canny, μπορούν να βρίσκουν τις ακμές οι οποίες χωρίζουν τα τμήματα μιας εικόνας, όμως δεν μας παρέχουν πληροφορίες για τις ακμές σαν οντότητα. Το επόμενο βήμα είναι να συνδέσουμε αυτά τα εικονοστοιχεία ακμών και να σχηματίσουμε το περίγραμμα.

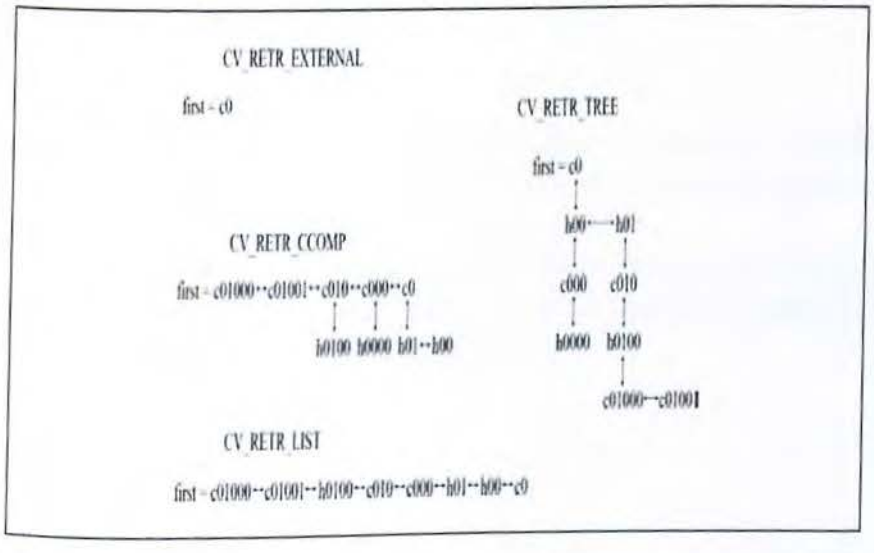
Περίγραμμα είναι μια λίστα από σημεία που παριστάνουν μια καμπύλη ή κοιλία σε μια εικόνα. Το περίγραμμα παρουσιάζεται σαν μια ακολουθία σημείων στην οποία κάθε καταχώρηση στην ακολουθία κωδικοποιεί πληροφορίες για την τοποθεσία του επόμενου σημείου στη καμπύλη.

Σημαντικό είναι να γνωρίζουμε ποιά είναι τα εσωτερικά και ποιά τα εξωτερικά σύνορα του αντικείμενου του οποίου βρίσκουμε το περίγραμμα. Στη παρακάτω εικόνα το πάνω μέρος δείχνει τις άσπρες περιοχές της εικόνας σημειωμένες από το A εως το E. Ακριβώς από κάτω φαίνεται το περίγραμμα αυτών των περιοχών. Οι διακεκομμένες γραμμές που παρατηρούμε αντιπροσωπεύουν τα εξωτερικά σύνορα του άσπρου τμήματος, και τα σημειώνουμε ως c (contours). Τα εσωτερικά όρια παριστάνονται από τις γραμμές που είναι σημειωμένες ως h (holes). Με αυτόν τον τρόπο διαχωρίζουμε τα άσπρα τμήματα της εικόνας του παραδείγματος από τα μαύρα τμήματα.



3.2.1 *Τα εσωτερικά σύνορα (C) και τα εξωτερικά σύνορα (h) των άσπρων τμημάτων της εικόνας*

Για την εύρεση του περιγράμματος χρησιμοποιούμε την συνάρτηση cvFindContours(). Η συνάρτηση αυτή τοποθετεί τα περιγράμματα που έχουν βρεθεί σε ένα κόμβο μορφής δέντρου που κωδικοποιεί τις σχέσεις μεταξύ των περιγραμμάτων. Ένα δέντρο περιγραμμάτων του παραπάνω παραδείγματος διαθέτει ένα βασικό κόμβο, το c0, με τα h00 και h01 να αποτελούν το δευτερεύον κόμβο. Τα στοιχεία του δευτερεύοντος κόμβου θα αποτελούν βασικό κόμβο για τα επόμενα περιγράμματα και ούτω καθεξής.



3.2.2 *Κόμβοι μορφής δέντρου, διαφορετικού τρόπου τοποθέτησης των περιγραμμάτων*

Στην εικόνα 3.2.2 βλέπουμε τους διαφορετικούς τύπους της μορφής δέντρου με τους οποίους μπορούμε να κατατάξουμε τα περιγράμματα της εικόνας.
Αυτοί είναι:

- CV_RETR_EXTERNAL: Ανακτά μόνο τα εξωτερικά περιγράμματα. Στο παράδειγμα που μελετάμε το μόνο εξωτερικό περίγραμμα είναι το c0

- CV_RETR_LIST: Ανακτά όλα τα περιγράμματα και τα τοποθετεί σε μια λίστα. Στην περίπτωση μας οκτώ περιγράμματα τοποθετούνται στη λίστα.

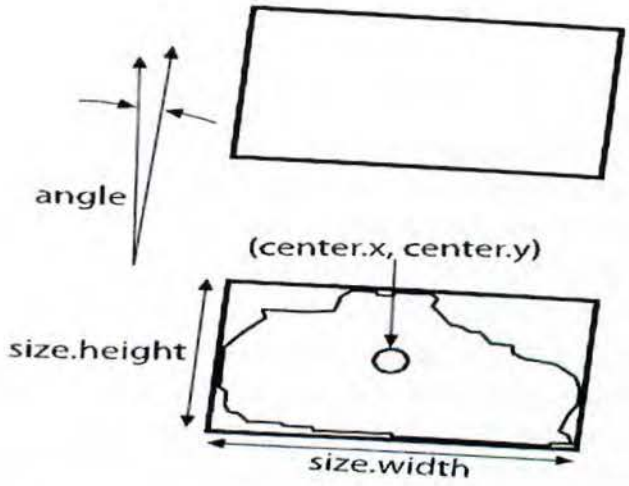
- CV_RETR_CCOMP: Ανακτά όλα τα περιγράμματα και τα ιεραρχεί σε δύο επίπεδα, όπου στο υψηλό επίπεδο ανήκουν εξωτερικά περιγράμματα, ενώ στο δεύτερο επίπεδο ανήκουν εσωτερικά περιγράμματα. Στο παράδειγμα μας έχουμε πέντε εξωτερικά όρια από τα οποία τρία περιέχουν εσωτερικά περιγράμματα. Το εξωτερικό περίγραμμα c0 περιέχει δύο holes, την h01 και h00.

- CV_RETR_TREE: Ανακτά όλα τα περιγράμματα και ανακατασκευάζει ολόκληρη την ιεραρχία των εμβολιασμένων περιγραμμάτων. Συνδέει τα εξωτερικά περιγράμματα με τα εσωτερικά, ξεκινώντας από το εξωτερικό περίγραμμα c0 και καταλήγοντας στα εσωτερικά περιγράμματα.

Άξιο αναφοράς είναι και ο τρόπος με τον οποίο θα προσεγγίσουμε το περίγραμμα. Ο συνηθέστερος και αποτελεσματικότερος τρόπος είναι η συμπίεση οριζοντίων, διαγώνιων, και κάθετων τμημάτων, αφήνοντας μόνο τα σημεία τερματισμού. Η μέθοδος που το πραγματοποιεί αυτό είναι η CV_CHAIN_APPROX_SIMPLE.

- **Εύρεση ελάχιστου τετραγώνου**

Τα σημεία που σχηματίζουν την περίμετρο του χεριού μας περιέχουν αρκετές πληροφορίες τις οποίες μπορούμε να εξάγουμε αλλά χρειάζεται και η εύρεση του ελάχιστου τετραγώνου που περικλείει την περίμετρο.
Για να βρούμε το ελάχιστο τετράγωνο που περικλείει το περίγραμμα του χεριού η OpenCV μας παρέχει την συνάρτηση cvMinAreaRect2(). Το αποτέλεσμα της εφαρμογής αυτής της συνάρτησης φαίνεται στην παρακάτω εικόνα.



3.2.3 *Αποτέλεσμα εφαρμογής συνάρτησης cvMinAreaRect2()*

Όπως μπορούμε να παρατηρήσουμε το τετράγωνο περικλείει την εικόνα και μας παρέχει πληροφορίες για την κλίση (angle) της εικόνας, το μέγεθος (size) και το κέντρο (center). Γωνία κλίσης είναι η γωνία που σχηματίζεται από το νοητό κάθετο άξονα y -σε καρτεσιανό σύστημα αναφοράς- και το τετράγωνο, και έχει τιμές που εκτείνονται από 0^0 εως 90^0. Με αυτό τον τρόπο αποκτάμε το χαρακτηριστικό της γωνίας.

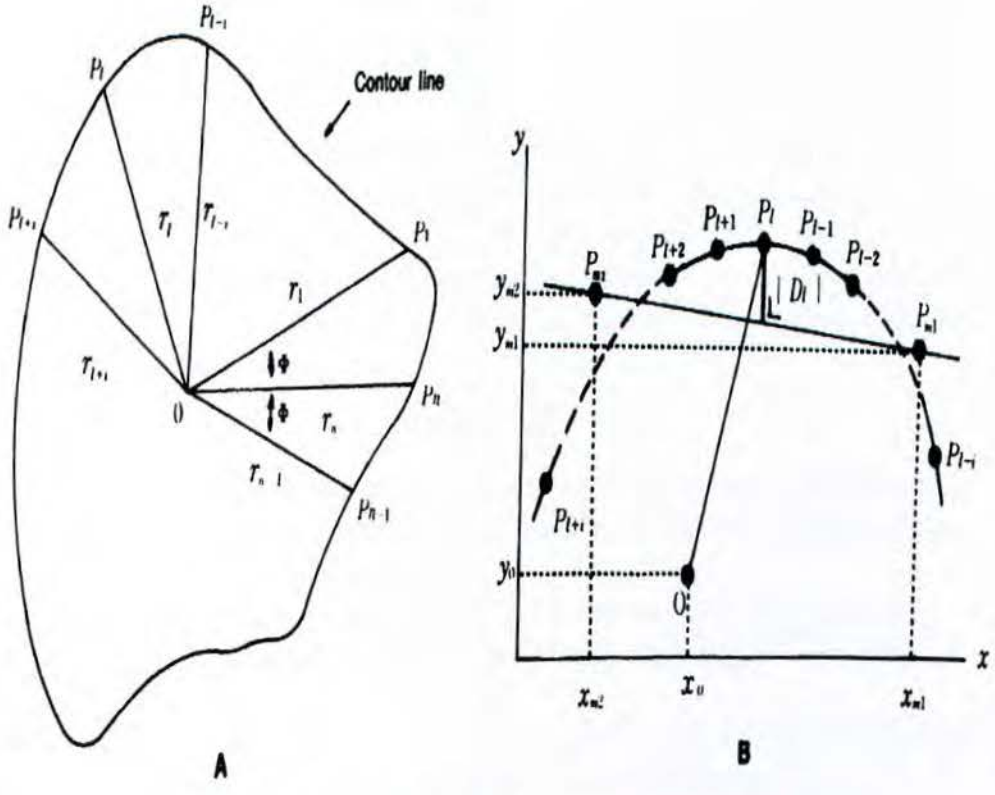
- **Αναλογία μεταξύ των αξόνων:** Το δεύτερο χαρακτηριστικό που πρέπει να εξάγουμε είναι η αναλόγια μεταξύ του πλάτους (size.width) του τετραγώνου προς το ύψος (size.height) του. Εφόσον έχουμε ακολουθήσει όλες τις παραπάνω διαδικασίες ή εξαγωγή αυτού του χαρακτηριστικού είναι αρκετά εύκολη καθώς η πληροφορία μας παρέχεται από το ελάχιστο τετράγωνο που σχηματίζεται γύρω από το περίγραμμα.

- **Κυρτά Σημεία**

Η δυνατότητα να σχηματίσουμε ένα πολύγωνο που ενώνει τις κορυφές των καμπυλών τις οποίες σχηματίζει το contour, είναι πολύ σημαντική για την αναγνώριση της κατάστασης του χεριού αλλά και του αριθμού των δακτύλων. Αφού υπολογίσουμε την convex hull μετά υπολογίζουμε τα ελαττώματα της κυρτότητας (convexity defects).
Η μέθοδος για την αναγνώριση των convexity defects προτάθηκε από τον Homma[6] και περιέχει τέσσερα κύρια βήματα επεξεργασίας.

[6] K. Homma and E.-I. Takenaka, "An image processing method for feature extraction of space-occupying lesions," *Journal of Nuclear Medicine* 26 (1985) σελ.1435-1438

1. Η απόκτηση των δεδομένων από το περίγραμμα κάθε εικόνας πρέπει να γίνεται κάτω από καθορισμένες συνθήκες για να έχουμε την ακριβή ταξινόμηση και σύγκριση των εικόνων. Σύμφωνα με τους κανόνες τα δομικά δεδομένα ενός περιγράμματος παρουσιάζονται από τα στοιχεία Pl (l=1,2,3, . . . ,n: n=360/Φ), Φ είναι το σταθερό γωνιακό διάστημα. Η αρχή 0(x_0,y_0) τοποθετείται τυχαία στην εικόνα και το σημείο Pl είναι οποιοδήποτε σημείο του περιγράμματος που έχει την υψηλότερη ένταση. Τα υπόλοιπα σημεία Pl απεικονίζονται ως σημεία με την μέγιστη ένταση στα μεμονωμένα διασταυρούμενα ευθύγραμμα τμήματα. Αυτά τα ευθύγραμμα τμήματα ζωγραφίζονται ακτινικά από την αρχή με σταθερή γωνία Φ. Στη συνέχεια μετράμε το μήκος rl (l=1,2,3, . . . ,n) από την αρχή έως το Pl.



3.2.4 Νόμοι απόκτησης των δεδομένων του περιγράμματος (A) και ο καθορισμός της απόστασης Dl (B)

2. Η αρχή 0(x_0,y_0) και τα σημεία Pl (x_l,y_l) λαμβάνονται στο πολικό σύστημα συντεταγμένων (r,Θ) και μετατρέπονται στο ορθογώνιο σύστημα (x,y).

