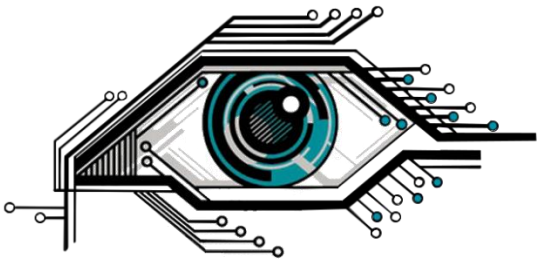
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΝΣΩΜΑΤΩΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΒΑΣΙΣΜΕΝΟΥ ΣΕ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗ ΛΟΓΙΚΗ, ΓΙΑ ΤΟΝ ΕΝΤΟΠΙΣΜΟ ΤΗΣ ΚΟΡΗΣ ΜΑΤΙΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΑΓΙΩΤΗΣ ΚΟΥΤΡΟΥΜΑΝΟΣ



ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

ΚΑΘ. ΔΟΛΛΑΣ ΑΠΟΣΤΟΛΟΣ (ΕΠΙΒΛΕΠΩΝ)

ΚΑΘ. ΠΝΕΥΜΑΤΙΚΑΤΟΣ ΔΙΟΝΥΣΙΟΣ

ΚΑΘ. ΖΕΡΒΑΚΗΣ ΜΙΧΑΛΗΣ

Χανία,Νοέμβριος 2018

ABSTRACT

Η ανίχνευσή του ματιού είναι ένα πεδίο έρευνας στον τομέα της μηχανικής όρασης που αναπτύσσεται με μεγάλο ρυθμό τα τελευταία χρόνια, καθώς μπορεί να εξυπηρετήσει σε πάρα πολλές εφαρμογές. Η παρούσα διπλωματική εργασία επικεντρώνεται στον εντοπισμό θέσης της κόρης, αφού αποτελεί την βάση για την υλοποίηση οποιασδήποτε εφαρμογής σχετίζεται με την ανίχνευσή του ματιού ή της ίριδας. Στόχος της παρούσης διπλωματικής εργασίας είναι η σχεδίαση και η υλοποίηση ενός ενσωματωμένου συστήματος, βασισμένου σε αναδιατασσόμενη λογική που θα εντοπίζει το κέντρο και την ακτίνα της κόρης. Για το σκοπό αυτό μελετήθηκαν ήδη γνωστοί αλγόριθμοι που σχετίζονται με τον εντοπισμό κόρης, μοντελοποιήθηκαν οι αλγόριθμοι Set και Pupil Labs στην Matlab και τα συμπεράσματα που εξήχθησαν χρησιμοποιήθηκαν για να προτείνουμε μία δικιά μας μοντελοποίηση. Στην συνέχεια με την βοήθεια του Xilinx System Generator υλοποιήσαμε την σχεδίαση Hardware της προτεινόμενης μοντελοποίησης, η οποία εκτελέστηκε στην FPGA ZC706.

Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ

1.1 Συνεισφορά της Διπλωματικής

Η ανίχνευσή του ματιού είναι ένα πεδίο έρευνας στον τομέα της μηχανικής όρασης που αναπτύσσεται με μεγάλο ρυθμό τα τελευταία χρόνια, καθώς μπορεί να εξυπηρετήσει σε πάρα πολλές εφαρμογές. Ο βαθμός που γίνεται η ανίχνευση, καθώς και το ποιο ακριβώς σημείο του ματιού τίθεται προς ανίχνευση, διαμορφώνεται ανάλογα με την εφαρμογή που υλοποιείται. Για παράδειγμα, μία εφαρμογή που έχει σαν στόχο την ανάλυση και την ταυτοποίηση της ίριδας θα αναζητήσει το μέρος της ίριδας που εμφανίζεται περιφερειακά της κόρης του ματιού, το οποίο θα υποστεί περεταίρω επεξεργασία στη συνέχεια. Αντίθετα, μία εφαρμογή που έχει σαν στόχο την δημιουργία ενός συστήματος παρακολούθησης του βλέμματος του ματιού, θα αναζητήσει την κόρη του ματιού και θα χαρτογραφήσει το κέντρο της. Κοινή συνισταμένη σε όλα τα παραπάνω αποτελεί ο εντοπισμός της θέσης της ίριδας. Ο εντοπισμός θέσης αποτελεί πρωταρχικό και βασικό στάδιο όλων των τεχνικών ανίχνευσης της ίριδας, εφόσον πάνω της θα βασιστεί η μετέπειτα επεξεργασία, που διαφοροποιείται ανάλογα με τον στόχο που έχει τεθεί. Επιπλέον αποτελεί το στάδιο που καταναλώνει περισσότερο χρόνο σε ένα σύστημα ανίχνευσης.

Η δημιουργία ενσωματωμένων συστημάτων για την ανίχνευση της ίριδας, που θα είναι μικρά σε μέγεθος και με χαμηλό κόστος, αποτελεί πεδίο έρευνας που εμφανίζεται πάρα πολύ έντονα. Εξαιτίας της μεγάλης κατανάλωσης χρόνου για τον εντοπισμό της θέσης, όπως αναφέρθηκε και προηγουμένως, εμφανίζεται η ανάγκη δημιουργίας εξειδικευμένου Hardware, που θα δίνει στο ενσωματωμένο σύστημα την δυνατότητα να πετύχει επιδόσεις που θα αντιστοιχούν σε πραγματικό χρόνο. Για αυτό τον λόγο οι FPGA (Field Programmable Gate Array) αποτελούν τεχνολογία που

μπορεί να δώσει λύσεις στα παραπάνω προβλήματα, τόσο γιατί χρησιμοποιούνται σε μεγάλο βαθμό για να υλοποιήσουν εργασίες που αφορούν την επεξεργασία εικόνας σε πραγματικό χρόνο, όσο γιατί ενδείκνυται για εκτελέσεις εξειδικευμένου hardware εξαιτίας της ρυθμιζόμενης λογικής τους και της χαμηλής κατανάλωσης ενέργειας σε σχέση με επεξεργαστές γενικού σκοπού.
Έχοντας γνώση ότι ο εντοπισμός θέσης της κόρης παίζει κυρίαρχο ρόλο και αποτελεί την βάση για την υλοποίηση οποιασδήποτε εφαρμογής που αφορά την επεξεργασία του ματιού, στην παρούσα διπλωματική φιλοδοξούμε να συνεισφέρουμε, προτείνοντας μία σχεδίαση Hardware υλοποιημένη στην FPGA ZC706. Αφού μελετήσαμε το ζήτημα της ανίχνευσης κόρης και αξιολογήσαμε διάφορους αλγόριθμους και τεχνικές που είναι ήδη γνωστοί και ασχολούνται με αυτό, όπως και αρκετές σχεδιάσεις σε FPGA που είναι σχετικές, προτείνουμε την δικιά μας σχεδίαση Hardware. Η υλοποίησή μας δεν περιορίζεται στην εξυπηρέτηση μιας συγκεκριμένης εφαρμογής, αλλά αντίθετα ανιχνεύει την θέσει της κόρης και επιστρέφει την τοπολογία του κέντρου της και την ακτίνα της, πληροφορίες που είναι αξιοποιήσιμες σε οποιαδήποτε εφαρμογή ανίχνευσης ματιού.

Επιπρόσθετα στην παρούσα διπλωματική το dataset που εξετάζεται αποτελείται από εικόνες ματιών NIR (Near Infra-Red) οι οποίες προτιμώνται για εφαρμογές ανίχνευσης ίριδας, εξαιτίας της καλής ευκρίνειας που διαθέτουν. Με αυτό ως δεδομένο αποκτά ιδιαίτερη σημασία ο εντοπισμός της κόρης, εφόσον στις NIR εικόνες ματιών, η κόρη είναι πολύ πιο ευδιάκριτη και ο εντοπισμός της προηγείται από αυτόν της ίριδας, αφού απλοποιεί τον τελευταίο ραγδαία.

1.2 Οργάνωση της Διπλωματικής

- Στο Κεφάλαιο 2 αναλύεται η σχετική έρευνα που χρειάστηκε για την υλοποίηση της διπλωματικής εργασίας. Επιπλέον γίνεται αναφορά στην βιβλιογραφία που μελετήθηκε και επισημαίνονται τα συμπεράσματα που μας προσέφερε.

- Στο Κεφάλαιο 3 γίνεται παρουσίαση των είδη γνωστών αλγόριθμων εντοπισμού θέσης της κόρης του ματιού, αναλύεται η προσέγγισή μας στους αλγόριθμους Set και Pupil Labs και τέλος γίνεται εκτενής παρουσίαση του δικού μας αλγόριθμου, που υλοποιήσαμε σε Matlab, αξιοποιώντας τα συμπεράσματα που βγάλαμε. Επιπλέον όπου χρειάζεται αναλύεται το θεωρητικό υπόβαθρο για την κατανόηση των αλγορίθμων.

- Στο Κεφάλαιο 4 γίνεται παρουσίαση και αναλυτική περιγραφή της σχεδίασης που υλοποιήσαμε σε Fpga με την χρήση του Xilinx System Generator, βασιζόμενοι στον αλγόριθμο που μοντελοποιήσαμε στην Matlab.

- Στο Κεφάλαιο 5 γίνεται παρουσίαση του τρόπου εξαγωγής των αποτελεσμάτων από την Fpga καθώς και παρουσιάζονται και αξιολογούνται τα αποτελέσματα της διπλωματικής εργασίας.

- Στο Κεφάλαιο 6 γίνεται μια αποτίμηση της εργασίας, των αποτελεσμάτων που εξήχθησαν και τέλος παρουσιάζονται μελλοντικές βελτιώσεις και επεκτάσεις.