$x_l = r_l^n * \cos\Theta_l + x_o$, $y_l = r_l^n * \sin\Theta_l + y_0$ όπου $\Theta_l = l * \Phi$ και r_l^n είναι η κανονικοποιημένη τιμή του r_l

3. Ακολούθως, τα σημεία P_{m1} (x_{m1}, y_{m1}) και P_{m2}(x_{m2},y_{m2}) σημείων N(N=1,2,3, . . .) στις δύο πλευρές του Pl υπολογίζονται από τις εξισώσεις (εικόνα 3.2.4 B):

$$x_{m1} = \frac{1}{N} \sum_{i=1}^{N} x_{l-i} \quad , \quad y_{m1} = \frac{1}{N} \sum_{i=1}^{N} y_{l-i}$$

$$x_{m2} = \frac{1}{N} \sum_{i=1}^{N} x_{l-i} \quad , \quad y_{m2} = \frac{1}{N} \sum_{i=1}^{N} y_{l-i}$$

Αυτός ο γεωμετρικός μέσος έχει ως ρόλο την λείανση, η οποία ανιχνεύει διαφορετικά μεγέθη των ελαττωμάτων και αποκλείει τον θόρυβο του background.

4. Η κυρτή αλλά και η κοίλη δομή δίνεται από το ποσοτικό μέγεθος της απόστασης, Dl, κάθετη γραμμή προς την ευθεία που ενώνει τα P_{m1} και P_{m2} σημεία , όπως φαίνεται και στο B σχήμα της εικόνας 3.2.4. Η απόσταση υπολογίζεται από τον παρακάτω τύπο:

$$D_l = \frac{\left[\frac{y_{m2} - y_{ml}}{x_{m2} - x_{ml}}\right] * (x_{m1} - x_l) + Yl - y_{m1}}{[(y_{m2} - y_{ml}) / (x_{m2} - x_{ml})]^2 + 1}$$

Η τιμή D_l χρησιμοποιείται για να χαρακτηρίζει την κυρτότητα της γραμμής του περιγράμματος. Η διάκριση μεταξύ κυρτού τμήματος και κοίλου γίνεται από την τιμή της D_l. Η απόσταση D_l εκφράζεται από την τοποθεσία των τριών σημείων P_l, P_{m1} και P_{m2}. Για παράδειγμα αν το P_l βρίσκεται πιο κοντά στο κέντρο της εικόνας απ' ότι η ευθεία γραμμή που ενώνει το P_{m1} και το P_{m2} , τότε χαρακτηρίζεται σαν κυρτή δομή. Όταν συμβαίνει το αντίθετο χαρακτηρίζεται σαν κοίλη δομή.

Με τη μέθοδο την οποία μόλις περιγράψαμε βρίσκουμε τις κυρτές και τις κοίλες δομές του περιγράμματος οποιουδήποτε αντικειμένου μιας εικόνας. Η απόσταση D_l αντιπροσωπεύει τις κυρτές και κοίλες δομές του περιγράμματος.

Στη περίπτωση του χεριού -που μας ενδιαφέρει- τα κυρτά σημεία στο περίγραμμα του χεριού αντιστοιχούν στη κορυφή των δακτύλων. Τα κοίλα σημεία αντιστοιχούν στη βάση των δακτύλων.

Στην εργασία μας μετράμε των αριθμό των κυρτών σημείων του περιγράμματος και αυτό το χαρακτηριστικό το χρησιμοποιούμε για να καθορίσουμε τη χειρονομία που πραγματοποιούμε κάθε στιγμή.

- **Λόγος περιμέτρου προς εμβαδόν**

Για να κατανοήσουμε τον τρόπο που παίρνουμε τα χαρακτηριστικά της περιμέτρου και του εμβαδού πρέπει να εξηγήσουμε το blob extraction.

- Blob Extraction

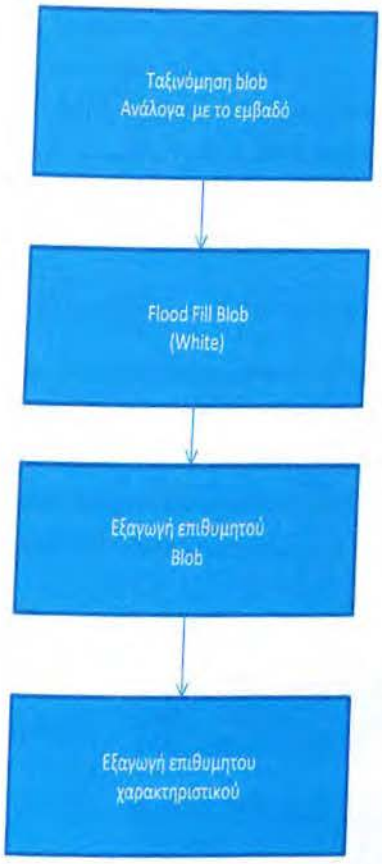
Απαραίτητο για να έχουμε blob extraction είναι η εικόνα που επεξεργαζόμαστε να είναι δυαδική. Δυαδική εικόνα αποκτάμε με τις τεχνικές του thresholding (κατώφλι) και του φιλτραρίσματος τις οποίες αναλύσαμε στο προηγούμενο κεφάλαιο.
Η διαδικασία εξαγωγής blob επιτρέπει την κατηγοριοποίηση των pixel της ψηφιακής εικόνας σε ξεχωριστές ομάδες. Δηλαδή δημιουργούνται διαφορετικά τμήματα στην εικόνα τα οποία έχουν διαφορετικά στοιχεία. Τα blob μπορούν να μετρηθούν, να φιλτραριστούν και να επεξεργαστούν με μεγάλη ευκολία.
Για να πετύχουμε την λήψη του blob η εικόνα σαρώνεται οριζόντια από την πάνω δεξιά γωνία ως την κάτω αριστερή, παίρνοντας την τιμή από κάθε pixel. Η εικόνα είναι δυαδική, άρα το pixel μπορεί να είναι είτε μαύρο είτε άσπρο. Τα άσπρα εικονοστοιχεία αναπαριστάνουν το χέρι ενώ τα μαύρα pixel αναπαριστάνουν το background. Μόλις εντοπιστεί η τιμή του pixel και σημειωθεί σαν άσπρο ή μαύρο, ελέγχονται τα γειτονικά του pixel με τρόπο ο οποίος καθορίζεται από τύπο της συνδεσιμότητας του σημειωμένου pixel με τα γειτονικά του pixel.
Στις εικόνες δύο διαστάσεων έχουμε δύο τύπους συνδεσιμότητας των pixel με τα γειτονικά τους pixel.

1. 4 connected: Αυτά τα pixel είναι γείτονες με κάθε pixel που ακουμπάει μια από τις άκρες τους. Συνδέονται οριζόντια και κάθετα, βρίσκονται σε συντεταγμένες $(x \pm 1, y)$ ή $(x, y \pm 1)$ και είναι συνδεμένα με κάθε pixel στο (x,y)
2. 8 connected: Αυτά τα pixel είναι γείτονες με κάθε εικονοστοιχείο που ακουμπά οποιαδήποτε πλευρά ή γωνία του. Αυτά τα pixel συνδέονται οριζόντια, κάθετα και διαγώνια. Οι συντεταγμένες τους είναι $(x \pm 1, y \pm 1)$ ή $(x \pm 1, y \mp 1)$ και συνδέονται με κάθε pixel στο (x,y).

Εφόσον γίνει και ο έλεγχος των γειτονικών pixel καθορίζεται σε ποιά ομάδα ανήκουν. Με αυτό τον τρόπο δημιουργούνται οι ξεχωριστές ομάδες στην εικόνα.

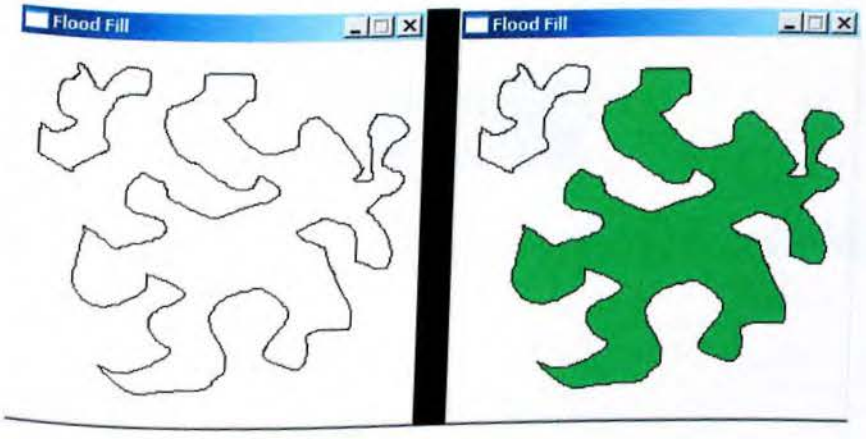
Στην εργασία μας χρησιμοποιούμε την βιβλιοθήκη cvBloblib η οποία μας παρέχει τις κατάλληλες συναρτήσεις για να αναπτύξουμε αλγόριθμο ο οποίος θα βρίσκει τα blobs της εικόνας και θα παρέχει τα χαρακτηριστικά των διαφορετικών blobs της εικόνας.

Παρακάτω παρουσιάζουμε το διάγραμμα ροής του αλγορίθμου της διαδικασίας εξαγωγής των blob.



3.2.5 *Διάγραμμα ροής του αλγορίθμου που ακολουθητέε για να γίνει το blob extraction*

Έχοντας ακολουθήσει όλες τις διαδικασίες που έχουν περιγραφεί έχουμε φτάσει στο σημείο η εικόνα μας να αποτελείτε από δύο διαφορετικά blobs. Το ένα είναι μαύρο και παριστάνει το background της εικόνας, και το άλλο blob είναι άσπρο και παριστάνει το χέρι. Αυτά τα δύο διαφορετικά blob ταξινομούνται ανάλογα με το εμβαδό, το χέρι που είναι το blob το οποίο μας ενδιαφέρει έχει μικρότερο εμβαδό από το background. Με αυτό τον διαχωρισμό που γίνεται στο πρώτο στάδιο του αλγορίθμου εφαρμόζουμε την διαδικασία του Flood Fill μόνο στο τμήμα της εικόνας που δείχνει το χέρι.
Ο σκοπός του Flood Fill είναι να γεμίσει ολόκληρες περιοχές συνδεμένων pixel με το ίδιο χρώμα. Στη παρακάτω εικόνα βλέπουμε το αποτέλεσμα της εφαρμογής του Flood fill.

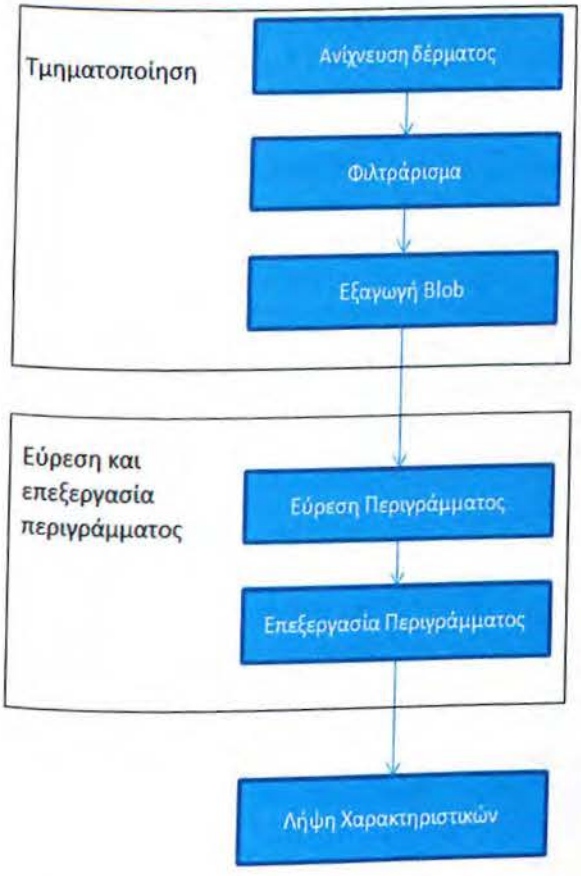


3.2.6 *Αποτέλεσμα Flood Fill*

Με την εφαρμογή του Flood Fill τα ελαττώματα του blob του χεριού μειώνονται και έχουμε καλύτερη απεικόνιση του χεριού.
Στο επόμενο βήμα εξάγουμε –απομονώνουμε και αναλύουμε- το blob του χεριού. Η ανάλυση γίνεται εφικτή με τις συναρτήσεις που μας παρέχει η βιβλιοθήκη cvBlobsLib. Το τελικό στάδιο είναι η λήψη της περιμέτρου και του εμβαδού. Τα δύο αυτά στοιχεία τα συνδυάζουμε κάτω από την σχέση:

(περίμετρος)^2/εμβαδόν. Η σχέση αυτή αποτελεί το τελικό χαρακτηριστικό που εξάγουμε.

Συμπέρασμα: Με την εξαγωγή και του τελευταίου χαρακτηριστικού που μας ενδιαφέρει ολοκληρώσαμε την διαδικασία της feature extraction. Η διαδικασία αυτή αποτελεί, ουσιαστικά, το τελευταίο βήμα της επεξεργασίας της εικόνας. Τα βήματα τα οποία ακολουθήσαμε και μας οδήγησαν σε αυτό το αποτέλεσμα μπορούν να περιγραφούν σχηματικά από την παρακάτω εικόνα.



3.2.7 *Το σχήμα περιγράφει τη διαδικασία που μας οδήγησε στη λήψη των επιθυμητών χαρακτηριστικών*

3.3 Αρχείο Χαρακτηριστικών

Το αρχείο των χαρακτηριστικών περιέχει όλες τις χειρονομίες και τις τιμές των χαρακτηριστικών που αντιστοιχούν σε καθεμία από αυτές.
Το αρχείο είναι της μορφής xml.

- **Xml**

Η ανάπτυξη του διαδικτύου και η ανάγκη που δημιουργήθηκε για ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων και πλατφορμών, έκανε επιτακτική τη δημιουργία ενός προτύπου ανταλλαγής και επεξεργασίας δεδομένων.
Το πρότυπο αυτό είναι η xml[7] και ξεκίνησε η ανάπτυξη της το 1996. Αργότερα, το 1998 εντάχθηκε στο W3C. Βασίζεται στη SGML (Standard Generalized Markup Language) και είναι ένα υποσύνολο της. Διατήρησε τα λειτουργικά της χαρακτηριστικά, αποβάλλοντας εκείνα που είναι δύσχρηστα στο προγραμματισμό της.
Τα έγγραφα XML (Extensible Markup Language) περιέχουν δεδομένα τα οποία περικλείονται από ετικέτες (tags). Με την XML μπορεί κάποιος να δημιουργήσει ένα σύνολο από ετικέτες που επιθυμεί και να τις χρησιμοποιήσει όπως αυτός επιθυμεί. Οι ετικέτες ορίζουν τη δομή και το νόημα των δεδομένων, δηλαδή καθορίζουν τι είναι δεδομένο. Αυτός ο καθορισμός κάνει δυνατή την επαναχρησιμοποίηση των δεδομένων με πολλούς τρόπους. Δηλαδή μπορούμε να χρησιμοποιήσουμε ένα σύστημα για να δημιουργήσουμε τα δεδομένα, να τα σημειώσουμε με ετικέτες xml και να επεξεργαστούμε αυτά τα δεδομένα σε άλλα συστήματα, ανεξάρτητα από το υλικό ή την πλατφόρμα του λειτουργικού συστήματος. Εξ' αιτίας αυτής της λειτουργίας η γλώσσα xml έχει γίνει από τους δημοφιλέστερους τρόπους μεταφοράς δεδομένων.
Η δομή της XML μοιάζει με αυτή της HTML. Ένα XML έγγραφο αποτελείται από tags, τα οποία είναι απαραίτητο να κλείνονται, π.χ. <angle>-26</angle>.
Επιτρέπονται άπειρα επίπεδα εμφωλευμένων tags. Επίσης απαγορεύεται να ξεκινούν οι ετικέτες με την λέξη 'XML' , είτε είναι γραμμένη στα πεζά είτε στα κεφαλαία.

```
<Six>6
<angle>-26</angle>
<ratio>1.20</ratio>
<nodef>1</nodef>
<comp>45</comp>
</Six>
```

3.3.1 *Παράδειγμα δομής εγγράφου XML*

Στο παράδειγμα 3.3.1 μπορούμε να δούμε τα δομικά στοιχεία που απαρτίζουν ένα έγγραφο xml. Αυτά τα στοιχεία είναι:

- Element: Το βασικό δομικό στοιχείο του εγγράφου. Το στοιχείο βρίσκεται ανάμεσα σε δυο tags που ανοίγουν και κλείνουν, π.χ. <angle>-26</angle>.

[7] ιστότοπος:
http://www.teiser.gr/icd/staff/tsimpiris/files/DBII%20PRESENTATION/13_xml_presentation.pdf

- Attribute: Είναι η ιδιότητα του στοιχείου element, π.χ. <book author="John">XML </book>. Επειδή η χρήση attributes κάνει δυσανάγνωστη τη μορφή του εγγράφου, δεν την χρησιμοποιήσαμε στο έγγραφο xml της εργασίας.

Entity: Οι οντότητες είναι αλφαριθμητικά που χρησιμοποιούνται ως συντομογραφίες άλλων αλφαριθμητικών.
<!Entity message "Welcome">
Με αυτόν τον τρόπο, όταν γράφουμε &message θα ισοδυναμεί με "Welcome" .
Το στοιχείο αυτό δεν είναι απαραίτητο στη περίπτωση μας.

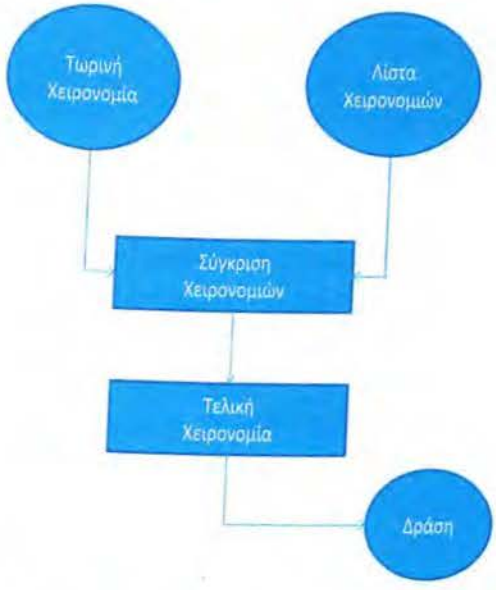
Τα πλεονεκτήματα που μα προσφέρει η γλώσσα XML είναι το κοινό πρότυπο μεταξύ διαφορετικών πλατφορμών, η ευελιξία στη δομή καθώς ο καθένας δημιουργεί όσες και όποιες ετικέτες επιθυμεί. Επιπλέον, τα αρχεία που κατασκευάζονται είναι ευανάγνωστα και μπορεί να γίνει αποθήκευση σε ASCII κείμενο. Επιπλέον οι περισσότερες εφαρμογές υποστηρίζουν την εξαγωγή και εισαγωγή στοιχείων από έγγραφο xml.

Μπορούμε να δούμε και μερικούς τομείς στους οποίους γίνεται χρήσης της XML, όπως:

- Επιχειρηματικά δεδομένα (πωλήσεις, τιμολόγια, παραγγελίες).
- Στοιχεία από βάσεις δεδομένων (όλα τα σύγχρονα RDBMS υποστηρίζουν εξαγωγή και εισαγωγή από XML).
- Δημιουργία γλωσσών σήμανσης σχετικά με έναν τομέα.
- Οικονομικά δεδομένα (οικονομικές και λογιστικές εφαρμογές).
- Human Resources XML (HR-XML).

3.4 Σύστημα Σύγκρισης

Στο στάδιο της σύγκρισης χαρακτηριστικών στόχος είναι η σύγκριση της τωρινής χειρονομίας μα μια λίστα προκαθορισμένων χειρονομιών. Όπως έχουμε αναφέρει η λίστα είναι ένα xml αρχείο το οποίο περιέχει όλα τα χαρακτηριστικά που είναι απαραίτητα για τον καθορισμό μιας χειρονομίας. Αν τα χαρακτηριστικά της χειρονομίας που πραγματοποιούμε ταιριάζουν με τα χαρακτηριστικά που βρίσκονται στη λίστα, τότε αναγνωρίζεται ποια χειρονομία πραγματοποιούμε. Παρακάτω παρουσιάζουμε σχηματικά το σύστημα σύγκρισης.



3.4.1 *Σχηματική διάταξη συστήματος σύγκρισης*

Όπως γίνεται κατανοητό από το παραπάνω σχήμα, ο χρήστης πρέπει να κάνει ακριβώς την ίδια χειρονομία με αυτή που βρίσκεται στη λίστα για να ταυτοποιηθεί η χειρονομία, πράγμα δύσκολο . Για τον λόγο αυτό αυξάνουμε το εύρος των τιμών των χαρακτηριστικών εφαρμόζοντας τιμές κατωφλίου στα χαρακτηριστικά. Με τον τρόπο αυτό η σύγκριση δίνει περισσότερη ευελιξία στο χρήστη της εφαρμογής.

Σε αυτό το σημείο πρέπει να εξηγήσουμε τον τρόπο με τον οποίο δουλεύει το σύστημα σύγκρισης.

Οι χειρονομίες τις οποίες έχουμε καθορίσει σ το xml αρχείο, φορτώνονται πριν ξεκινήσει ο αλγόριθμος. Από την xml μορφή που έχουν τα χαρακτηριστικά τα μετατρέπουμε -με τη χρήση της κλάσης Class ,την οποία δημιουργούμε- σε τύπους δεδομένων ικανούς να τους αναγνωρίσει και να επεξεργαστεί η γλώσσα C ++ , στην οποία αναπτύσσουμε τον αλγόριθμο.

Αφού φορτωθούν οι πληροφορίες, ο αλγόριθμος κατευθύνεται στο κυρίως πρόγραμμα. Σε αυτό το στάδιο επεξεργάζεται την εικόνα και λαμβάνει τα χαρακτηριστικά της χειρονομίας που πραγματοποιεί ο χρήστης. Οι πληροφορίες των τωρινών χειρονομιών αποθηκεύονται σε μια δομή δεδομένων με όνομα Actual Gesture. Η data structure παρέχει πλέον τις πληροφορίες της χειρονομίας που πραγματοποιεί κάθε φορά ο χρήστης της εφαρμογής. Στη συνέχεια οι πληροφορίες για τα χαρακτηριστικά της τωρινής χειρονομίας ελέγχονται μια προς μια με τις πληροφορίες που έχουμε καταγράψει στην xml λίστα. Για κάθε χαρακτηριστικό (feature) ελέγχουμε τη τιμή του. Το αποτέλεσμα της σύγκρισης είναι η τελική χειρονομία. Η τελική χειρονομία καθορίζει και την δράση η οποία θα πραγματοποιηθεί από το συνολικό σύστημα. Τον τρόπο με τον οποίο θα υλοποιηθεί η δράση θα τον δούμε στο επόμενο κεφάλαιο.

Κεφάλαιο 4^0
Εφαρμογή Αλγορίθμου

Ο αλγόριθμος που έχουμε αναπτύξει είναι ικανός να αναγνωρίζει τις χειρονομίες που πραγματοποιεί ο χρήστης. Σε αυτό το κεφάλαιο θα μελετήσουμε τα συστήματα επικοινωνίας, διακοπών και χρονισμού του μικροελεγκτή Avr Atmega16.

4.1 Περιγραφή

Σκοπός του σταδίου αυτού είναι η σωστή χρήση του μικροελεγκτή ώστε κάθε χειρονομία να υλοποιεί μια διαφορετική κίνηση. Για τον λόγο αυτό οι χειρονομίες ταξινομήθηκαν και απέκτησαν έναν αριθμό, από το 0 έως το 10 , ο οποίος τις χαρακτηρίζει.

Ο χρήστης της εφαρμογής πραγματοποιεί μια χειρονομία η οποία αξιολογείται μέσω του συστήματος σύγκρισης. Το αποτέλεσμα του συστήματος είναι η τελική χειρονομία ή αλλιώς ένας αριθμός μεταξύ 0 και 10 που αποτελεί την πληροφορία η οποία θα μεταδοθεί σειριακά και ασύρματα στο μικροελεγκτή.

Την πληροφορία αυτή πρέπει να την αξιοποιήσει ο μικροελεγκτής για να αποφασίσει την ενέργεια που πρέπει να πραγματοποιήσει και η ενέργεια μεταφράζεται σε κίνηση των σέρβο κινητήρων, η λειτουργία των οποίων παρέχει την δυνατότητα στην μαριονέτα να κουνάει τα χέρια και τα πόδια, καθώς και να πραγματοποιεί συνδυαστικές κινήσεις. Γίνεται σαφές ότι ο αλγόριθμος αντιλαμβάνεται έντεκα διαφορετικές χειρονομίες. Ο αριθμός των κινήσεων της μαριονέτας μπορεί να είναι μεγαλύτερος, καθώς δίνεται η δυνατότητα συνδυασμού κινήσεων. Με την ιδιότητα αυτή κάθε χειρονομία μπορεί να αντιστοιχεί σε λειτουργία παραπάνω του ενός κινητήρα. Δηλαδή μια χειρονομία μπορεί να θέτει σε λειτουργία δύο ή και τρείς κινητήρες.

Ο αριθμός των χειρονομιών μπορεί να είναι μεγαλύτερος των έντεκα εάν ο χρήστης επιθυμεί κάτι τέτοιο. Στη περίπτωση ανάπτυξης εφαρμογής που απαιτεί πολύ μεγάλο αριθμό χειρονομιών ίσως ο αλγόριθμος που περιγράψαμε να μην λειτουργήσει ικανοποιητικά, καθώς πρέπει να ληφθούν περισσότερα των τεσσάρων χαρακτηριστικών ώστε να αναγνωρισθεί μια χειρονομία. Υπάρχουν μοντέλα, όπως το Hidden Markov model, που χρησιμοποιούνται στην παραπάνω περίπτωση.

Ο μεγάλος αριθμός χειρονομιών δεν είναι πάντα πλεονέκτημα για μια εφαρμογή, καθώς δεν είναι φιλική προς τον χρήστη. Είναι δύσκολο για τον χρήστη να θυμάται την ενέργεια που πραγματοποιεί κάθε χειρονομία και η εφαρμογή γίνεται δυσλειτουργική. Επιπλέον παράγοντας είναι ο υπολογιστικός χρόνος ο ποίος πρέπει να κρατηθεί σε χαμηλά επίπεδα. Οι ενέργειες που πραγματοποιούμε έχουν σκοπό την ορθή λειτουργία της εφαρμογής.

4.2 Οι Χειρονομίες ως Διακοπές

Όπως έχουμε προαναφέρει χρειαζόμαστε ένα λογισμικό για να γράψουμε, να μεταγλωττίσουμε και να φορτώσουμε το πρόγραμμα στον μικροελεγκτή. Το λογισμικό που επιλέξαμε είναι το WinAvr, για τους λόγους που έχουμε ήδη επισημάνει σε προηγούμενο κεφάλαιο. Για την ανάπτυξη του προγράμματος θα χρειαστούμε το Avr Studio από την Atmel, ελεύθερο πρόγραμμα το οποίο οργανώνει όλα τα αρχεία που χρειάζονται για να γίνει η μεταγλώττιση του κώδικα, ώστε να τρέξει τον κώδικα ο μικροελεγκτής. Για να γίνει η μεταγλώττιση το Avr Studio ενσωματώνει το λογισμικό WinAvr.
Στο Avr Studio γράφουμε τον κώδικα σε γλώσσα C, το μεταγλωττίζουμε και το μεταφέρουμε στον μικροελεγκτή. Ο κώδικας καθορίζει τις ενέργειες που θα υλοποιούν οι χειρονομίες.

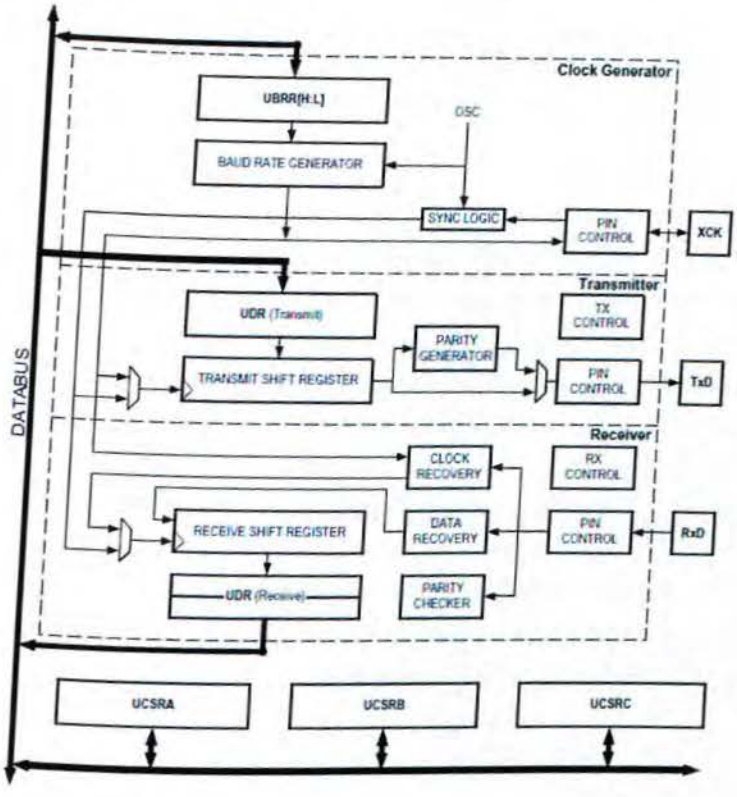
- **Σύστημα USART του Atmega16**

Ο μικροελεγκτής πρέπει συχνά να ανταλλάσει δεδομένα με άλλους μικροελεγκτές ή περιφερειακές συσκευές. Τα δεδομένα μπορούν νε μεταδοθούν χρησιμοποιώντας τεχνικές παράλληλης ή σειριακής μετάδοσης. Με την παράλληλη τεχνική ένα ολόκληρο byte δεδομένων μπορεί να μεταδοθεί από την συσκευή εκπομπής προς την συσκευή λήψης. Παρότι αυτό είναι αποτελεσματικό από την άποψη χρόνου, απαιτεί οχτώ διαφορετικές γραμμές για την μετάδοση της πληροφορίας.
Ο μικροελεγκτής Atmega16 είναι εξοπλισμένος με το σειριακό υποσύστημα επικοινωνίας USART, το οποίο παρέχει full duplex επικοινωνία μεταξύ του πομπού και του δέκτη. Αυτό επιτυγχάνεται εξοπλίζοντας τον Atmega16 με διαφορετικό υλικό (hardware) για τον πομπό και τον δέκτη. Τυπικά το USART χρησιμοποιείτε για ασύγχρονη επικοινωνία. Στα ασύγχρονα σειριακά συστήματα επικοινωνίας -όπως το USART- δεν υπάρχει κοινό ρολόι χρονισμού μεταξύ πομπού και δέκτη, αλλά bits πλαισίου τοποθετούνται στην αρχή και στο τέλος ενός byte δεδομένων για να διατηρηθεί ο συγχρονισμός μεταξύ πομπού και δέκτη. Αυτά τα bits προειδοποιούν το δέκτη ότι ένα byte δεδομένων εισόδου έχει φτάσει, και επίσης σηματοδοτούν την ολοκλήρωση της λήψης του byte δεδομένων. Ο ρυθμός μετάδοσης των δεδομένων για ένα ασύγχρονο σειριακό σύστημα είναι χαμηλότερος σε σύγκριση με ένα σύγχρονο σειριακό σύστημα επικοινωνίας, αλλά απαιτεί μια μόνο γραμμή μεταξύ του πομπού και του δέκτη.