2 ΣΧΕΤΙΚΗ ΕΡΕΥΝΑ

2.1 Εφαρμογές

Άτομα που έχουν χάσει τον έλεγχο μεγάλου μέρους των μυών τους και δεν είναι πλέον σε θέση να εκτελούν κινήσεις, ως αποτέλεσμα ασθενειών ή ατυχημάτων, μπορούν να ωφεληθούν ευρέως από τα συστήματα παρακολούθησης ματιών για να αλληλοεπιδρούν και να επικοινωνούν στην καθημερινότητα τους. Τα συστήματα παρακολούθησης ματιών παρέχουν πολλές επιλογές σε αυτά τα άτομα, όπως μια διεπαφή δακτυλογράφησης χρησιμοποιώντας το μάτι, που θα μπορούσε να δημιουργεί έξοδο κειμένου από ομιλία. Μέσω του ελέγχου των ματιών εμφανίζονται αρκετές δυνατότητες, συμπεριλαμβανομένης της οδήγησης ηλεκτρικών αναπηρικών αμαξιδίων ή της ενεργοποίησης της τηλεόρασης και άλλων συσκευών.

Οι Kocejko et al. [1] εισήγαγαν το «Eye Mouse», το οποίο μπορεί να χρησιμοποιηθεί από άτομα με σοβαρές αναπηρίες. Ο δρομέας του ποντικιού καθορίζεται με βάση τις πληροφορίες που αποκτώνται σχετικά με τη θέση των ματιών ως προς την οθόνη. Έτσι δίνεται η δυνατότητα χρήσης υπολογιστή από άτομα με σοβαρές κινητικές αναπηρίες.

Οι Fu και Yang [2] πρότειναν να χρησιμοποιηθούν πληροφορίες που λαμβάνονται από την παρακολούθηση του ματιού, μέσω βίντεο, για να ελέγχεται μια οθόνη. Η Οθόνη ελέγχεται σύμφωνα με την κίνηση του ματιού.

Οι Lupu et al. [3] πρότειναν το " Asistsys " που είναι μια επικοινωνία για ασθενείς που πάσχουν από νευρο-κινητικές αναπηρίες. Αυτό το σύστημα βοηθάει τους ασθενείς να εκφράζουν τις ανάγκες ή τις επιθυμίες τους.

Η αναγνώριση ίριδας χρησιμοποιείται ευρέως για τη βιομετρική πιστοποίηση. Ο εντοπισμός της ίριδας είναι ένα σημαντικό και κρίσιμο βήμα για τέτοιου είδους εφαρμογές. Μελέτες σχετικά με αυτό παρουσιάζονται στα [4], [5].

Τα συστήματα ηλεκτρονικής μάθησης είναι ηλεκτρονικά συστήματα διδασκαλίας και σήμερα είναι πολύ συνηθισμένα. Ωστόσο, παρά το γεγονός ότι οι χρήστες συνήθως έχουν συνηθίσει να αλληλοεπιδρούν με μηχανές, η εμπειρία μάθησης μπορεί να είναι αρκετά διαφορετική. Συγκεκριμένα, το «συναισθηματικό» μέρος είναι σημαντικό στην αλληλεπίδραση μεταξύ δασκάλου και εκπαιδευόμενου και λείπει από τις διαδικασίες μάθησης που βασίζονται σε υπολογιστή. Οι Calvi et al. [6] παρουσίασαν μία μέθοδο E-learning, κατά την οποία τα δεδομένα των ματιών χρησιμοποιούνται για την παρακολούθηση των δραστηριοτήτων των χρηστών, καθώς και διάφορων συμπεριφορών και συναισθηματικών καταστάσεων.

Αρκετή έρευνα διεξάγεται σχετικά με την εφαρμογή μεθόδων παρακολούθησης ματιών στην αυτοκινητοβιομηχανία, με στόχο την ανάπτυξη συστημάτων παρακολούθησης και υπό βοήθειας που χρησιμοποιούνται σε αυτοκίνητα. Για παράδειγμα, ο εντοπισμός του βλέμματος θα μπορούσε να χρησιμοποιηθεί σε αυτοκίνητα για να προειδοποιήσει τους οδηγούς όταν αρχίζουν να κουράζονται ή να κοιμούνται κατά την οδήγηση. Τέτοιες μελέτες είναι οι [7], [8].

2.2 Μέθοδοι και Τεχνικές

2.2.1 Αλγόριθμοι Για την Εύρεση της Θέσης της Κόρης

Όσον αφορά αλγόριθμους που εξειδικεύονται στον εντοπισμό θέσης της κόρης του ματιού, μελετήθηκαν συγκεκριμένα οι αλγόριθμοι Else [9], Excuse [10], Swirski [11], Starbust [12], Set [13] και Pupil Labs [14] όπως και η σύγκριση τους, που γίνεται στο [15]. Οι συγκεκριμένοι αλγόριθμοι αναλύονται εκτενώς στο 3^{ο} Κεφάλαιο της παρούσας διπλωματικής εργασίας.

2.2.2 Γενικές Τεχνικές που Μελετήθηκαν

2.2.2.1 Ανίχνευση ματιού με Pattern Recognition

Οι Raudonis et al. [16] κατά την μέθοδό τους, χρησιμοποίησαν Principal Component Analysis (PCA) για την εύρεση έξι βασικών στοιχείων του ματιού προκειμένου να μειώσουν προβλήματα, που αφορούν τις μεγάλες διαστάσεις μιας εικόνας. Έπειτα τεχνητό Νευρωνικό δίκτυο χρησιμοποιείται για να εντοπιστεί και να ταξινομηθεί η θέση της κόρης του ματιού.

Οι Tang και Zhang [17] πρότειναν μία μέθοδο που χρησιμοποιεί αλγόριθμο ανίχνευσής σε συνδυασμό με την πρόβλεψη της ανίχνευσης. Ουσιαστικά χρησιμοποιείται ένα μοντέλο το οποίο σε βίντεο προβλέπει που είναι πιθανόν να βρίσκεται η θέση του ματιού λαμβάνοντας υπόψιν τα προηγούμενα frame.

Οι Kuo et al. [18] παρουσίασαν μία μέθοδο κατά την οποία αξιολογούνται μία σειρά από κρυφές παραμέτρους, οι οποίες βασίζονται στην παρατήρηση των δεδομένων. Η διαδικασία της ανίχνευσης της κόρης του ματιού ξεκινάει αφού εκτιμηθούν πιθανές θέσεις ως προς την πιθανότητα να είναι η τοποθεσία της κόρης του ματιού.

Οι Lui et al. [19] πρότειναν μία γρήγορη και σταθερή μέθοδο ανίχνευσής η οποία βασίζεται στον αλγόριθμο Viola-Jones για να ανιχνευθεί αρχικά το πρόσωπο χρησιμοποιώντας χαρακτηριστικά ψηφιακής εικόνας. Ύστερα εφαρμόζεται Template Matching, προκειμένου να εντοπιστεί εν κατακλείδι η τοποθεσία των ματιών.

2.2.2.2 Εντοπισμός ματιού με βάση τις φασματικές αντανακλάσεις

Οι Yang et al. παρουσίασαν δύο μεθόδους εντοπισμού του ματιού, βασισμένες στις φασματικές αντανακλάσεις. Σε αυτές τις μεθόδους το διάνυσμα από το κέντρου του ματιού μέχρι την αντανάκλαση χρησιμοποιείται για να καταγραφεί η συνολική κίνηση του ματιού ανάμεσα στα frame ενός βίντεο. Η πρώτη μέθοδος [20] αφορούσε την αναγνώριση των ματιών σε μεγάλες εικόνες και στον διαχωρισμό των ματιών από το πρόσωπο, χρησιμοποιώντας παραδείγματα όπου υπήρχαν αντικείμενα με έντονες αντανακλάσεις (γυαλιά κ.λπ.) . Η δεύτερη μέθοδος [21] αφορά φωτογραφίες ματιών και τον διαχωρισμό της κόρης από το υπόλοιπο μάτι με την βοήθεια των αντανακλάσεων.

2.2.2.3 Εντοπισμός του ματιού με βάση το σχήμα

Οι Chen και Kubo [22] πρότειναν μία τεχνική όπου χρησιμοποιείται μια ακολουθία ανίχνευσης προσώπου και φίλτρα Gabor. Οι πιθανές περιοχές προσώπου στην εικόνα ανιχνεύονται με βάση το χρώμα του δέρματος. Στη συνέχεια, η υποψήφια περιοχή του ματιού καθορίζεται αυτόματα χρησιμοποιώντας τη γεωμετρική δομή του προσώπου.

Οι Kocejko et al. [1] πρότειναν τον αλγόριθμο ανίχνευσης μακρύτερης γραμμής για την ανίχνευση της θέσης της κόρης. Αυτός ο αλγόριθμος βασίζεται στην υπόθεση ότι η κόρη είναι κυκλική. Οι

μεγαλύτερες κάθετες και οριζόντιες γραμμές της κόρης εντοπίζονται. Το κέντρο της μακρύτερης γραμμής, μεταξύ των κάθετων και των οριζόντιων γραμμών είναι το κέντρο της κόρης.

Οι Khairosfaizal και Nor'aini [23] παρουσίασαν ένα απλό σύστημα εντοπισμού οφθαλμών, που βασίστηκε σε κυκλικό μετασχηματισμό Hough και τον εφάρμοσαν σε εικόνες προσώπου. Το πρώτο βήμα είναι η ανίχνευση της περιοχής προσώπου, η οποία πραγματοποιείται με μια υπάρχουσα μέθοδο ανίχνευσης. Στη συνέχεια, η αναζήτηση για το μάτι βασίζεται στο κυκλικό σχήμα του ματιού και στην ανίχνευσή του από τον μετασχηματισμό Hough.

Οι Sundaram et al. [5] πρότειναν μια μέθοδο εντοπισμού της ίριδας που προσδιορίζει τα εξωτερικά και εσωτερικά όρια της ίριδας. Η διαδικασία περιλαμβάνει δύο βασικά βήματα: ανίχνευση σημείων ακμής και τον κυκλικό μετασχηματισμό Hough.

2.3 Υλοποιήσεις στο Hardware

2.3.1 Android

Όσον αφορά τις υλοποιήσεις σε Android μελετήθηκαν δύο διπλωματικές εργασίες από φοιτητές του Πολυτεχνείου Κρήτης. Η Σπύρου Αγγελική [24] παρουσίασε μία βιομετρική ανάλυση της ίριδας σε εφαρμογή Android, ενώ ο Σιαχάμης Χαράλαμπος [25] παρουσίασε μελέτη του φωτισμού που χρειάζεται η ίριδα προκειμένου να ανιχνευθεί από μία εφαρμογή Android.

2.3.2 FPGA

Για τον εντοπισμό της ίριδας είναι διαθέσιμοι πολλοί αλγόριθμοι αλλά οι υλοποιήσεις τους στο Hardware δεν είναι το ίδιο εύκολα διαθέσιμες. Πολλές υλοποιήσεις έχουν γίνει σε Fpga όμως στις

περισσότερες δεν παρουσιάζεται συγκεκριμένα η ανίχνευση της ίριδας, αλλά κυρίαρχα το σύνολο της εφαρμογής που αυτή εντάσσεται. Ο εντοπισμός της ίριδας εφαρμόστηκε σε FPGA στο [26] και χρειάστηκε χρόνος εκτέλεσης 159 ms, αλλά δεν αποκαλύφθηκε ούτε ο hardware σχεδιασμός αρχιτεκτονικής του εντοπισμού, ούτε η αξιολόγηση της ακρίβειας. Οι López et al. στο [26] χρησιμοποίησαν τον αλγόριθμο Daugman, βασισμένο σε μοντέλα IDRO (Intero-Differential Operator) και σε ενεργά μοντέλα περιγράμματος, προκειμένου να υλοποιήσουν την Hardware σχεδίαση του εντοπισμού της ίριδας. Σε μια πιο πρόσφατη εργασία, το [27] παρουσιάζει τον σχεδιασμό αρχιτεκτονικής και την υλοποίηση του CHT για το σύστημα ανίχνευσης ορίων της ίριδας σε πραγματικό χρόνο, αλλά αυτή η εργασία δεν περιλαμβάνει την υλοποίηση του χάρτη ακμής. Η σχεδίαση του CHT [27] τρέχει στην FPGA σε 5 ms, για τον εντοπισμό κύκλου στο χάρτη ακμών της εικόνας του ματιού. Εναλλακτικές υλοποιήσεις του CHT σε Hardware είναι οι [28], [29], ενώ μελετήθηκαν και πιο συνολικές υλοποιήσεις σε Fpga [30], [31] που όμως εξειδικεύονται προς μία συγκεκριμένη εφαρμογή.

2.4 Εκμάθηση του Xilinx System Generator

Σημαντικό κομμάτι της σχετικής έρευνας ήταν και η εκμάθησή του Xilinx System Generator και για αυτό το σκοπό μελετήθηκαν τα εγχειρίδια της Xilinx και της Matlab, αλλά και εργασίες που έχουν γίνει με το συγκεκριμένο εργαλείο και αφορούν την επεξεργασία εικόνας [32], [33], [34].

3 Μοντελοποιηση Αλγοριθμων και Θεωρητικο Υποβαθρο

Βασικό κομμάτι της παρούσας διπλωματικής εργασίας είναι η μελέτη και η ανάλυση ποικίλων τεχνικών και αλγόριθμων, που σχετίζονται με την ανίχνευση της τοποθεσίας της κόρης του ματιού. Έχοντας θέσει εκ των προτέρων σαν στόχο, η διπλωματική εργασία μας να καταλήξει σε μία πρόταση για υλοποίηση στο Hardware, οι αλγόριθμοι που εξετάστηκαν και μελετήθηκαν θα αξιολογηθούν σύμφωνα με αυτό τον σκοπό.

Για αυτό το λόγω χρησιμοποιήθηκε η Matlab που ενδείκνυται για την μελέτη και την αξιολόγηση αλγορίθμων, ειδικότερα για τεχνικές που αφορούν τον τομέα της επεξεργασίας εικόνας, αφού προσφέρει άμεση και γρήγορη ανάλυση, όπως επίσης και οπτική απεικόνιση των αποτελεσμάτων. Ειδικότερα μελετήσαμε κάποιους γνωστούς αλγόριθμους, προκειμένου να λάβουμε συμπεράσματα για την αποτελεσματικότητά τους, όπως επίσης υλοποιήσαμε και την δικιά μας μοντελοποίηση, προκειμένου να διέπεται περισσότερο από την λογική της αξιοποίησής της, στο Hardware.

Επιπλέον ιδιαίτερη αναφορά πρέπει να γίνει και στο Dataset εικόνων στο οποίο δοκιμάσαμε τους διάφορους αλγόριθμους. Η φύση του προβλήματος, δηλαδή η εύρεση της τοποθεσίας κόρης, μας αφήνει το περιθώριο να εξειδικεύσουμε το Dataset. Έτσι χρησιμοποιήσαμε το Dataset Casia

Lamp, το οποίο αποτελείται από εικόνες NIR. Σε αυτή μας την επιλογή συνέβαλαν, και το ότι είναι αρκετά δύσκολο να βρεθεί διαθέσιμο Dataset με φωτογραφίες ματιών, αλλά και κυρίαρχα το ότι η τεχνολογία NIR είναι αυτή που χρησιμοποιείται σε τέτοιου είδους εφαρμογές σήμερα, καθώς με την χρήση της αποφεύγονται πολλά προβλήματα όπως ο φωτισμός, οι αντανακλάσεις κλπ.

3.1 Θεωρητική Μελέτη Αλγορίθμων

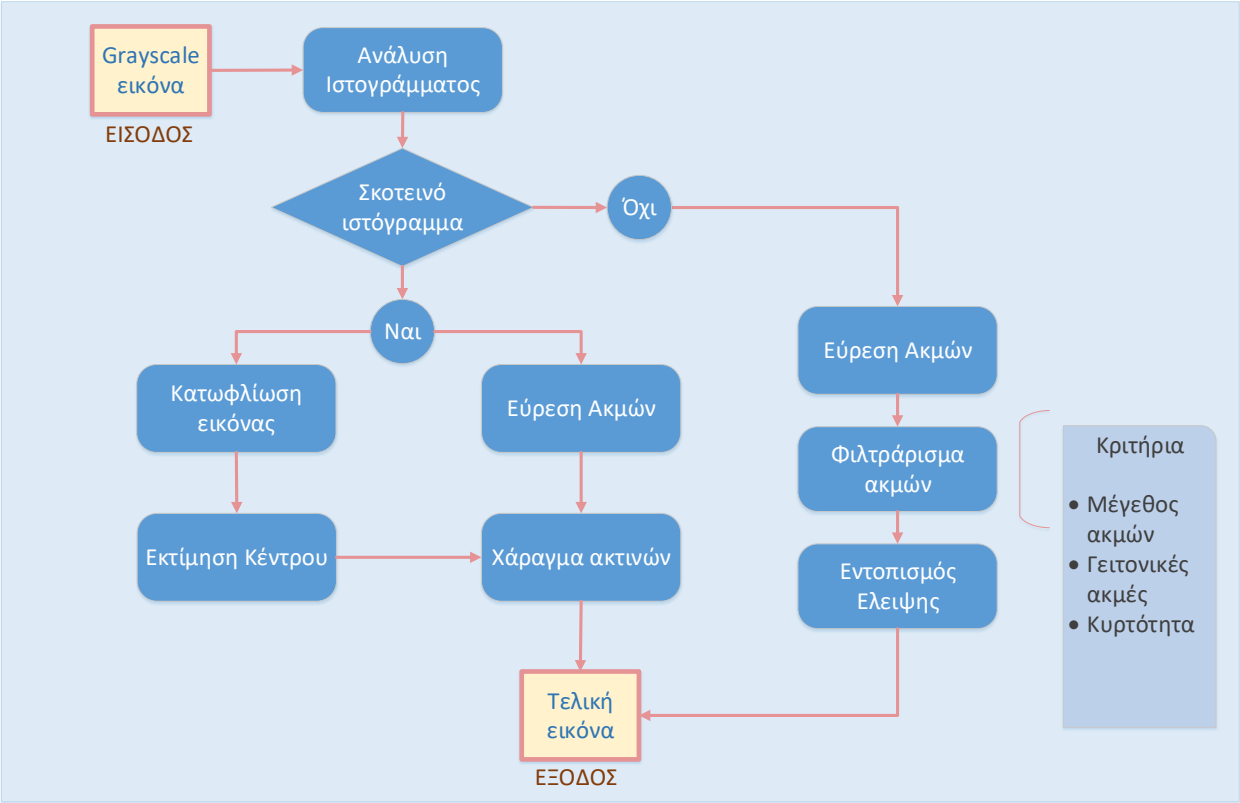
Πρώτο βήμα για να αρχίσει η μελέτη πάνω στο πρόβλημα που έχει τεθεί, αυτό της εύρεσης της θέσης της κόρης, είναι να εξεταστούν οι ήδη υπάρχοντες αλγόριθμοι. Αυτό έχει σαν στόχο να τους κρίνουμε, για το αν κάνουν συγκεκριμένα στην δικιά μας περίπτωση και στο σύστημα που έχουμε στόχο να υλοποιήσουμε, καθώς και να τους υλοποιήσουμε με κάποια προσέγγιση πιο φιλική προς το Hardware. Για αυτόν τον σκοπό εξετάστηκαν 6 αλγόριθμοι.