Το σύστημα USART του Amega16 είναι αρκετά ευέλικτο, δίνοντας την ευκαιρία στο χρήστη να επιλέγει τον ρυθμό μετάδοσης δεδομένων. Επίσης μπορεί να επιλέγει το πλάτος των δεδομένων, από 5 έως 9 bits, με ένα ή δύο stop bits. Επιπλέον, ο δέκτης είναι εξοπλισμένος με υλικό για τον έλεγχο των parity bit. Ένα parity bit επιτρέπει την εύρεση ενός λάθους bit σε ένα byte δεδομένων.

- **Επισκόπηση Συστήματος**

Στο μπλοκ διάγραμμα του συστήματος USART, που βλέπουμε παρακάτω, έχουμε τέσσερα κύρια μέρη: τη γεννήτρια ρολογιού, το υλικό μετάδοσης, το υλικό λήψης και τους τρείς καταχωρητές ελέγχου (UCSRA,UCSRB,UCSRC).



4.2.1 *Μπλοκ Διάγραμμα συστήματος USART*

I. **USART γεννήτρια ρολογιού:** Η γεννήτρια ρολογιού του USART παρέχει την πηγή ρολογιού για το σύστημα και θέτει τον ρυθμό των bit σε αυτό. Το baud rate (ρυθμό bit) ξεχωρίζει από την πηγή ρολογιού. Το baud rate στο USART σύστημα ορίζεται από τον παρακάτω τύπο.

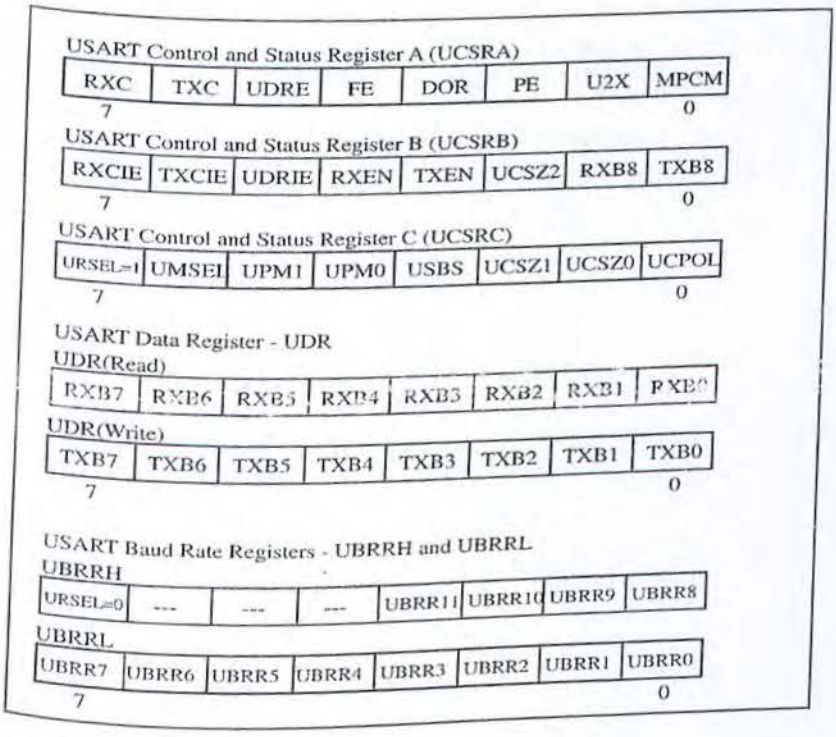
Baud rate = (συχνότητα ρολογιού μικροελεγκτή)/ 2*(UBRR+1).
Όπου UBRR: είναι το περιεχόμενο του καταχωρητή UBRR (0-4095).

II. **Εκπομπός USART:** Ο εκπομπός αποτελείται από τον καταχωριτή Transmit Shift Register, στον οποίο φορτώνονται τα δεδομένα τα οποία θέλουμε να μεταδοθούν. Η φόρτωση γίνεται μέσω του καταχωρητή UDR. Τα start και stop bit πλαισίου τοποθετούνται αυτόματα στο δεδομένο μέσα στον καταχωρητή Transmit Shift Register. Τα δεδομένα βγαίνουν ένα bit τη φορά από τον καταχωρητή Transmit Shift Register μέσω του TxD pin με προκαθορισμένο ρυθμό. Ο USART εκπομπός είναι εξοπλισμένος με δύο flags κατάστασης,

το UDRE και το TXC. Το UDRE είναι άσσος όταν ο buffer του εκπομπού είναι άδειος. Το TXC γίνεται λογικό ένα όταν ο καταχωρητής Transmit Shift Register είναι άδειος.

III. **Δέκτης USART:** Είναι σχεδόν πανομοιότυπος με τον εκπομπό USART, εκτός από την κατεύθυνση της ροής των δεδομένων, η οποία είναι αντίστροφη. Τα δεδομένα λαμβάνονται ένα bit τη φορά μέσω του RxD pin σε καθορισμένο baud rate. Επίσης ο δέκτης USART είναι εξοπλισμένος με flag RXC, το οποίο γίνεται λογικό ένα όταν δεδομένα που δεν έχουν διαβαστεί υπάρχουν στο buffer του δέκτη.

IV. **Καταχωρητές USART:** Οι καταχωρητές είναι οι μνήμες εκείνες που μπορούμε να επέμβουμε και να ενεργοποιήσουμε το σύστημα USART. Έχουμε ήδη μελετήσει το σκοπό των καταχωρητών UDR UBRR(H:L) και θα δούμε και τους υπόλοιπους καταχωρητές του συστήματος. Το URSEL bit καθορίζει σε ποιον καταχωρητή έχουμε πρόσβαση, εάν URSEL=1 τότε ο UCSRC καταχωρητής ενεργοποιείτε, αλλιώς ενεργοποιείτε ο UBRRH καταχωρητής.
Ο UCSRA περιέχει τα RXC, TXC, UDRE bit, η λειτουργία των οποίων έχει αναφερθεί.
Ο UCSRB περιέχει τα bits για ενεργοποίηση του δέκτη και του πομπού (RXEN καιTXEN αντίστοιχα). Τα παραπάνω bits αποτελούν διακόπτες για τον δέκτη και το πομπό. Ο UCSRB καταχωρητής περιέχει και το UCSZ2 bit το οποίο καθορίζει το μέγεθος του χαρακτήρα του δεδομένου. Το UCSZ[1:0] bit περιλαμβάνεται στο UCSRB και UCSRC καταχωρητή.
Ο UCSRC επιτρέπει στο χρήστη να προσαρμόζει τα χαρακτηριστικά του δεδομένου κατά τη βούληση του. Να τονιστεί ότι και ο πομπός και ο δέκτης πρέπει να ρυθμιστούν με τα ίδια χαρακτηριστικά των δεδομένων για σωστή μετάδοση.
Παρακάτω βλέπουμε τους καταχωρητές του συστήματος USART.


USART Control and Status Register A (UCSRA)

RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
7							0

USART Control and Status Register B (UCSRB)

RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
7							0

USART Control and Status Register C (UCSRC)

URSEL=1	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
7							0

USART Data Register - UDR

UDR(Read)

RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0

UDR(Write)

TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
7							0

USART Baud Rate Registers - UBRRH and UBRRL

UBRRH

URSEL=0	---	---	---	UBRR11	UBRR10	UBRR9	UBRR8

UBRRL

UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0
7							0

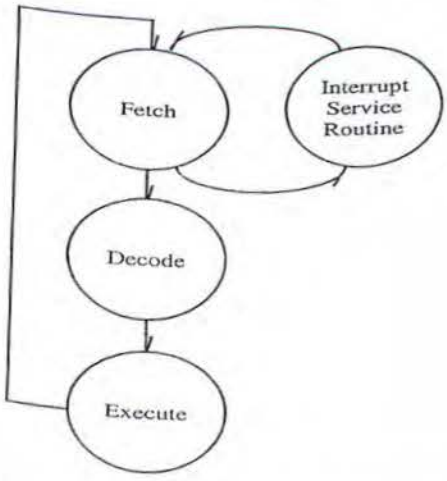
4.2.2 Καταχωρητές του συστήματος *USART*

- **Σύστημα Διακοπών**

Ένας μικροελεγκτής συνήθως εκτελεί εντολές με μια προκαθορισμένη διαδικασία. Παρόλα αυτά, ο μικροελεγκτής πρέπει να είναι εξοπλισμένος να αντιμετωπίζει μη προγραμματισμένα γεγονότα, τα οποία μπορούν να συμβούν εντός και εκτός του μικροελεγκτή. Για να διαχειριστεί τέτοιου είδους γεγονότα, ο μικροελεγκτής είναι εφοδιασμένος με σύστημα διακοπών.
Το σύστημα διακοπών επιτρέπει να ανταποκρίνεται ο μικροελεγκτής σε υψηλής προτεραιότητας γεγονότα, τα οποία μπορούν να σχεδιάζονται ότι θα συμβούν αλλά δεν ξέρουμε το πότε θα συμβούν. Όταν συμβαίνει ένα γεγονός διακοπής, ο μικροελεγκτής ολοκληρώνει τις εντολές που πραγματοποιεί και μετά μεταβαίνει στο πρόγραμμα ελέγχου της διακοπής και πραγματοποιεί τις εντολές που περιέχονται στο πρόγραμμα. Η ISR (Interrupt Service Routine) είναι η συνάρτηση στην οποία οργανώνονται οι εντολές του προγράμματος που εκτελείτε όταν υπάρχει διακοπή. Κάθε είδος διακοπής έχει τη δικιά του ρουτίνα ISR. Μόλις η ρουτίνα διακοπής ολοκληρωθεί, ο μικροελεγκτής συνεχίζει να εκτελεί την διαδικασία που πραγματοποιούσε πριν τη διακοπή.

- **Σύστημα Διακοπών του Atmega16**

Ο μικροελεγκτής Atmega16 είναι εξοπλισμένος να χειρίζεται εικοσιένα διαφορετικές πηγές διακοπών, τρείς από τις οποίες έχουν προέλευση από το εξωτερικές πηγές, ενώ οι εναπομείναντες δεκαοκτώ διακοπές υποστηρίζουν την λειτουργία των περιφερικών υποσυστημάτων του μικροελεγκτή.
Όταν συμβαίνει μια διακοπή, ο μικροελεγκτής ολοκληρώνει την εντολή στην οποία βρίσκεται και αποθηκεύει την επόμενη εντολή στη στοίβα, στη συνέχεια αρχίζει να εκτελεί τις εντολές που βρίσκονται μέσα στη ρουτίνα διακοπής, η οποία ενεργοποιείτε λόγω κάποιας πηγής διακοπής. Επίσης κλείνει το σύστημα διακοπών για να αποτρέψει περισσότερες διακοπές να συμβούν. Η εκτέλεση της ρουτίνας ISR γίνεται με το φόρτωση της αρχικής διεύθυνσης της ρουτίνας ISR -καθορισμένη για συγκεκριμένη διακοπή- στο μετρητή προγράμματος (program counter). Μετά αρχίζει να εκτελείτε η ISR μέχρι να εμφανιστεί η εντολή reti, η οποία ενημερώνει για το τέλος των εντολών διακοπής και επιστρέφει στην εκτέλεση των εντολών του κυρίως προγράμματος.



4.2.3 *Σχηματική περιγραφή εκτέλεσης διακοπής από τον μικροελεγκτή*

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

4.2.4 *Οι πηγές διακοπών του Atmel Atmega16 μικροελεγκτή*

Όπως προαναφέραμε κάθε χειρονομία αντιστοιχεί σε έναν αριθμό από το 0 έως το 10, ο οποίος είναι η πληροφορία που μεταδίδεται σειριακά και ασύρματα στον μικροελεγκτή. Ο μικροελεγκτής αντιλαμβάνεται αυτήν την πληροφορία εισόδου ως εσωτερική διακοπή, η πηγή της οποίας είναι το RXC, δηλαδή ελέγχει αν τα δεδομένα λήψης έχουν ολοκληρωθεί. Στη περίπτωση που δεν έχουμε δεδομένα για λήψη ο μικροελεγκτής εκτελεί τον κώδικα που βρίσκεται στο κυρίως πρόγραμμα. Στην αντίθετη περίπτωση ενεργοποιείτε η ρουτίνα διακοπής.

Η ρουτίνα διακοπής περιέχει τον κώδικα ο οποίος ελέγχει τα δεδομένα που αποστέλλονται σειριακά και ασύρματα στον καταχωρητή UDR του υποσυστήματος USART του Atmega16. Εάν τα δεδομένα είναι κάποιος αριθμός από 0 έως 10 σε μορφή χαρακτήρα (string), τότε εκτελείται μια λειτουργία η οποία αναφέρεται στην κίνηση των σερβοκινητήρων που είναι συνδεδεμένοι με τον μικροελεγκτή.