3.1.1 Αλγόριθμος Excuse

Ο αλγόριθμος ExCuSe βασίζεται στην ανίχνευση ακμών και σε μορφολογική επεξεργασία. Η αλγοριθμική ροή παρουσιάζεται στο σχήμα 3.1. Στο πρώτο βήμα, η εικόνα που μπαίνει σαν είσοδος, κανονικοποιείται και υπολογίζεται το ιστόγραμμα της. Αν βρεθούν υψηλές τιμές στην περιοχή του φωτεινού ιστογράμματος, η κόρη μπορεί να βρεθεί με βάση την ανάλυση των ακμών. Προκειμένου να επιτευχθεί αυτό, ένας ανιχνευτής ακμών Canny εφαρμόζεται στην εικόνα Οι ακμές που προκύπτουν φιλτράρονται, αφαιρώντας τις λεπτές γραμμές και χρησιμοποιώντας αραίωση στις παχιές ακμές με τη βοήθεια μορφολογικής επεξεργασίας. Όλες οι υπόλοιπες ακμές εξομαλύνονται και οι ορθογώνιες ακμές αφαιρούνται χρησιμοποιώντας μορφολογικά πρότυπα. Για κάθε συνδεδεμένη γραμμή, η μέση θέση υπολογίζεται. Με βάση αυτές τις πληροφορίες, οι ευθείες γραμμές εξαιρούνται από την περαιτέρω επεξεργασία. Όλες οι υπόλοιπες καμπύλες

γραμμές διατηρούνται και επεξεργάζονται περεταίρω. Για κάθε καμπύλη, υπολογίζεται η μέση τιμή έντασης και η καμπύλη με τη χαμηλότερη τιμή επιλέγεται ως καμπύλη της κόρης. Στη συνέχεια, μια έλλειψη εφαρμόζεται σε αυτή την καμπύλη.

Figure 3.1: Διάγραμμα ροής Excuse Algorithm



Σε περίπτωση που δεν υπάρχουν υψηλές τιμές στην περιοχή του φωτεινού ιστογράμματος, ακολουθούνται τα εξής βήματα. Κατά 'αρχάς προσδιορίζεται από τον αλγόριθμο μια προσεγγιστική θέση κόρης και στη συνέχεια βελτιώνεται αυτό σταδιακά, προσεγγίζοντας το

κέντρο της κόρης. Η προσεγγιστική θέση της κόρης εκτιμάται με βάση την επεξεργασία γωνιακής ενσωματωμένης προβολής στην εικόνα, αφού έχει καταωφλιωθεί πρώτα. Πιο συγκεκριμένα, η εικόνα περνάει από ένα κατώφλι και τα Pixel πάνω από το όριο του κατωφλιού, αθροίζονται κατά μήκος των γραμμών. Αυτή η διαδικασία γίνεται τέσσερις φορές. Μόλις πραγματοποιηθεί μια προσεγγιστική εκτίμηση του κέντρου κόρης, χρησιμοποιείται σαν πληροφορία στην εικόνα που έχουν βρεθεί οι ακμές, έπειτα από Canny Edge Detection. Από το προσεγγιστικό κέντρο χαράσσονται ακτίνες προς τα εξωτερικά μέρη της εικόνας. Η καμπύλη στην οποία συμπίπτουν οι ακτίνες είναι οι καμπύλη της κόρης και τα σημεία επαφής χρησιμοποιούνται για να υπολογιστεί η τελική έλλειψη και να βρεθεί το τελικό αποτέλεσμα.

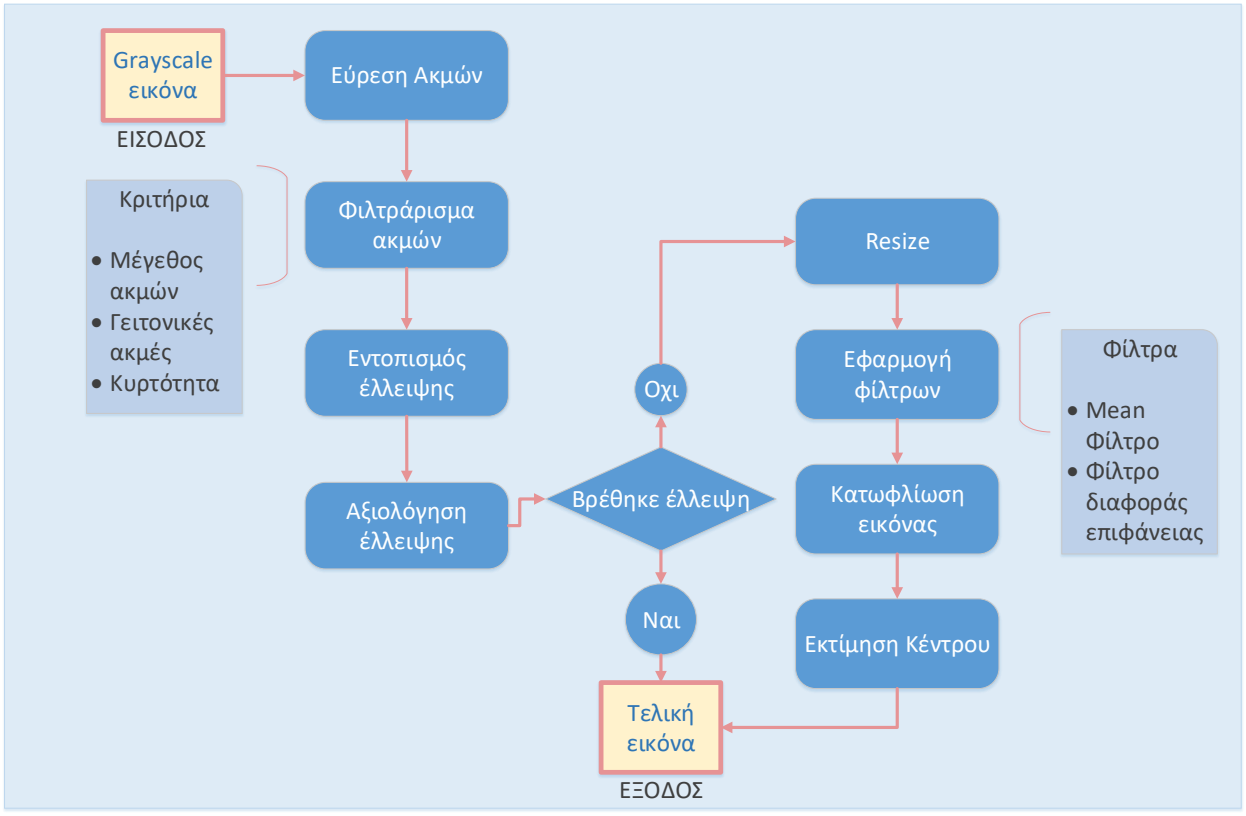
3.1.2 Αλγόριθμος Else

Παρόμοια με τον αλγόριθμο ExCuSe, ο αλγόριθμος Else λειτουργεί, εφαρμόζοντας Canny Edge Detection σε εικόνες με μάτια. Το κέντρο της κόρης αρχικά, βρίσκεται κατά προσέγγιση. Έπειτα συνδέσεις ακμών που δεν είναι χρήσιμες στο να περιγράψουν την έλλειψη, που αναφέρεται στην κόρη, σβήνονται με μορφολογικό τρόπο. Οι ακμές, συλλέγονται και αξιολογούνται με βάση κριτήρια, όπως η κυρτότητα και η δυνατότητα τοποθέτησης ελλείψεων πάνω τους,. Επομένως γίνεται η προσπάθεια να εφαρμοστεί έλλειψη πάνω στις ακμές. Στην περίπτωση που βρεθεί έλλειψη, που μετά την αξιολόγηση της καλύπτει τα ζητούμενα, τότε έχει εντοπιστεί το τελικό αποτέλεσμα.

Σε περίπτωση που δεν εντοπιστεί έλλειψη, όταν φιλτράρονται οι ακμές και οι εναπομείναντες δεν πληρούν τις προϋποθέσεις, διεξάγεται μια δεύτερη ανάλυση. Πιο συγκεκριμένα, όπως και στον ExCuSe, ο ElSe εκτιμά πρώτα μια πιθανή υποψήφια θέση και στη συνέχεια βελτιώνει αυτή τη θέση. Αρχικά η εικόνα γίνεται Resize προκειμένου να περιοριστεί ο χρόνος εκτέλεσης των παρακάτω διεργασιών. Έπειτα εφαρμόζεται συνέλιξη με δύο διαφορετικά φίλτρα: (1) ένα φίλτρο διαφοράς επιφανείας για τον υπολογισμό της περιοχής - διαφορά μεταξύ ενός εσωτερικού κύκλου

και ενός περιβάλλοντος ευθείων ακμών - (2)και ένα mean φίλτρο. Τα αποτελέσματα και των δύο συνελίξεων πολλαπλασιάζονται και αφού το αποτέλεσμα γίνει rescale στο μέγεθος της αρχικής εικόνας, θα αποτελέσει βάση για τα επόμενα βήματα. Η εικόνα που προκύπτει περιγράφει την περιοχή της κόρης σαν την πιο φωτεινή – λευκή περιοχή. Έτσι κάνοντας κατωφλίωση με κατάλληλο κατώφλι προκύπτει μόνο η περιοχή της κόρης από την οποία μπορεί να εκτιμηθεί το κέντρο της, φτάνοντας στο τελικό αποτέλεσμα.

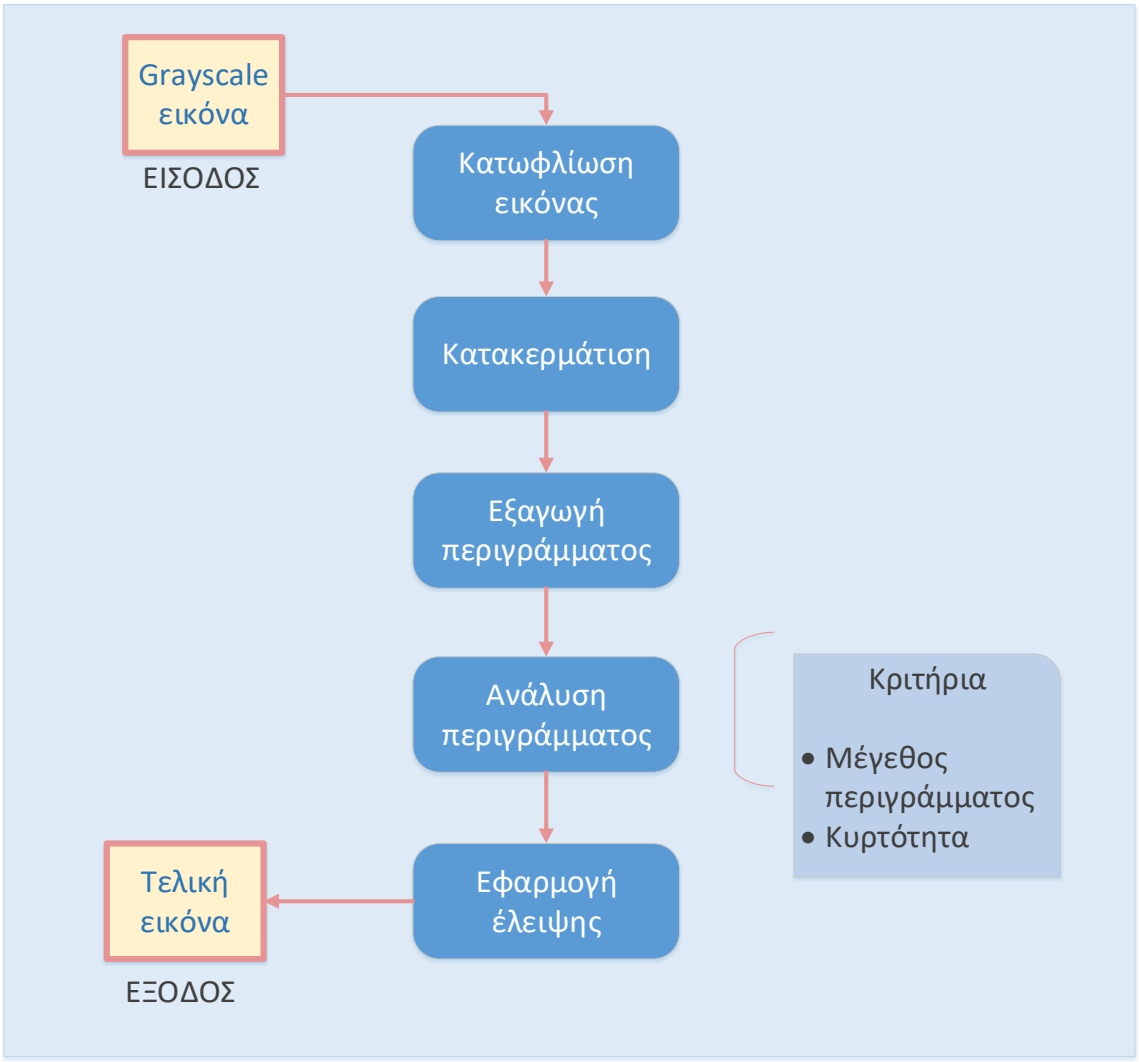
Figure 3.2: Διάγραμμα ροής Else Algorithm



3.1.3 Αλγόριθμος Set

Η προσέγγιση SET αποτελείται από ένα συνδυασμό προσαρμόσιμων από τον χρήστη και αυτόματων εκτιμήσεων βήμα – βήμα, προκειμένου να βρεθεί το κέντρο της κόρης. Πριν από την ανίχνευση της κόρης, δύο παράμετροι, δηλαδή ένα κατώφλι για τη μετατροπή της εισόδου σε μια δυαδική εικόνα και το μέγεθος των τμημάτων που εξετάζονται για την ανίχνευση των κόρων, ρυθμίζονται από τον χρήστη. Η κατωφλιωμένη εικόνα κατατάσσεται αρχικά και τα εικονοστοιχεία ταξινομούνται στη συνέχεια σε μέγιστες συνδεδεμένες περιοχές, για να βρεθούν υποψήφιες κόρες. Για κάθε τμήμα μεγαλύτερο από μία τιμή κατωφλίου, εντοπίζεται το περιθωριακό τμήμα. Στο τελευταίο βήμα, προσαρμόζεται έλλειψη σε κάθε εξαγόμενο τμήμα. Η έλλειψη που είναι πλησιέστερη στον κύκλο θεωρείται ως η κόρη του ματιού.

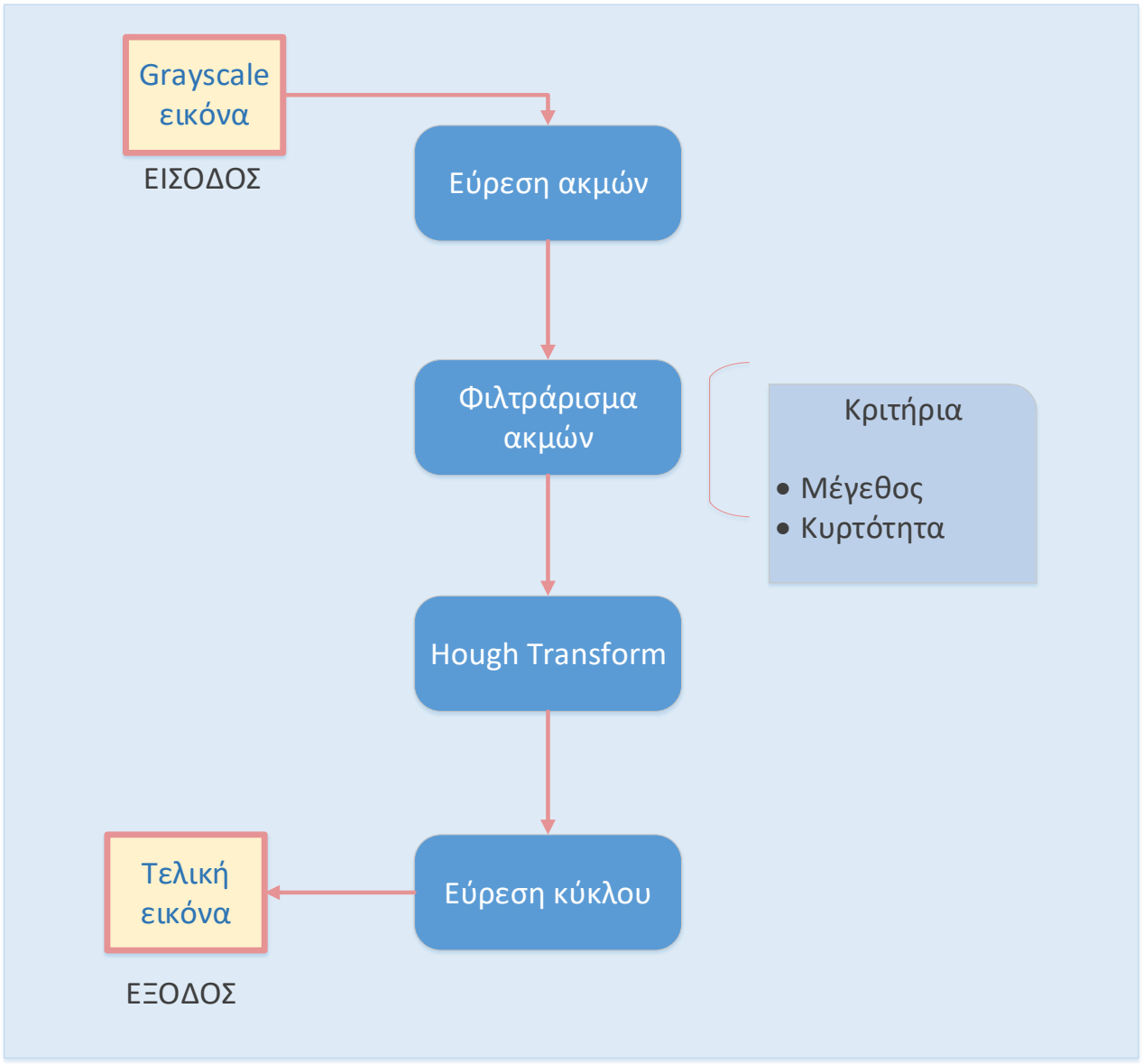
Figure 3.3: Διάγραμμα ροής Set Algorithm



3.1.4 Αλγόριθμος Pupil Labs

Ο ανιχνευτής Pupil Labs είναι ενσωματωμένος σε μία ανοικτή πλατφόρμα παρακολούθησης της κόρης του ματιού και απαρτίζεται από αρκετά βήματα επεξεργασίας. Σε ένα πρώτο βήμα, η εικόνα του ματιού μετατρέπεται σε κλίμακα του γκρι. Η αρχική εκτίμηση της περιοχής της κόρης εντοπίζεται μέσω της ισχυρότερης απόκρισης για ένα χαρακτηριστικό κέντρο. Ο αλγόριθμος στη συνέχεια χρησιμοποιεί ένα φίλτρο Canny για να ανιχνεύσει τις ακμές στην εικόνα του οφθαλμού και να φιλτράρει αυτές τις ακμές με βάση την γειτονική ένταση των Pixel. Στη συνέχεια, αναζητά πιο σκοτεινές περιοχές, ενώ το σκοτάδι καθορίζεται με τη χρήση μιας οριζόντιας μετατόπισης της χαμηλότερης κορυφής στο ιστόγραμμα των εντάσεων των Pixel στην εικόνα του οφθαλμού. Οι ακμές ύστερα συλλέγονται, φιλτράρονται και με αυτόν τον τρόπο εξαιρούνται εκείνες που προέρχονται από φασματικές αντανακλάσεις. Έτσι εξάγονται σε περιγράμματα χρησιμοποιώντας connected components. Τα περιγράμματα απλοποιούνται και στη συνέχεια φιλτράρονται και χωρίζονται σε υπό-περιγράμματα βάσει κριτηρίων συνέχειας καμπυλότητας. Έπειτα εφόσον οι ακμές έχουν ελαχιστοποιηθεί εφαρμόζεται κυκλικός μετασχηματισμός Hough προκειμένου να εντοπιστεί ο τελικός κύκλος που περιγράφει την κόρη του ματιού.