4.3 Hardware

Οι πληροφορίες έχουν μεταφερθεί στον μικροελεγκτή και θέλουμε να μεταφραστούν σε λειτουργία των κινητήρων. Ο Atmega16 παρέχει δυνατότητες σύνδεσης εξωτερικού υλικού, στην εργασία μας χρησιμοποιούμε rc σέρβο κινητήρες. Τα πλεονεκτήματα και ο τρόπος λειτουργίας των σερβοκινητήρων rc έχουν αναφερθεί στο πρώτο κεφάλαιο. Στην ενότητα αυτή παρουσιάσουμε το σύστημα χρονισμού του μικροελεγκτή, τον τρόπο σύνδεσης των περιφερειακών και τον σκοπό τους στη δημιουργία της εργασίας.

- **Σύστημα Χρονισμού του Atmega16**

Ένας από τους σημαντικότερους λόγους χρήσης των μικροελεγκτών στα ενσωματωμένα συστήματα είναι η ικανότητα τους να χειρίζονται εργασίες που συνδέονται με το χρόνο. Σε μια απλή εφαρμογή μπορούμε να προγραμματίσουμε το μικροϋπολογιστικό σύστημα να ανάβει και να σβήνει μια εξωτερική συσκευή σε προκαθορισμένο χρόνο. Σε μια πολυπλοκότερη εφαρμογή μπορούμε να παράγουμε ψηφιακές κυματομορφές με πλάτος που μεταβάλλεται (PWM) για να ελέγχουμε την ταχύτητα ενός κινητήρα. Παρακάτω κάνουμε μια σύντομη αναφορά στους όρους που θα χρησιμοποιήσουμε για την περιγραφή του συστήματος χρονισμού και του PWM.

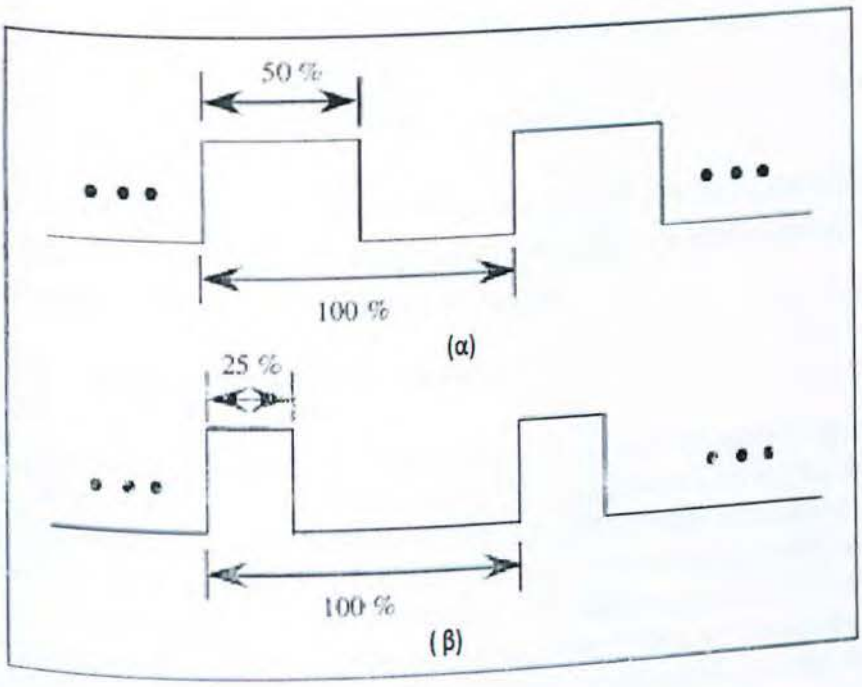
Συχνότητα: Υποθέτουμε ότι έχουμε ένα σήμα της μορφής x(t) το οποίο επαναλαμβάνει τον εαυτό του. Ονομάζουμε αυτό το σήμα περιοδικό με περίοδο Τα εάν ικανοποιεί την παρακάτω σχέση: $x(t) = x(t+T)$.

Για να μετρήσουμε τη συχνότητα ενός περιοδικού σήματος, μετράμε πόσες φορές επαναλαμβάνει τον εαυτό του σε χρονική περίοδο ενός δευτερολέπτου. Η μονάδα της συχνότητας είναι το Hertz, ή οι κύκλοι ανά δευτερόλεπτο. Για παράδειγμα ημιτονικό σήμα 60 Hz σημαίνει ότι σε έναν ολόκληρο κύκλο επαναλαμβάνει τον εαυτό του 60 φορές το δευτερόλεπτο.

Περίοδος: Το αντίστροφο της συχνότητας είναι η περίοδος. Εάν κάποιο γεγονός συμβαίνει με συχνότητα 1Hz, η περίοδος του γεγονότος είναι 1 second. Για να βρούμε την περίοδο εφαρμόζουμε την παρακάτω σχέση $T=1/f$, όπου f είναι η συχνότητα . Η περίοδος και η συχνότητα ενός σήματος χρησιμοποιούνται –σε ένα ενσωματωμένο σύστημα- για να καθορίσουν χρονικούς περιορισμούς.

Duty Cycle: Σε πολλές εφαρμογές, οι περιοδικοί παλμοί χρησιμοποιούνται για να ελέγξουν σήματα. Ένα παράδειγμα είναι η χρήση περιοδικού παλμού για τον έλεγχος ενός σέρβο κινητήρα. Για να πετύχουμε τον έλεγχο της θέσης, κατεύθυνσης αλλά και της ταχύτητας, χρησιμοποιείται περιοδικός παλμός σήματος με duty cycle που μεταβάλλεται στο χρόνο.

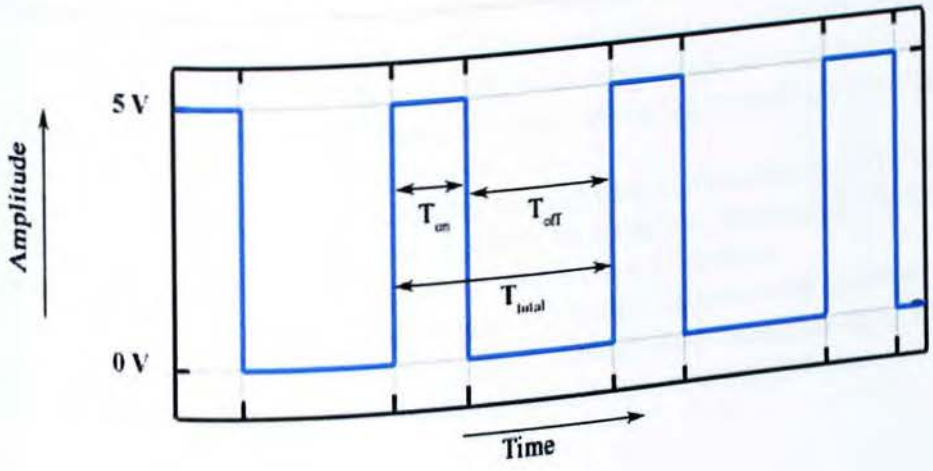
Duty cycle ορίζεται ως το ποσοστό μιας περιόδου που το σήμα "υψηλό". Στην παρακάτω εικόνα βλέπουμε duty cycle ποσοστού 50% (α) και 25% (β).



4.3.1 *Duty cycle 50% (α) και duty cycle 25% (β)*

PWM

Το Pulse Width Modulation είναι τεχνική που χρησιμοποιείται στα συστήματα ελέγχου. Εφαρμόζεται σε εφαρμογές οι οποίες περιλαμβάνουν έλεγχο ταχύτητας, έλεγχο ισχύος, έλεγχο μετρήσεων και επικοινωνίας.
Η διαμόρφωση του πλάτους παλμού επιτυγχάνεται με τη βοήθεια ενός τετραγωνικού παλμού του οποίου το duty cycle αλλάζει για να πάρουμε μια μεταβαλλόμενη τάση εξόδου, ως αποτέλεσμα της μέσης τιμής της κυματομορφής.
Θεωρούμε το τετραγωνικό παλμό της εικόνα 4.3.2



4.3.2 *Τετραγωνικός παλμός*

Στην εικόνα 4.3.2 ο τετραγωνικός παλμός έχει περίοδο $T = T_{on}+T_{off}$, το duty cycle του τετραγωνικού κύματος είναι:

$$D = T_{on} / (T_{on}+T_{off}) = T_{on} / T_{total}$$

Η τάση εξόδου είναι: $Vout = D * Vin$ ή $Vout = (T_{on} / T_{total}) * Vin$

Άρα η τάση εξόδου εξαρτάται από το T_{on}. Μεταβάλλοντας το T_{on} μεταβάλλεται και η τάση εξόδου. Η μέγιστη τιμή που μπορεί να αποκτήσει η τάση εξόδου (Vout) είναι ίση με το Vin, δηλαδή όταν το κλάσμα είναι ίσο με τη μονάδα.

- **Επισκόπηση Συστήματος Χρονισμού**

Η λειτουργία του συστήματος περιστρέφεται γύρω από την χρήση ενός σήματος ελέγχου, το οποίο αποτελεί το σήμα του ρολογιού του μικροελεγκτή. Το σύστημα ρολογιού χρησιμοποιείτε για να ενημερώνονται τα περιεχόμενα ενός καταχωρητή ειδικού σκοπού, του free- running μετρητή. Η δουλειά ενός free- running μετρητή είναι να μετράει ,δηλαδή να αυξάνει την τιμή του, κάθε φορά που βλέπει μια ακμή ανόδου (ή καθόδου) ενός σήματος ρολογιού. Εάν ένα ρολόι τρέχει με ρυθμό 2 MHz, ο free-running μετρητής θα μετράει κάθε 0.5 μs. Οι υπόλοιπες μονάδες που συνδέονται με τον χρονιστή, αναφέρονται στα δεδομένα του free-running μετρητή για να πραγματοποιήσουν εργασίες σχετικές με το χρόνο, όπως: μέτρηση περιόδων, σύλληψη χρονικών γεγονότων και δημιουργία σημάτων σχετικών με το χρόνο.
Για ενέργειες που σχετίζονται σήματα εισόδου, ο μικροελεγκτής διαθέτει υλικά στοιχεία χρονιστών που ανιχνεύουν τις αλλαγές των λογικών σημάτων σε ένα ή παραπάνω pins εισόδου. Τέτοια στοιχεία βασίζονται σε έναν free-running μετρητή για να συλλάβουν εξωτερικά γεγονότα. Μπορούμε να αξιοποιήσουμε αυτή την ιδιότητα μετρώντας την περίοδο ενός σήματος εισόδου, το πλάτος του παλμού και να υπολογίσουμε τον χρόνο μεταξύ των αλλαγών ενός λογικού σήματος.
Για χρονικές συναρτήσεις εξόδου, ο μικροελεγκτής χρησιμοποιεί έναν συγκριτή, μετρητή free-running, λογικούς διακόπτες, καταχωρητές ειδικού σκοπού για να παράγει σήματα σχετικά με τον χρόνο σε ένα ή παραπάνω pins εξόδου. Ο συγκριτής ελέγχει αν ταιριάζει η τιμή του μετρητή free –running με την τιμή που έχει τοποθετήσει ο προγραμματιστής στους καταχωρητές ειδικού σκοπού. Η διαδικασία έλεγχου εκτελείται σε κάθε κύκλο ρολογιού, όταν προκύπτει ισότητα, πραγματοποιείται μια προγραμματισμένη λογική αλλαγή σε ορισμένη -από τον προγραμματιστή- πόρτα εξόδου του μικροελεγκτή. Χρησιμοποιώντας αυτήν την ιδιότητα κάποιος μπορεί να παράγει έναν παλμό με επιθυμητό πλάτος, ένα PWM σήμα για τον έλεγχο ενός κινητήρα ή απλούστερες εφαρμογές που έχουν άμεση σχέση με το χειρισμό του χρόνου. Επιπλέον μπορούμε να σύστημα χρονισμού για να μετρήσουμε το πλάτος του παλμού ενός μη περιοδικού σήματος.
Ένας δεύτερος σκοπός του συστήματος χρονισμού είναι να παράγει σήματα για να ελέγχει εξωτερικές συσκευές, όπως τον rc servo motor της εργασίας. Οι συσκευές συνδέονται φυσικά με τον μικροελεγκτή στο σύστημα θυρών I/O που διαθέτει.
Οι καταχωρητές του συστήματος χρονισμού του Atmega16 είναι οι δύο 8-bit καταχωρητές Timer 0 και Timer 2, καθώς και ο 16-bit Timer1. Αποτελούν απαραίτητα στοιχεία του συστήματος χρονισμού για να εκτελεί τους σκοπούς που προαναφέρθηκαν.

- **Θύρες I/O**

Ο μικροελεγκτής Atmega16 για να μπορεί να ανταλλάσει δεδομένα με εξωτερικές συσκευές διαθέτει θύρες εισόδου- εξόδου. Οι ακροδέκτες είναι οργανωμένοι σε τέσσερις θύρες, τις PORTA,PORTB,PORTC,PORTD. Κάθε pin μπορεί να ρυθμιστεί σαν είσοδος ή σαν έξοδος, ανεξάρτητα από τους υπόλοιπους ακροδέκτες της θύρας.
Κάθε ακροδέκτης της πόρτας αποτελείται από 3 bit καταχωρητών τους DDxn, PORTxn, και PINxn. Οι καταχωρητές αυτοί είναι 8-bit, και κάθε bit αναφέρεται στον αντίστοιχο ακροδέκτη της θύρας. Το DDxn bit στον DDRx καταχωρητή επιλέγει την κατεύθυνση του συγκεκριμένου pin. Εάν το DDxn είναι λογικός άσσος, το Pxn ρυθμίζεται σαν pin εξόδου. Εάν το DDxn είναι λογικό μηδέν, το Pxn θεωρείτε pin εισόδου.
Το Portxn περιέχει τις τιμές εξόδου των ακροδεκτών που είναι ρυθμισμένοι ως έξοδοι. Εάν η PORTxn είναι λογικό ένα όταν ρυθμίζεται ο ακροδέκτης σαν pin εισόδου, ενεργοποιείτε η pull-up αντίσταση. Για να αλλάξουμε την κατάσταση στη pull-up αντίσταση, πρέπει η PORTxn να γίνει λογικό μηδέν ή το pin να ρυθμιστεί ως pin εξόδου. Εάν στη PORTxn γράφουμε λογικό ένα τότε το pin οδηγείτε σε κατάσταση υψηλή, ενώ όταν στη PORTxn γράφουμε λογικό 0 το pin οδηγείτε σε κατάσταση χαμηλή.
Ο PINx περιέχει τις τιμές εισόδου των ακροδεκτών που είναι ρυθμισμένοι ως είσοδοι - λογικό ένα για υψηλή τάση, λογικό μηδέν για χαμηλή τάση. Ο καταχωρητής αυτός είναι μόνο για ανάγνωση.
Παρακάτω βλέπουμε και τους DDRx, PORTx, PINx καταχωρητές μιας θύρας.