Figure 3.4: Διάγραμμα ροής Pupil Labs

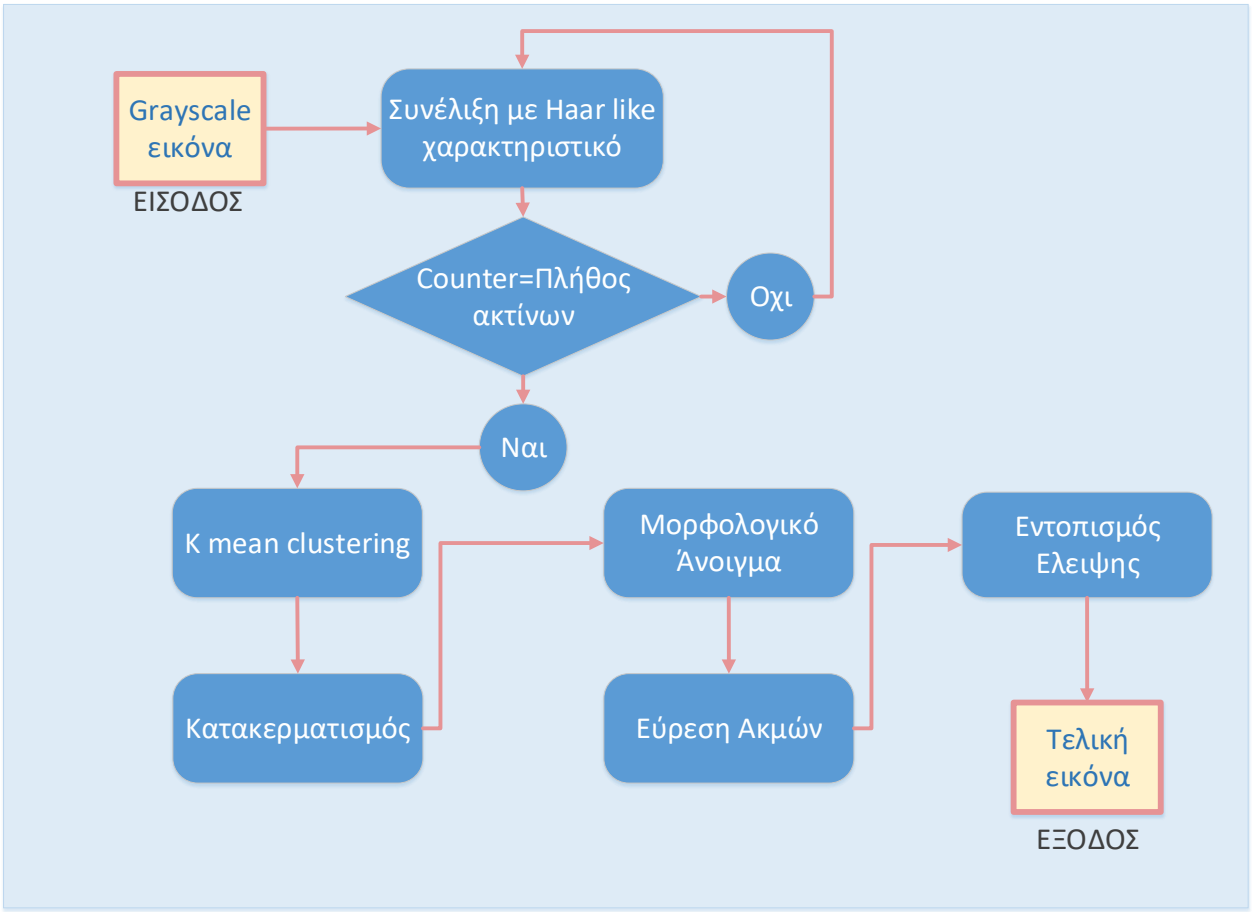


3.1.5 Αλγόριθμος Swirski

Ο αλγόριθμος Swirski λειτουργεί σε τρία κύρια στάδια. Για να βρει την κόρη, ο «ανιχνευτής Swirski κάνει συνέλιξη την εικόνα με ένα χαρακτηριστικό Haar-like, Ο αλγόριθμος επαναλαμβάνει αυτή την συνέλιξη για ένα σύνολο πιθανών ακτινών ανάμεσα σε ένα καθορισμένο, από τον χρήστη, ελάχιστο και μέγιστο. Η θέση της ισχυρότερης απόκρισης είναι το εκτιμώμενο κέντρο της κόρης, με το μέγεθος της περιοχής να καθορίζεται από την αντίστοιχη ακτίνα. Η αρχική εκτίμηση περιφέρειας είναι απίθανο να επικεντρώνεται με ακρίβεια στην κόρη. Επομένως, στο δεύτερο βήμα, ο αλγόριθμος προσεγγίζει τη θέση της κόρης μέσα σε αυτήν την περιοχή με βάση τη συσσώρευση K-mean: Το ιστόγραμμα εικόνας χωρίζεται σε σκούρο (εικονοστοιχεία κόρης) και φως.

Ο αλγόριθμος στη συνέχεια δημιουργεί μια τμηματική δυαδική εικόνα της περιοχής της κόρης με το να οριοθετεί οποιοδήποτε Pixel, πάνω από τη μέγιστη ένταση, στο σκοτεινό σύμπλεγμα. Στη συνέχεια, βρίσκονται τα συνδεδεμένα στοιχεία στην κατατμημένη εικόνα και το μεγαλύτερο από αυτά επιλέγεται ως κόρη. Το κέντρο μάζας αυτού του συστατικού προσεγγίζει τη θέση της κόρης. Το τελικό στάδιο του αλγορίθμου βελτιώνει την εκτίμηση της θέσης, προσαρμόζοντας μια έλλειψη στο όριο μεταξύ της κόρης και της ίριδας. Για την αφαίρεση χαρακτηριστικών όπως οι βλεφαρίδες και οι λάμψεις, εκτελείται μια μορφολογική "ανοιχτή" λειτουργία για να κλείνει μικρά φωτεινά κενά στην περιοχή της σκοτεινής κόρης, χωρίς να επηρεάζεται σημαντικά το περίγραμμα. Ο αλγόριθμος βρίσκει έπειτα το όριο μεταξύ κόρης και ίριδας χρησιμοποιώντας ανιχνευτή ακμών Canny. Τέλος, ο αλγόριθμος ταιριάζει με έλλειψη στα ακραία Pixel.

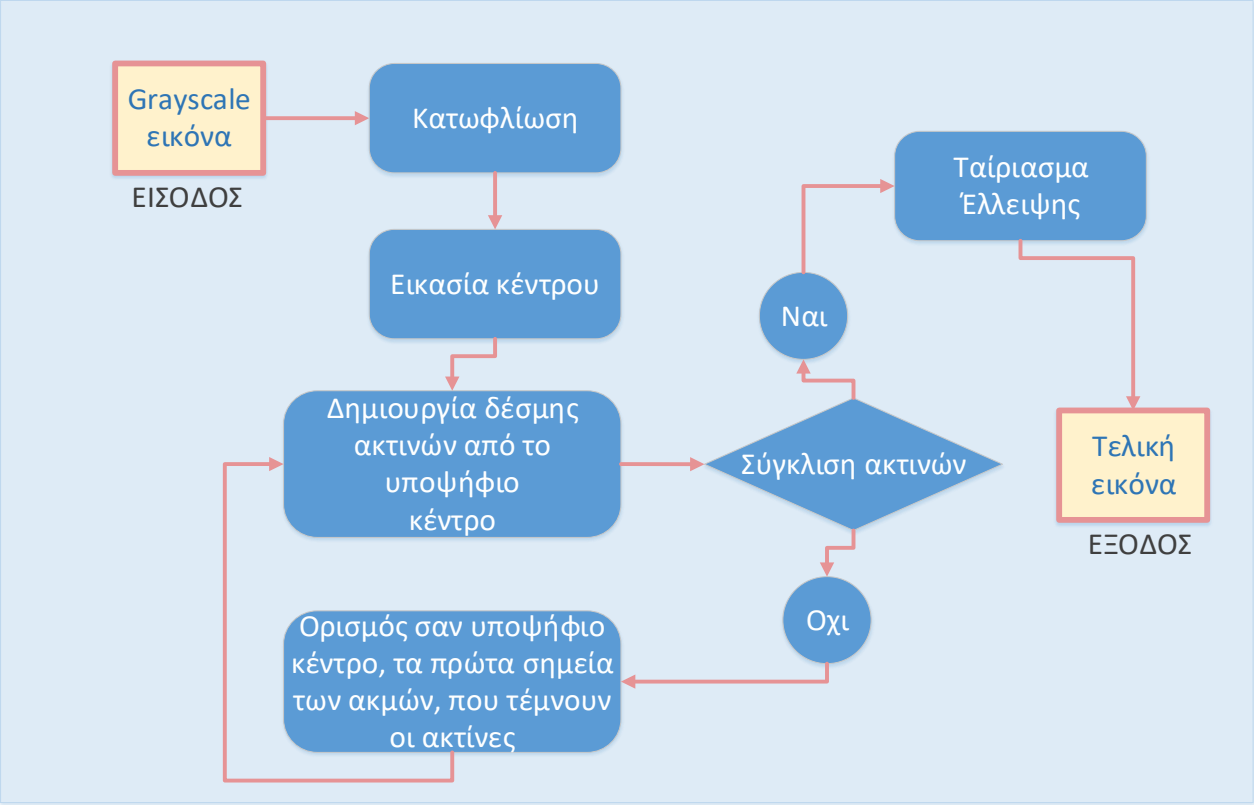
Figure 3.5: Διάγραμμα ροής Swirski



3.1.6 Αλγόριθμος Starbust

Στο πρώτο βήμα του Starburst, η εικόνα ξεθωριάζει χρησιμοποιώντας ένα Gaussian φίλτρο. Ο αλγόριθμος χρησιμοποιεί στη συνέχεια προσαρμοστικό κατώφλι σε μια τετραγωνική περιοχή ενδιαφέροντος σε κάθε πλαίσιο, για να εντοπίσει την αντανάκλαση του κερατοειδούς. Η θέση του κερατοειδούς δίνεται από το γεωμετρικό κέντρο της μεγαλύτερης περιοχής στην εικόνα, χρησιμοποιώντας το καθορισμένο κατώφλι. Η ακτινική παρεμβολή χρησιμοποιείται περαιτέρω για την αφαίρεση της αντανάκλασης του κερατοειδούς από την εικόνα του οφθαλμού. Το κεντρικό βήμα του αλγορίθμου, που έδωσε επίσης σε αυτή τη μέθοδο το όνομά του, είναι η εκτίμηση του περιγράμματος της κόρης, ανιχνεύοντας ακμές κατά μήκος ενός περιορισμένου αριθμού ακτινών, που εκτείνονται από μια κεντρική καλύτερη εικασία του κέντρου της. Οι ακτίνες είναι ανεξάρτητες και αξιολογούνται ανά Pixel, μέχρις ότου να σημειωθεί υπέρβαση ενός ορίου, υποδεικνύοντας την ακμή της κόρης. Ένα χαρακτηριστικό σημείο ορίζεται σε εκείνη την θέση ως υποψήφιο άκρο περιγράμματος και η επεξεργασία κατά μήκος της ακτίνας διακόπτεται. Για κάθε υποψήφια κόρη δημιουργείται μια άλλη δέσμη ακτινών που δημιουργεί ένα δεύτερο σύνολο υποψηφίων περιγράμματος κόρης. Η διαδικασία αυτή επαναλαμβάνεται μέχρι τη σύγκλιση. Η τοποθέτηση του μοντέλου τελικά εκτελείται ακολουθώντας ένα παράδειγμα συνεννόησης τυχαίου δείγματος (RANSAC) για να βρεθεί η καλύτερη ελλειπτική εφαρμογή που περιγράφει το όριο της κόρης

Figure 3.6: Διάγραμμα ροής Starbust



3.2 Θεωρητική Αξιολόγηση και Υλοποίηση Αλγορίθμων Set και Pupil Labs

Προκειμένου να προχωρήσουμε στο επόμενο βήμα οι παραπάνω αλγόριθμοι έπρεπε να αξιολογηθούν θεωρητικά προκειμένου να δουλέψουμε και να εφαρμόσουμε εκείνους που θα μας

δώσουν χρήσιμα συμπεράσματα για να δημιουργήσουμε την δικιά μας μοντελοποίηση. Το σύστημα μας έχει την λογική του εντοπισμού της κόρης σε συγκεκριμένες συνθήκες. Επίσης αφού έχουμε συγκεκριμενοποιήσει τις συνθήκες μπορούμε με αυτό τον τρόπο να κατευθυνθούμε σε πιο απλές λύσεις που δεν θα μας στερούν αποδοτικότητα, θα μας εξασφαλίζουν όμως μικρότερο κόστος και μικρότερο χρόνο εκτέλεσης.

Υπό αυτή την έννοια απορρίψαμε τους αλγόριθμους όπως ο Else και ο Excuse που έχουν κυρίαρχο στόχο να ανιχνεύσουν την θέση της κόρης του ματιού σε φωτογραφίες που μπορεί να είναι κακής ποιότητας ή να έχουν κακό φωτισμό και μεγάλες αντανακλάσεις. Θέλοντας να μειώσουμε τον χρόνο εκτέλεσης, μπορούμε να κερδίσουμε σε αυτά που μας προσφέρουν αυτοί οι δύο αλγόριθμοι ρυθμίζοντας άλλες παραμέτρους.

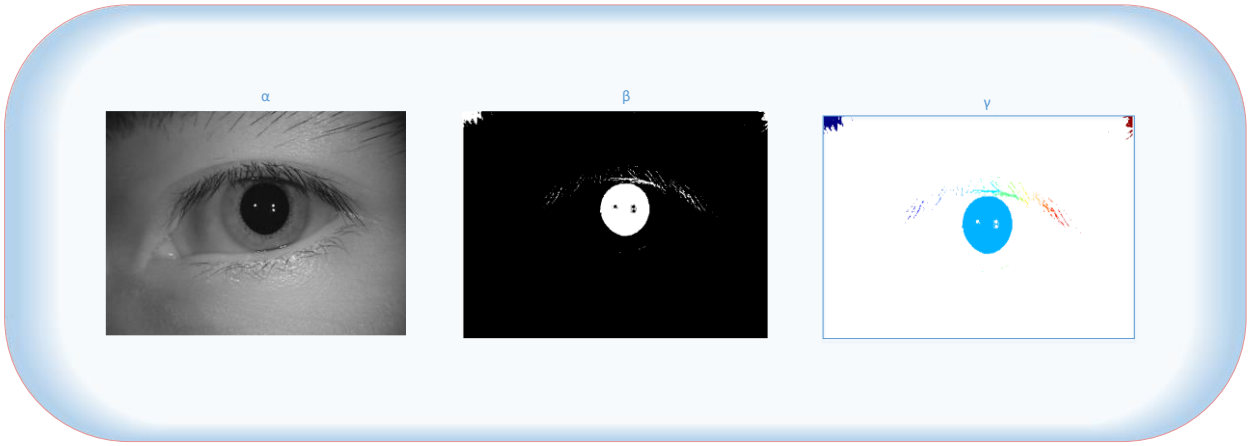
Με την ίδια λογική απορρίφθηκε ο αλγόριθμος Swirski, καθώς μεγάλο μέρος του αφορά τον εντοπισμό της περιοχής της κόρης, πράγμα που αν προσαρμόσουμε την απόσταση που μπορεί να λαμβάνεται η εικόνα, για ένα σύστημα υπο-βοήθειας ατόμων με κινητικές δυσκολίες για παράδειγμα, δεν χρειάζεται να εντάσσεται στον αλγόριθμο, εξασφαλίζοντάς μας, μία πιο εύκολη και ελαφριά υλοποίηση. Επιπλέον ο Αλγόριθμος Starbust έχει αυξημένη πολυπλοκότητα καθώς χρειάζεται να προσπελασθεί η εικόνα αρκετές φορές και κάθε φορά να εφαρμόζονται μαθηματικές πράξεις πράγμα που για το Hardware είναι κάτι που θέλουμε να αποφύγουμε.

Οι αλγόριθμοι Set και Pupil Labs Μπορούν να χαρακτηριστούν ως οι πιο απλοί και φιλικοί προς το Hardware, χωρίς να υστερούν σε αποτελεσματικότητα σε παραδείγματα εικόνων, όπως αυτά που έχουμε στο dataset. Επομένως αυτοί επιλέχτηκαν προκειμένου να εξεταστούν στην Matlab και από αυτούς να βγουν συμπεράσματα για την υλοποίηση που θα ακολουθήσουμε στην συνέχεια.

Πιο συγκεκριμένα ο αλγόριθμος Set υλοποιήθηκε σύμφωνα με τα πρότυπα που μας δόθηκαν και περιγράφονται παραπάνω. Βασίζεται σε μεγάλο βαθμό στον κατακερματισμό της εικόνας και στην ομαδοποίηση των Pixel σε υπό ομάδες. Έτσι αφού έχουμε τις ομάδες των pixel ομαδοποιημένες σύμφωνα με το κριτήριο των Connected Components μπορούμε να εξάγουμε συμπεράσματα για το περίγραμμα τους και το κατά πόσο ικανοποιεί κάποιο κριτήριο κυρτότητας και πιο συγκεκριμένα αν το περίγραμμα ικανοποιεί το ποσοστό της εξίσωσης του κύκλου που έχει οριστεί. Εν τέλει στην ομάδα, που το περίγραμμά της ταιριάζει με τα κριτήρια και τις παραμέτρους που έχουμε ορίσει, ταυτίζεται πάνω της μία έλλειψη η οποία εν τέλει περιγράφει την κόρη του ματιού.

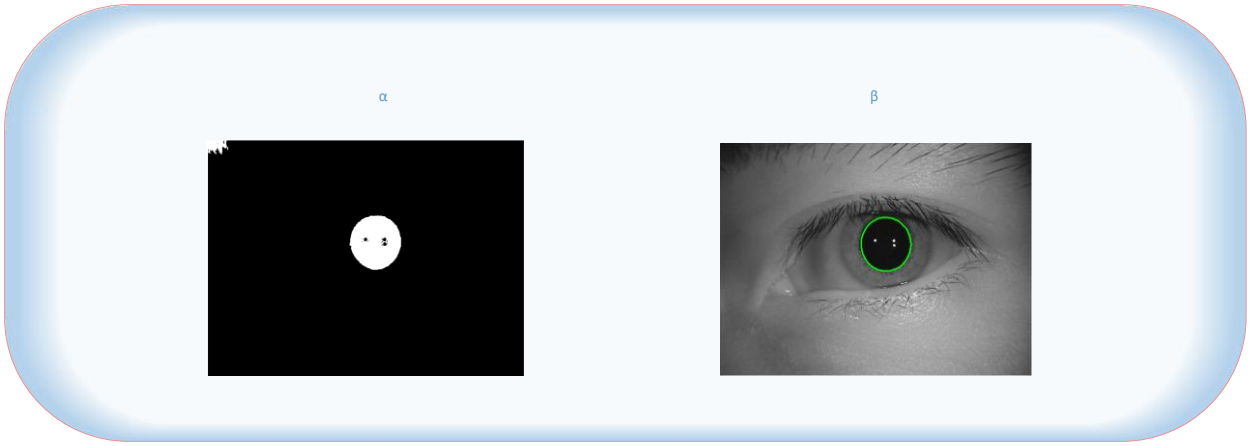
Όσον αφορά τον κατακερματισμό της εικόνας, στην μηχανική όραση ορίζεται ως η διαδικασία που χωρίζει μια ψηφιακή εικόνα σε πολλαπλά σύνολα Pixel, (super-pixels). Ο στόχος της είναι να απλοποιήσει και / ή να αμβλύνει την αναπαράσταση μιας εικόνας σε κάτι που είναι πιο σημαντικό και πιο εύκολο να αναλυθεί. Αυτή η τεχνική συνήθως χρησιμοποιείται για τον εντοπισμό των περιγραμμάτων, που περιγράφουν διάφορα αντικείμενα καθώς και το σχήμα τους (γραμμές, καμπύλες) στις εικόνες. Στην δική μας περίπτωση χρησιμοποιείται για να εξαχθούν τα αντικείμενα και να αναλυθούν, αρχικά σύμφωνα με το μέγεθός τους, φιλτράροντας αυτά που ξεπερνάνε ένα κατώφλι. Έτσι αρχικά γίνεται κατωφλίωση της εικόνας προκειμένου να μετατραπεί σε δυαδική. Στην συνέχει χρησιμοποιούμε την συνάρτηση Bwlabel της Matlab προκειμένου να ομαδοποιήσουμε τα διάφορα αντικείμενα που απαρτίζονται από Pixel συνδεδεμένα μεταξύ τους. Όλη αυτή η διαδικασία περιγράφεται στο παρακάτω σχήμα.