7	6	5	4	3	2	1	0
DDRx7	DDRx6	DDRx5	DDRx4	DDRx3	DDRx2	DDRx1	DDRx0

7	6	5	4	3	2	1	0
PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0

7	6	5	4	3	2	1	0
PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0

4.3.3 *Καταχωρητές DDRxn, PORTxn, PINxn*

Στην εργασία συνδέουμε τους κινητήρες με τους ακροδέκτες PD4,PD5,PD7 και PB3, ορίζοντας άσσο στα αντίστοιχα bit των καταχωρητών DDRD και DDRB. Οι παραπάνω ακροδέκτες αποτελούν pin εξόδου και μεταφέρουν στους σέρβο κινητήρες το pwm σήμα των αντίστοιχων χρονιστών. Το αποτέλεσμα είναι να κινούνται οι κινητήρες στη θέση που επιθυμούμε και με την κατάλληλη ταχύτητα.
Με την ολοκλήρωση αυτού του σταδίου η χειρονομία μεταφράζεται σε κίνηση της μαριονέτας, η οποία μπορεί να κινηθεί ανάλογα με τις χειρονομίες που πραγματοποιεί ο χρήστης της εφαρμογής. Σε περίπτωση που δεν πραγματοποιείτε καμία χειρονομία τότε η μαριονέτα παραμένει ακίνητη στην αρχική της κατάσταση.

Εφαρμογή: Στο κεφάλαιο αυτό παρουσιάσαμε τα στοιχεία του μικροελεγκτή που μας βοήθησαν να επικοινωνήσουμε σειριακά με τον ηλεκτρονικό υπολογιστή, αλλά και τα στοιχεία που μας επέτρεψαν τη σύνδεση των r/c σέρβο κινητήρων.

Όσον αφορά τη σειριακή επικοινωνία χρειαστήκαμε το υποσύστημα USART, το οποίο περιέχει ο μικροελεγκτής Atmega16. Εκεί ορίσαμε το baud rate = 9600, ένα stop bit, κανένα parity bit και μέγεθος δεδομένων 8-bit. Την επιθυμητή τιμή του baud rate την ορίζουμε στο καταχωρητή UBRR[H:L], τα χαρακτηριστικά του δεδομένου τα προσαρμόζουμε στο UCSRC καταχωρητή.

Εκμεταλλευόμαστε πλήρως και το σύστημα χρονισμού που μας παρέχει ο μικροελεγκτής. Χρησιμοποιούμε τους Timer0, Timer1, Timer2, για να ελέγξουμε τους σέρβο κινητήρες. Στον Timer0 έχουμε τις παρακάτω ρυθμίσεις: TCCR0= _BV(WGM00) | _BV(CS02) | _BV(COM01). Αυτό σημαίνει ότι λειτούργει σε phase correct PWM mode και η συχνότητα του υποσυστήματος χρονισμού είναι 25500 Hz. Για να κάνει κίνηση από 0 εως 90 μοίρες, ο κινητήρας πρέπει να έχει συχνότητα από 0 εως 2Hz και να οδηγήτε από σήμα συχνότητας 50Hz. Στον timer0 χρησιμοποιούμε το phase corect PWM mode και την συχνότητα του μικροελεγκτή τη βρίσκουμε από τον τύπο $f_{oc0PWM}=f_{clk\_IO}/(N*510)$, με f_{oc0PWM} = 50Hz και N=256. Στον timer1 και στον καταχωρητή OCR1(AB) τοποθετούμε την επιθυμητή τιμή ακολουθώντας την σχέση TOP = clk_IO / (2 x N x Desired-Frequency) , όπου N =256 και clk_IO= 6,528 MHz. Με την ίδια λογική αρχικοποιούμε και τους υπόλοιπους Timers, με τη μόνη διαφορά ότι ο Timer1 είναι 16-bit. Με τον τρόπο αυτό ελέγχουμε τη θέση του κινητήρα.

Στόχος είναι η σωστή χρήση των καταχωρητών και των συστημάτων επικοινωνίας του μικροελεγκτή, ώστε να πετύχουμε το καλύτερο αποτέλεσμα με το μικρότερο υπολογιστικό κόστος.

ΚΕΦΑΛΑΙΟ 5^{ο}
Παρουσίαση Κώδικα

Στο κεφάλαιο αυτό θα αναφερθούμε στον αλγόριθμο που φτιάξαμε, κάνοντας παρουσίαση κάποιων σημαντικών τμημάτων του κώδικα. Η επεξήγηση της λειτουργίας του κώδικα φανερώνει τα βήματα που μας οδήγησαν στην επίτευξη του τελικού αποτελέσματος. Στη συνέχεια γίνεται παρουσίαση τμημάτων από το κώδικα για την αναγνώριση του ανθρώπινου χεριού και λήψης των επιθυμητών χαρακτηριστικών, από το αρχείο xml, καθώς και από το κώδικα που φορτώνουμε στο μικροελεγκτή.

5.1 Κώδικας αναγνώρισης χεριού και λήψης χαρακτηριστικών:

```
CvCapture* capture=cvCaptureFromCAM(0);
if(!cvQueryFrame(capture))
{printf("chk camera ");}
CvSize sz=cvGetSize(cvQueryFrame(capture));
IplImage* src=cvCreateImage(sz,8,3);
```

Το πρώτο πράγμα που κάνουμε στο κυρίως πρόγραμμα είναι να ενεργοποιήσουμε την κάμερα η οποία είναι συνδεδεμένη στον υπολογιστή που χρησιμοποιούμε. Κάνουμε έλεγχο αν η κάμερα λαμβάνει εικόνα, αν δεν λειτουργεί εμφανίζει μήνυμα λάθους, και δημιουργούμε μια καινούργια εικόνα με το όνομα src, την οποία την αποθηκεύουμε σε μια θέση μνήμης.

```
while(c!=27)
{
src=cvQueryFrame(capture);
if(!src)break;
```

Στη συνέχεια παρουσιάζουμε τις εικόνες που τραβάει η κάμερα σαν ακολουθία εικόνων (βίντεο), μέχρι να πατηθεί το πλήκτρο Escape από το πληκτρολόγιο. Εάν δεν υπάρχει εικόνα το πρόγραμμα διακόπτεται.

```
cvSmooth(src,src,CV_GAUSSIAN,3,3,0,0);
cvCvtColor(src,YCrCb_image,CV_BGR2YCrCb);
cvInRangeS(YCrCb_image,YCrCb_min,YCrCb_max,YCrCb_mask);
```

Λειαίνουμε την εικόνα με το Gauss φίλτρο και την μετατρέπουμε από το χρωματικό πρότυπο RGB στο YCrCb. Στο νέο χρωματικό μοντέλο απορρίπτουμε όλες τις τιμές χρώματος εκτός από αυτές που απεικονίζουν δέρμα. Οι τιμές (3,3,0,0) στο φιλτράρισμα, αφορούν το μέγεθος του πλάτους και του ύψους του παραθύρου του φίλτρου. Τα μηδενικά στην πραγματικότητα είναι οι προκαθορισμένες τιμές των σ_x και σ_y. Οι τιμές δίνονται από το παρακάτω τύπο:

$\sigma_x=(n_x/2)*0.3+0.8$ n: μέγεθος παραθύρου του φίλτρου

$\sigma_y=(n_y/2)*0.3+0.8$

Μεγάλο μέγεθος παραθύρου διαστρεβλώνει την εικόνα και δεν γίνεται ορθή αναγνώριση.

```
cvDilate(YCrCb_mask,YCrCb_mask,0,1);
```

```
cvErode(YCrCb_mask,YCrCb_mask,0,1);
```

Ακολουθούμε τη διαδικασία κλεισίματος, που περιγράψαμε στο κεφάλαιο 2, για να έχουμε καλύτερο αποτέλεσμα στην επεξεργασία του βίντεο. Το μηδέν σημαίνει ότι χρησιμοποιείτε μια 3Χ3 μήτρα συνέλιξης για την επεξεργασία εικόνας, ενώ ο άσος δείχνει πόσες φορές εφαρμόζεται αυτή η μήτρα στην εικόνα που επεξεργαζόμαστε.

```
blobs=CBlobResult(YCrCb_mask,NULL,0);
blobs.Filter(blobs,B_INCLUDE,CBlobGetArea(),B_GREATER,5000);
blobs.GetNthBlob(CBlobGetArea(),blobs.GetNumBlobs(),blobarea);
```

Μετά τη διαδικασία του κλεισίματος πραγματοποιούμε εξαγωγή των blob. Βρίσκουμε τις άσπρες περιοχές της δυαδικής εικόνας, διώχνουμε όσες περιοχές έχουν εμβαδό μεγαλύτερο του 5000 (εμπειρική επιλογή) και κρατάμε το blob με το μεγαλύτερο εμβαδό.

```
for(int i=0;i<num_blobs;i++)
 {
 blobarea=blobs.GetBlob(i);
       blobarea.FillBlob(blobimage,CV_RGB(255,255,255));
 }
```

Γεμίζουμε κάποια κενά του Blob με άσπρο χρώμα για να έχουμε ομοιόμορφο αποτέλεσμα. Για να το κάνουμε αυτό επιστρέφουμε στο χρωματικό μοντέλο RGB.

```
cvCanny(blobgray,YCrCb_edge,129,100,5)
```

Υστερα από τη διαδικασία του fill blob γυρνάμε στη δυαδική εικόνα για να υλοποιήσουμε τη διαδικασία του αλγόριθμου Canny. Βρίσκουμε τις τιμές που καθορίζονται από το υψηλό και από το χαμηλό κατώφλι, στο στάδιο που ονομάζεται κατωφλίωση με υστέρηση. Η τελευταία τιμή αναφέρεται στο μέγεθος της οπής που χρησιμοποιείται από το παράγωγο Sobel όπως έχουμε ήδη αναφέρει. Το αποτέλεσμα της διαδικασίας είναι η εύρεση του περιγράμματος του χεριού.

```
cvFindContours(blobgray,storage,&contours,sizeof(CvContour),CV_RE
TR_LIST,CV_CHAIN_APPROX_SIMPLE);
CvSeq* contours2=NULL
double result=0,result2=0;
while(contours)
{
result=fabs(cvContourArea(contours,CV_WHOLE_SEQ));
if (result>result2) {result2=result;contours2=contours;};
contours=contours->h_next;
                    }
```

Σε αυτή τη διαδικασία το περίγραμμα αποκτά αξία καθώς αποτελείται από ακολουθία σημείων. Αφού βρίσκουμε την ακολουθία των σημείων που δημιουργούν το περίγραμμα, φροντίζουμε να πάρουμε τις μεγαλύτερες τιμές της ακολουθίας.

```
CvSeq* hull = cvConvexHull2(approx,0,CV_CLOCKWISE,0);
 CvSeq* defect=cvConvexityDefects(approx,hull,dftstorage);
for(;defect;defect=defect->h_next)
{           nodef=defect->total;
gesture.s_nodef=defect->total;
cout<<"nodefect "<<gesture.s_nodef<<endl;
defectArray=(CvConvexityDefect*)cvMemStorageAlloc(defectArrays,sizeof(CvConvexi
tyDefect)*nodef);
cvCvtSeqToArray(defect,defectArray,CV_WHOLE_SEQ);
```

Στη συνέχεια σχηματίζουμε ένα πολύγωνο που περικλείει το περίγραμμα του χεριού και βρίσκουμε τα κυρτά σημεία του πολυγώνου. Οι κορυφές των δακτύλων παριστάνονται από τα κυρτά σημεία του πολυγώνου.

```
struct ActualGesture
{
float s_angle;
float s_ratio;
float s_comp;
float s_nodef;
};
```

Δημιουργήσαμε την δομή δεδομένων ActualGesture για να λαμβάνουμε τα χαρακτηριστικά της γωνίας, της αναλογίας των αξόνων της έλλειψης, τη σχέση μεταξύ εμβαδού και περιμέτρου και τον αριθμό των κυρτών σημείων του πολυγώνου. Τα χαρακτηριστικά λαμβάνονται τη στιγμή που ο χρήστης της εφαρμογής πραγματοποιεί την χειρονομία.

```
class Class
{
public:
float Convert(const TiXmlElement* element)
{
assert(element);
assert(element->Value());
assert(element->GetText());
std::istringstream stream(element->GetText());
float f;
stream>>f;
return f;
}
int ConvertInt (const char* value)
{
std::istringstream stream(value);
int in;
stream>>in;
return in;
}
};
```

Δημιουργούμε την κλάση Class, η οποία μετατρέπει τη μορφή των αρχείων xml σε τύπους δεδομένων που αναγνωρίζει η γλώσσα C++. Οι πληροφορίες που παίρνουμε από το αρχείο xml είναι οι τιμές των χαρακτηριστικών που έχουμε υπολογίσει από τις μετρήσεις που πραγματοποιήσαμε.

```
for(features=Gest->FirstChildElement();features;
features=features->NextSiblingElement())
{
t_angle=conv.Convert(features->FirstChildElement());
t_ratio=conv.Convert(features->FirstChildElement("ratio"));
t_nodef=conv.Convert(features->FirstChildElement("nodef"));
t_comp=conv.Convert(features->FirstChildElement("comp"));
```

Μέσα στην εντολή επανάληψης for γίνεται η μετατροπή των text δεδομένων του αρχείου xml σε γνώριμους τύπους δεδομένων (float).

```
if((gesture.s_nodef==t_nodef && gesture.s_nodef<=t_nodef+1)
                    &&
(gesture.s_comp>=t_comp-20 && gesture.s_comp<=t_comp+20)
                    &&
```

```
(gesture.s_ratio>=t_ratio-0.2 && gesture.s_ratio<=t_ratio+0.2)
                        &&
(gesture.s_angle>=t_angle-20 && gesture.s_angle<=t_angle+20))
                        {
intNum=conv.ConvertInt(features->FirstChild()->Value());
                        }
```

Σε αυτό το επίπεδο γίνεται ο τελικός προσδιορισμός της χειρονομίας. Πρόκειται για μια απλή εντολή if, η οποία συγκρίνει τα χαρακτηριστικά της χειρονομίας που κάνει ο χρήστης με τα προκαθορισμένα χαρακτηριστικά του xml αρχείου, τα οποία έχουν προκύψει μετά από μετρήσεις.
Μετά την ολοκλήρωση της σύγκρισης επιλέγεται η χειρονομία, τα χαρακτηριστικά της οποίας έχουν ταιριάξει απόλυτα.

```
bool WriteComPort(CString PortSpecifier,CString data)
{
DCB dcb;
DWORD byteswritten;
HANDLE hPort=CreateFile(
PortSpecifier,
GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
0,
NULL
);
if (!GetCommState(hPort,&dcb))
return false;
dcb.BaudRate=CBR_9600;
dcb.ByteSize=8;
dcb.Parity=NOPARITY;
dcb.StopBits=ONESTOPBIT;
if (!SetCommState(hPort,&dcb))
return false;
 WriteFile(hPort,data,2,&byteswritten,NULL);
CloseHandle(hPort);
}
```

Αναπτύσσουμε την παραπάνω συνάρτηση για να ανοίξουμε την σειριακή επικοινωνία της πλατφόρμας των Windows. Επιλέγουμε την θύρα επικοινωνίας που θέλουμε, καθώς και τη μορφή των δεδομένων που θέλουμε να μεταφέρουμε σειριακά. Η μορφή των δεδομένων είναι χαρακτήρες (string). Η συνάρτηση είναι τύπου Boolean, με αυτόν τον τρόπο σε περίπτωση που η θύρα δεν είναι ανοικτή παίρνουμε μήνυμα λάθους. Επίσης ορίζουμε το baud rate = 9600 το μέγεθος byte = 8 και ένα stop byte. Αυτή τη μορφή θα έχει το πλαίσιο της πληροφορίας που θα μεταφέρουμε σειριακά από τον υπολογιστή προς τον μικροελεγκτή.

5.2 Κώδικας προκαθορισμένων χαρακτηριστικών σε αρχείο XML

Στον κώδικα που έχουμε αναπτύξει στο αρχείο xml έχουμε σκοπό την αποθήκευση χαρακτηριστικών, τα οποία λάβαμε μετά από μετρήσεις που πραγματοποιήσαμε σε διαφορετικές καταστάσεις φωτεινότητας.
Με τον τρόπο αυτό δημιουργούμε προκαθορισμένες χειρονομίες τις οποίες πρέπει ο χρήστης να επαναλάβει. Η κάθε χειρονομία χαρακτηρίζεται από έναν αριθμό, από το 0 έως το 10 . Φροντίζουμε να φορτωθούν οι πληροφορίες του αρχείου στο κυρίως πρόγραμμα και

να μετατραπούν σε δεδομένα αντιληπτά από την γλώσσα C++. Η μορφή του κώδικα φαίνεται παρακάτω.

```
<?xml version="1"?>
<Gest>
<Six>6
<angle>-26</angle>
<ratio>1.20</ratio>
<nodef>1</nodef>
<comp>45</comp>
</Six>
<Seven>7
<angle>-30</angle>
<ratio>1.83</ratio>
<nodef>0</nodef>
<comp>28</comp>
</Seven>
<Eight>8
<angle>-78</angle>
<ratio>1.0</ratio>
<nodef>2</nodef>
<comp>60</comp>
</Eight>
<Nine>9
<angle>-10</angle>
<ratio>1.5</ratio>
<nodef>1</nodef>
<comp>50</comp>
</Nine>
  ⋮
</Gest>
```

Η δομή που ακλουθείτε είναι η δημιουργία εμφωλευμένων στοιχείων. Το πρώτο στοιχείο το οποίο θεωρείτε ως Root element, είναι το Gest. Αυτό περιέχει και άλλα στοιχεία μέσα του, τα οποία θεωρούνται σαν First Child Elements, Second Child Elements και τα λοιπά. Κάθε Child Element περιλαμβάνει μια τιμή (0,1,...,10) η οποία είναι ο αριθμός που καθορίζει την χειρονομία. Μέσα στα child elements υπάρχουν τα siblings elements (αδελφά στοιχεία) που καθορίζουν τις διαφορετικές τιμές των χαρακτηριστικών κάθε χειρονομίας. Κάθε child element έχει τα δικά του sibling elements.

5.3 Κώδικας στον μικροελεγκτή

Ο κώδικας που αναπτύσσουμε για τον μικροελεγκτή πρέπει να είναι απλός και σύντομος, για τον λόγο αυτό δίνουμε ιδιαίτερη βαρύτητα στη σωστή χρήση των καταχωρητών. Οι εργασίες που θέλουμε να εκτελεί ο μικροελεγκτής είναι η κίνηση των κινητήρων και η σειριακή επικοινωνία με τον υπολογιστή. Κινούμαστε πάνω σε αυτά τα πλαίσια για να πετύχουμε τα καλύτερα δυνατά αποτελέσματα.

```
void init_uart(void)
{
UCSRB |= _BV(TXEN) | _BV(RXEN) | _BV(RXCIE);
UCSRC |= _BV(UCSZ1) | _BV(UCSZ0);
```

```
UBRRH=0;
UBRRL=51;
SREG |= _BV(SREG_I);
}
void Tx(unsigned char data)
{
while((UCSRA & 0x20)==0x00)
{;}
UDR=data;
}

unsigned char Rx(void)
{
while((UCSRA & 0x80)==0x00)
{;}
return UDR;
}
```

Είναι η συνάρτηση που αναπτύσσουμε για να ενεργοποιηθεί το σύστημα επικοινωνίας USART. Ενεργοποιούμε τα bit λήψης, εκπομπής από τον κατάλληλο καταχωρητή, καθορίζουμε το μέγεθος του byte και τον ρυθμό λήψης. Οι τιμές ταιριάζουν με τις αντίστοιχες τιμές μετάδοσης που ορίσαμε. Εάν οι τιμές ήταν διαφορετικές δεν θα πετυχαίναμε την σωστή μετάδοση και λήψη δεδομένων. Επίσης ελέγχουμε τα buffers για να ξέρουμε αν μπορεί να γίνει λήψη των δεδομένων. Οι πληροφορίες μεταφέρονται από στον UDR καταχωρητή.

```
void initTimer()
{
TCCR0= _BV(WGM00) | _BV(CS02) | _BV(COM01);
TCCR2=_BV(CS22) | _BV(WGM20) | _BV(CS00)| _BV(COM21);
TCCR1B=_BV(CS11) | _BV(WGM13);
TCCR1A=0xA0;
ICR1H=0x27;
ICR1L=0x10;
DDRD=_BV(PD5)| _BV(PD4) | _BV(PD7);
DDRB=_BV(PB3);
}
```

Καθορίζουμε τους ακροδέκτες εξόδου, στους οποίους συνδέονται οι κινητήρες. Επίσης υθμίζουμε τους καταχωρητές ώστε να παράγουμε τα σήματα pwm που επιθυμούμε. Τα ήματα θα κινούν τους κινητήρες προς την επιθυμητή κατεύθυνση με την κατάλληλη αχύτητα. Είναι σημαντικό να γνωρίζουμε τις ιδιότητες των σέρβο κινητήρων rc για να άνουμε τις ιδανικές επιλογές στη ρύθμιση των καταχωρητών. Οι ιδιότητες για τους ερβοκινητήρες rc έχουν ήδη αναφερθεί στο πρώτο κεφάλαιο.

```
ISR(USART_RXC_vect)
{
  uint8_t input=UDR;
  if (input=='0')
{
OCR1A=300;
OCR1B=300;
}
```

```
if (input=='1')
{
OCR1B=300;
OCR1A=1200;
}
if (input=='4')
{
OCR0=30;
OCR2=70;
}
```

Στο παραπάνω κώδικα, αναγράφεται ένα μικρό τμήμα για τον τρόπο με τον οποίο οι χειρονομίες μετατρέπονται σε κινήσεις συγκεκριμένων κινητήρων. Όπως παρατηρούμε πρόκειται για μια διακοπή λήψης. Ελέγχουμε τον καταχωρητή UDR και ανάλογα με τον αριθμό οι κινητήρες κατευθύνονται σε συγκεκριμένη θέση.
Με αυτό τον τρόπο δημιουργούνται οι κινήσεις που πραγματοποιεί η μαριονέτα.

Αντιμετώπιση Προβλημάτων: Κατά τη διάρκεια ανάπτυξης της πτυχιακής εργασίας, ήρθαμε αντιμέτωποι με προβλήματα διαφόρων τύπων. Το πρώτο πρόβλημα ήταν η επιλογή του κατάλληλου λογισμικού. Στόχος της εργασίας είναι η χρήση όσο το δυνατόν λιγότερων ηλεκτρονικών στοιχείων, η μειωμένη χρήση των υπολογιστικών πόρων, μειωμένη κατανάλωση ισχύος και διατήρηση του χαμηλού κόστους κατασκευής. Η βιβλιοθήκη της OpenCV καλύπτει τις παραπάνω προϋποθέσεις με τον καλύτερο τρόπο. Επιπλέον προβλήματα ήταν η μη συμβατή έκδοση της επιπρόσθετης βιβλιοθήκης CVBlobsLib με το IDE που εργαζόμασταν. Για να το αντιμετωπίσουμε αυτό χρειάστηκε να προσθέσουμε επικεφαλίδες στους πηγαίους κώδικες που διέθετε.
Επιπλέον δυσκολία ήταν η απότομη μεταβολή της φωτεινότητας. Πρόβλημα που δεν λύνεται απολύτως, αλλά γίνονται αρκετές προσπάθειες αντιμετώπισης του, στο κώδικα της εφαρμογής και στις αρχικές ρυθμίσεις της κάμερας. Η ανίχνευση αντικειμένων που μπορεί να θεωρηθούν ως ανθρώπινο χέρι είναι μια δυσκολία που αντιμετωπίστηκε με την χρήση των τεχνικών του blob detection, με τη αναγνώριση ως ανθρώπινου χεριού του αντικειμένου με το μεγαλύτερο εμβαδόν.
Επιπρόσθετο πρόβλημα ήταν η μορφή με την οποία θα γινόταν η μεταφορά δεδομένων από τον υπολογιστή στον μικροελεγκτή. Επιλέξαμε να τα δεδομένα να μεταφέρονται ως χαρακτήρες ώστε ο κώδικας ελέγχου να είναι απλός και σύντομος. Επίσης η εύρεση της περιόδου του σέρβο κινητήρα ήταν ένα αποτέλεσμα που προήλθε μετά από αρκετές δοκιμές και πειράματα.
Σκοπός ήταν η καλύτερη δυνατή αντιμετώπιση των παραπάνω προβλημάτων ώστε να έχουμε το επιθυμητό αποτέλεσμα.

6.ΕΠΙΛΟΓΟΣ

Στη πτυχιακή εργασία αναπτύχθηκε η μέθοδος εξαγωγής χαρακτηριστικών για την αναγνώριση των ανθρώπινων χειρονομιών που οδηγούν στη κίνηση μιας μαριονέτας με σχοινιά. Η μέθοδος αναγνώρισης βασίζεται στη λήψη σταθερών χαρακτηριστικών, που δεν μεταβάλλεται η τιμή τους όταν κουνιέται το χέρι. Η ιδιότητα αυτή είναι καθοριστική, καθώς είναι δύσκολο για τον χρήστη της εφαρμογής να έχει σταθερό το χέρι του ενώ πραγματοποιεί μια χειρονομία.

Ακλουθήσαμε τις τεχνικές της υπολογιστικής όρασης και της επεξεργασίας εικόνας για να οδηγηθούμε στο τελικό αποτέλεσμα. Αλγόριθμοι και μέθοδοι που έχουν να κάνουν τόσο με το φιλτράρισμα της εικόνας όσο και με την βελτίωση της απόδοσης του συνολικού προγράμματος. Προσπαθήσαμε να αξιοποιήσουμε τις δυνατότητες της υπολογιστικής όρασης για να μετατρέψουμε την εικόνα σε πηγή πληροφοριών. Με τον τρόπο αυτό ένα βίντεο δεν είναι απλώς μια ακολουθία εικόνων, αλλά δεδομένα που μπορούμε να εκμεταλλευτούμε για την υλοποίηση εργασιών.

Επίσης, χρησιμοποιούμε το πρότυπο μετάδοσης του Zigbee έτσι ώστε να στείλουμε στο μικροελεγκτή τα δεδομένα, που επιθυμούμαι, ασύρματα. Αυτό δίνει ευελιξία στην εφαρμογή μας, ενώ δεν είναι απαραίτητο η κατασκευή να επιβαρύνεται με την χρήση περιττών καλωδίων.

Έγινε προσπάθεια να διατηρηθεί απλή η εφαρμογή. Για τον λόγο αυτό δεν χρησιμοποιήθηκαν αισθητήρια ή πολλά ηλεκτρονικά στοιχεία. Επίσης για τον διαχωρισμό του χεριού δεν χρησιμοποιείται γάντι ή patches έτσι ώστε να διευκολύνει την εργασία, αλλά προτιμήσαμε να κάνουμε την εφαρμογή όσο το δυνατόν πιο εύχρηστη. Επιπλέον προσπαθήσαμε να κρατήσουμε το κόστος της κατασκευής σε χαμηλά επίπεδα με την αποφυγή χρήσης περιττών στοιχείων.

Η απλότητα της εφαρμογής ήταν βασικό ζητούμενο και για τον λόγο αυτό ο αριθμός των χειρονομιών είναι περιορισμένος. Το αποτέλεσμα είναι ο χρήστης να μπορεί να θυμάται τις κινήσεις που αντιστοιχούν σε κάθε χειρονομία. Επιπλέον, για να μπορεί ο χρήστης να κινεί το χέρι του μέσα στα όρια της εικόνας που λαμβάνει η κάμερα, χρησιμοποιούμε ένα πίνακα. Προτιμάμε ο πίνακας να είναι μαύρος, ώστε να αποφύγουμε τυχόν αντανακλάσεις του φωτός. Οι έντονες αλλαγές της φωτεινότητας είναι σημαντικό πρόβλημα, το οποίο επηρεάζει την αναγνώριση, αλλά δεν έχει βρεθεί ολοκληρωτική λύση έως σήμερα.

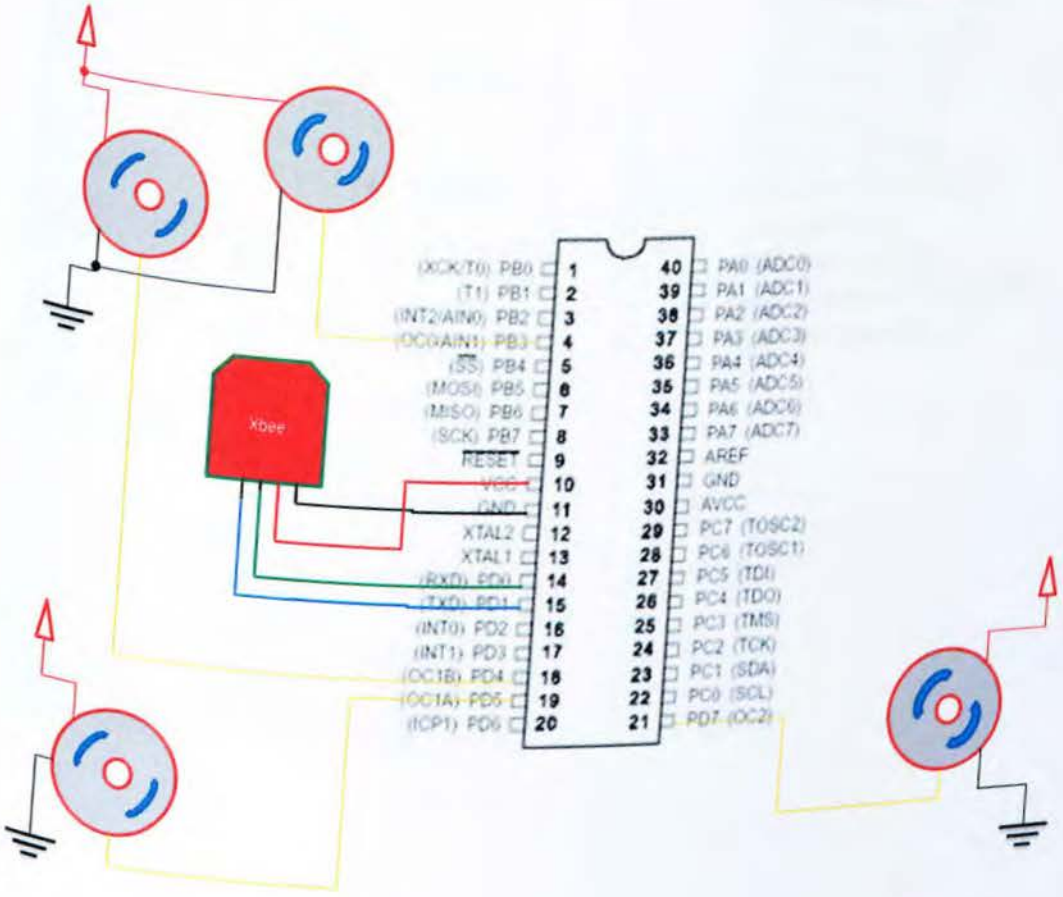
Εάν φτάσουμε σε υψηλά επίπεδα αναγνώρισης του χεριού, τότε μπορούν να δημιουργηθούν σημαντικές εφαρμογές σε διάφορους τομείς. Εφαρμογές που θα διευκολύνουν την καθημερινότητα των ανθρώπων και εκτείνονται σε τομείς όπως η ψυχαγωγία, η ρομποτική και σε ιδανικό σημείο στην χειρουργική. Οι δυνατότητες είναι αρκετές αλλά υπάρχουν αρκετά προβλήματα που πρέπει να λυθούν.

7. ΠΑΡΑΡΤΗΜΑ

Στο σχήμα που παρουσιάζεται παρακάτω βλέπουμε την σύνδεση των στοιχείων που χρησιμοποιήσαμε με τον μικροελεγκτή. Έχουμε τέσσερις κινητήρες οι οποίοι συνδέονται με τους κατάλληλους ακροδέκτες χρησιμοποιώντας το καλώδιο δεδομένων που διαθέτουν. Τα υπόλοιπα καλώδια, μαύρο, κόκκινο, συνδέονται στη γείωση και στη τροφοδοσία αντίστοιχα. Στο σχήμα η τροφοδοσία είναι το πράσινο βέλος και η γείωση οι τρείς μαύρες γραμμές.

Το Zigbee συνδέει τους ακροδέκτες λήψης και εκπομπής δεδομένων με τα αντίστοιχα data in κι data out pin που διαθέτει. Να σημειώσουμε ότι το Xbee module που χρησιμοποιούμε για να δημιουργήσουμε το δίκτυο ασύρματης επικοινωνίας, χρειάζεται 3.3V στην είσοδο του. Για τον λόγο αυτό χρειάζεται και Xbee explorer Regulated πλακέτα. Οι ακροδέκτες τάσης και γείωσης του μικροελεγκτή μπορούν να συνδεθούν κατευθείαν στη πλακέτα και να έχουμε μετάδοση πληροφορίας χωρίς να μειώνεται η ποιότητα της πληροφορίας που μεταφέρουμε.

Όπως παρατηρούμε δεν υπάρχει πολυπλοκότητα στην σύνδεση του Atmega16 με εξωτερικές συσκευές.



7.1 *Σύνδεση μικροελεγκτή Avr Atmega16 με τους σερβοκινητήρες και το Xbee Module*

Στις παρακάτω εικόνες παρουσιάζονται τα χαρακτηριστικά των σέρβο κινητήρων rc και της κεραίας Xbee για την δημιουργία του δικτύου επικοινωνίας μεταξύ pc και Avr Atmega16.

TowerPro MG90 - Micro Servo

XBee 1mW Chip Antenna - Series 1

Basic Information:

Modulation:	Analog
Torque:	**4.8V:** 30.6 oz-in (2.20 kg-cm) **6.0V:** 34.7 oz-in (2.50 kg-cm)
Speed:	**4.8V:** 0.11 sec/60° **6.0V:** 0.10 sec/60°
Dimensions:	**Length:** 0.91 in (23.1 mm) **Width:** 0.48 in (12.2 mm) **Height:** 1.14 in (29.0 mm)
Gear Type:	Metal
Rotation/Support:	Dual Bearings

Features:

- 3.3V @ 50mA
- 250kbps Max data rate
- 1mW output (+0dBm)
- 300ft (100m) range
- Built-in antenna
- Fully FCC certified
- 6 10-bit ADC input pins
- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set

Ορολογία:

2D : Επίπεδο δύο διαστάσεων
3D : Επίπεδο τριών διαστάσεων
ADC : Μετατροπέας αναλογικού σήματος σε ψηφιακή μορφή
Assembly : Γλώσσα μηχανής, χρησιμεύει στον προγραμματισμό μικροελεγκτών
Avr ATmega16: Μικροελεγκτής
Background : Φόντο
Baud rate : Ταχύτητα με την οποία μεταδίδονται ψηφιακά δεδομένα
Bit : Δυαδικό ψηφίο
Blob extraction: Λήψη τμήματος εικόνας, συνήθως περιέχει σημαντικές πληροφορίες
Bluetooth : Πρότυπο μετάδοσης δεδομένων
Byte : Οκτώ δυαδικά ψηφία
C++ : Γλώσσα προγραμματισμού
Canny : Τεχνική ανίχνευσης περιγράμματος, το συναντάμε στο computer vision
Class : Κλάση στο προγραμματισμό
Closing : Διαδικασία κλεισίματος, μορφολογική επεξεργασία εικόνας
Color space : Χρωματικός χώρος
Computer Vision: Υπολογιστική όραση
Contours : Περίγραμμα, τεχνική της υπολογιστικής όρασης
Convex : Κυρτό σχήμα
Convolution : Συνέλιξη
Counter : Μετρητής, καταχωρητές μικροελεγκτή
CPU : Κεντρική μονάδα επεξεργασίας
Data structure: Δομή δεδομένων, το συναντάμε στο προγραμματισμό, στη γλώσσα C++
Dilation : Διαστολή, μέθοδος φιλτραρίσματος εικόνας
Duty cycle : Ποσοστό παλμού σε κατάσταση υψηλή, σε τετραγωνικό σήμα
EEPROM : Μνήμη στην οποία μπορούμε να γράφουμε
Erosion : Συστολή, μέθοδος φιλτραρίσματος εικόνας
Feature extraction: Λήψη χαρακτηριστικών, τεχνική της υπολογιστικής όρασης
FFD : Συσκευή πλήρους λειτουργίας
Flags : bits που ενημερώνουν καταχωρητές
Flood Fill : Μέθοδος επεξεργασίας εικόνας
Foreground : Προσκήνιο
Full duplex : αποστολή πληροφοριών προς δύο κατευθύνσεις
GNU GCC : Μεταγλωττιστής που περιέχεται στο λογισμικό WinAvr
Gradient vector: διάνυσμα κλίσης, δείχνει την υψηλότερη μεταβολή της έντασης ενός pixel
Histogram : Ιστόγραμμα, γραφική αναπαράσταση μιας εικόνας
Hertz : μονάδα μέτρησης της συχνότητας
IDE : Περιβάλλον ολοκληρωμένης ανάπτυξης, περιέχει editor, compiler κ.α.
ISR : Καταχωρητής εξυπηρέτησης διακοπών
OpenCV : Βιβλιοθήκη συναρτήσεων υπολογιστικής όρασης
Opening : Διαδικασία κλεισίματος, μορφολογική επεξεργασία εικόνας
Pixel : εικονοστοιχείο, φανερώνει την ανάλυση της εικόνας
Pixel depth : βάθος εικονοστοιχείου, φανερώνει τον αριθμό των χρωμάτων μιας εικόνας
PWM : Μετατόπιση πλάτους τετραγωνικού παλμού
RFD : Συσκευή μειωμένης λειτουργίας
RGB : Χρωματικό πρότυπο
RISC : Αρχιτεκτονική κατασκευής μικροεπεξεργαστή
Segmentation : Τμηματοποίηση, διαχωρισμός επιθυμητού αντικειμένου από το περιβάλλον
Servo motors : Σέρβο κινητήρες

Skin segmentation: Διαχωρισμός τμημάτων χρώματος δέρματος από το περιβάλλον
Smoothing : Λείανση, τεχνική της επεξεργασίας εικόνας
SPI : Σειριακή επικοινωνία. Ο πομπός και ο δέκτης έχουν την ίδια πηγή ρολογιού
SRAM : Στατική μνήμη τυχαίας προσπέλασης
Timer : Χρονιστής
Threshold : κατωφλίωση, τεχνική για την επεξεργασία εικόνας
TWI : Σύστημα επικοινωνίας μεταξύ συσκευών και μικροελεγκτή
USART : Σύγχρονο/ ασύγχρονο σειριακό σύστημα επικοινωνίας
VDC : Τάση συνεχούς ρεύματος
Visual Studio :Περιβάλλον ολοκληρωμένης ανάπτυξης, γράφουμε κώδικες κ΄προγράμματα
WinAvr : Λογισμικό
WPAN : Ασύρματο προσωπικό δίκτυο
YCrCb : Κωδικοποιημένο χρωματικό πρότυπο
ZigBee : Πρότυπο μετάδοσης δεδομένων

Βιβλιογραφία

[1] Σύρκος 2005, Γεώργιος Π. Σύρκος, « Ψηφιακή Επεξεργασία Σήματος», Αθήνα 2007

[2] Κυτάγιας χ.χ, Δ. Κυτάγιας, «C++ Προγραμματισμός», Σύγχρονη Εκδοτική

[3] Σκόδρας κ΄Αναστόπουλος 2002, Σκόδρας Α. και Αναστασόπουλος Β. «Ψηφιακή Επεξεργασία Εικόνων και Σημάτων» , Ανοικτό Πανεπιστήμιο, 2002

[4] E. Stergiopoulou, N. Papamarkos και A. Atsalakis, «Hand Gesture Recognition Via a New Self-Organized Neural Network »

[5] F.Perales and M. Absolo , «Vision por Ordenador»

[6] Rayo 2009, Simon Garces Rayo, «Hand Gestures And Hand Movement Recognition For Multimedia Player Control», 2009

[7] Bradski and Kaehler 2008,Gary Bradski and Adrian Kaehler, «Learning OpenCV», 2008

[8] Malik 2003, Shahzad Malik, «Real-time Hand Tracking and Finger Tracking for Interaction», 2003

[9] Chang and Tseng 2011, Rong-Chi Chang, Fei-Chun Tseng, «Automatic Detection and Correction for Glossy
Reflections in Digital Photograph», Journal Of Multimedia, Vol. 6, No. 2, April 2011

[10] Deitel and Deitel χ.χ, H.M. Deitel και P.J. Deitel, «C Προγραμματισμός,Εισαγωγή στη C++ και την Java», Εκδότης:Μ.Γκιουρδας

[11] Ondřej Novák, František Ondřej, Jan Ondřej, «Gesture recognition»

[12] Intel, «Open Source Computer Vision Library, Reference Manual», 2001

[13] Rehg and Kanade 1993, James M. Rehg Takeo Kanade, «DigitEyes: Vision-Based Human Hand Tracking », December 1993

[14] Barry B. Brey, «Using the Serial Ports in Visual C++», 2005

[15] Forsyth και Ponce 2002, David A. Forsyth and Jean Ponce «Computer Vision A Modern Approach»,Prentice Hall, 2002

[16] Auslander and Kempf 1996, David M. Auslander and Carl J. Kempf, «Μηχανοτρονική», Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., 1998

[17] Szeliski 2010, «Computer Vision: Algorithms and Applications», by Richard Szeliski, Springer, 2010

[18] Λιανός 2011, Λιανός Αναστάσιος «Ευφυής Φορητή Πλατφόρμα Ηλεκτρονικού Εμπορίου με Δυνατότητα Επεξεργασίας Εικόνας και Ενσωμάτωσης Γεωγραφικών Δεδομένων», Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, 2011

[19] Παπαμάρκος 2005, Παπαμάρκος Η. Νικόλαος «Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας», Γκιουρδας Β., 2005

[20] Pass and Zabih 2000, G.Pass and R.Zabih «Comparing Images Using Joint Histograms», Multimedia Systems, Vol. 75, No 1, pp 73-85, 2000

[21] Barrett and Pack 2007, Steven F. Barrett and Daniel J. Pack «Atmel Avr Microcontroller Primer, Programming and Interfacing», 2007

[22] San José State University Dept. of Mechanical and Aerospace Engineering 2010, « Introduction to the ATmega16 », 2010

[23] Gislason 2008, Drew Gislason «Zigbee Wireless Networking», Newnes 2008

[24] Kudoand Sklansky,M. Kudo and J. Sklansky, «Comparison of algorithms that select features for patternclassifiers», *Pattern Recognition*, 2000,pp. 25-41

[25] Jones and Rehg 1999, M. Jones and J. Rehg, «Statistical Color Models With Application To Skin Detection», In Proccessings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1999 Vol 1, pp. 274-280

[26] Παναγιώτης Αργυρέας, « Προηγμένες εφαρμογές των μαθηματικών στην ψηφιακή επεξεργασία σήματος με χρήση της Matlab», ΑΤΕΙ Κρήτης Παράρτημα Χανίων τμ. Ηλεκτρονικής,2010

[27] Homma and Takenaka ,1985,K. Homma and E.-I. Takenaka, "An image processing method for feature extraction of space-occupying lesions," *Journal of Nuclear Medicine* 26, 1985

[28] http://el.wikipedia.org/wiki/RGB, Το χρωματικό μοντέλο RGB

[29] http://en.wikipedia.org/wiki/YCbCr, Το χρωματικό μοντέλο YCrCb

[30] http://opencv.willowgarage.com/wiki/, Η βιβλιοθήκη OpenCV

[31] http://www.grinninglizard.com/tinyxml/, Αρχεία xml

[32] http://www.aishack.in, Περιγραφή OpenCV