Figure 3.7: Segmentation Set Algorithm



Στην συνέχεια σκοπός του αλγόριθμου είναι πάνω στα αντικείμενα που έχουν απομείνει να εντοπιστεί η κυκλική κόρη του ματιού. Για αυτό τον σκοπό πρώτα θα αφαιρεθούν οι ομάδες που αποτελούνται από λίγα ή από πολλά pixel. Για να γίνει αυτό μας δίνεται η δυνατότητα, επειδή ξέρουμε πόσο περίπου θα είναι η κόρη του ματιού, λόγω της αρχικοποίησης του συστήματός μας. Στις ομάδες που απομένουν θα γίνει εξαγωγή του περιγράμματος και σύμφωνα με τις ιδιότητες της συνάρτησης Bwlabel μπορούμε να χρησιμοποιήσουμε την πληροφορία που μας δίνει για κάθε ομάδα. Έτσι θα χρησιμοποιηθεί κατά πόσο το περίγραμμα ταιριάζει με την εξίσωση του κύκλου. Αν ταιριάζει πάνω από ένα κατώφλι τότε αυτή η συγκεκριμένη ομάδα είναι η κόρη του ματιού. Τέλος προσαρμόζεται πάνω της μία έλλειψη και έχουμε το τελικό αποτέλεσμα.

Figure 3.8: Ellipse Fitting Set Algorithm



Ο Αλγόριθμος Pupil Labs έχει διαφορετική λογική τόσο στο κομμάτι φιλτραρίσματος και κατακερματισμού της εικόνας αλλά ως και προς το κομμάτι εντοπισμού του κυκλικού μέρους της εικόνας. Όσον αφορά το πρώτο κομμάτι η εικόνα υφίσταται επεξεργασία προκειμένου να εξαχθούν οι ακμές. Υστέρα με περεταίρω επεξεργασία οι ακμές φιλτράρονται σύμφωνα με το μέγεθός τους αλλά και με την κυρτότητα προκειμένου να περιοριστεί σε μεγάλο βαθμό το μέγεθος της πληροφορίας του τελικού χάρτη ακμών. Για την λεπτομερή εξαγωγή των ακμών χρησιμοποιήθηκε ο ανιχνευτής ακμών Canny Edge Detection και μία δικιά μας υλοποίηση του.

Η διαδικασία του αλγορίθμου ανίχνευσης ακμών Canny μπορεί να αναλυθεί σε 5 διαφορετικά βήματα:

- Εφαρμογή Gaussian φίλτρου για να εξομαλύνθεί η εικόνα και να αφαιρεθεί ο θορυβος.
- Ευρεση των Intensity Gradients της εικόνας.
- Εφαρμογή Non Maximum Suppression για να απαλοιφθούν οι ψεύτικες ακμές.
- Διπλή κατωφλίωση.
- Track edges by hysteresis για την ανίχνευση των τελικών ακμών απαλόιφωντας όλες τις άλλες ακμές που είναι αδύναμες και δεν συνδέονται με ισχυρές.
-

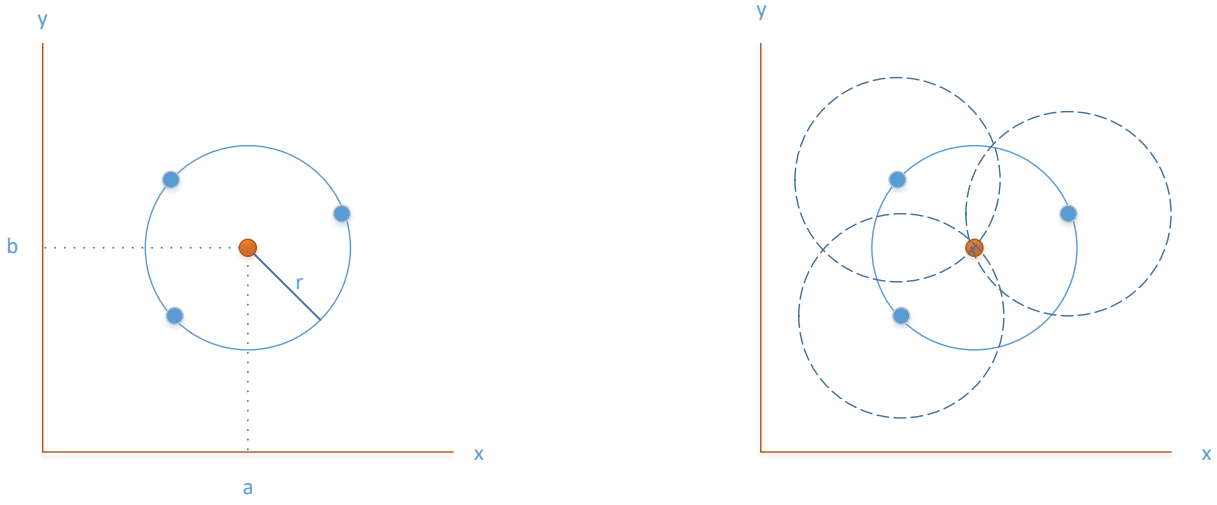
Figure 3.9: Αποτελέσματα Canny Edge Detection

Αφού εντοπιστουν οι ακμές από τον ανιχνευτή Canny τότε προχωράμε στο φιλτράρισμα του. Αυτό είναι ιδιαίτερα σημαντικο καθώς στο επόμενο στάδιο θα υλοποιηθεί Circular Hough

Transform ο οποίος είναι ιδιαίτερα απαιτητικός σε κόστος. Έτσι οι ακμές περιορίζονται σε μεγάλο βαθμο για να μείνει η πληροφορία που αρκεί για να εξαχθεί ο κύκλος στο πάρακάτω βήμα.
Ο Hough transform είναι μια τεχνική για να βρούμε σχήματα πάνω σε εικόνες.

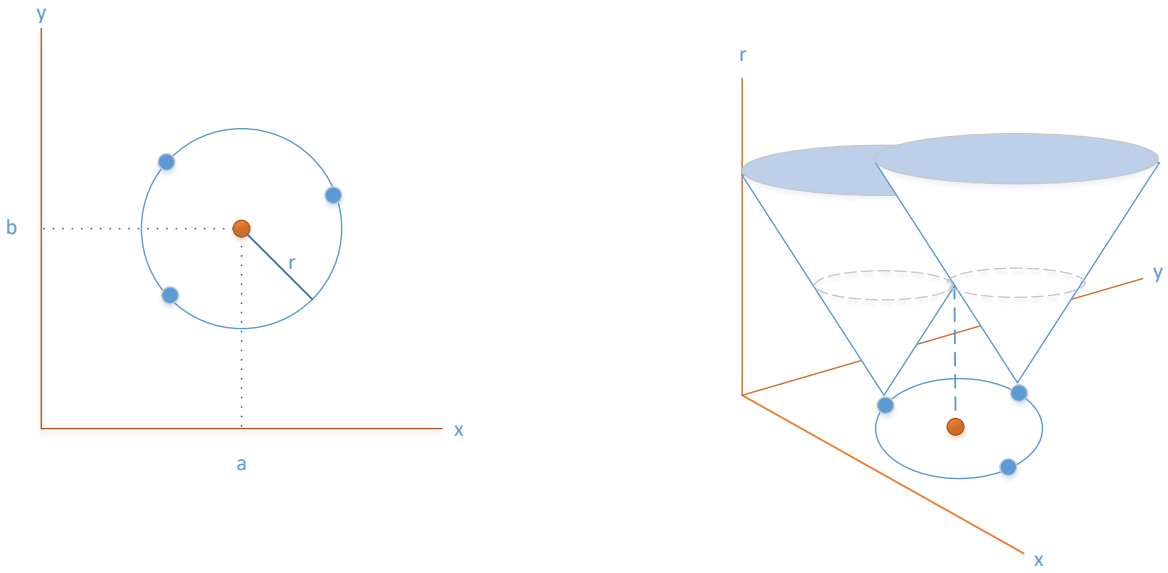
Ο Hough transform ορίζει μια χαρτογράφηση από μία εικόνα σε ένα χώρο παραμέτρων που αντιπροσωπεύεται από μια συσσωρευμένη παράταξη. Στην δικιά μας περίπτωση θα εξετάσουμε την ανίχνευση κύκλων. Έτσι θα χρησιμοποιηθεί ο κυκλικός μετασχηματισμός Hough (CHT). Ένας κύκλος μπορεί να οριστεί από τρεις παραμέτρους, τις συντεταγμένες του κέντρου του (a, b) και την ακτίνα r. Επομένως για να ισχυριστούμε ότι έχουμε γνώση για την τοποθεσία ενός κύκλου πάνω σε μία εικόνα, πρέπει να γνωρίζουμε τους παραπάνω αγνώστους. Επιπλέον ένας κύκλος με κέντρο (a,b) και ακτίνα r περιγράφεται από την εξίσωση : $(x-a)^2+(y-b)^2=r^2$. Τα χ , y είναι το σύνολο των σημείων που περιγράφουν την περιφέρεια του κύκλου και ορίζονται από τις έξεις παραμετρικές αναπαραστάσεις : x = a + r cosθ και y = b + r sinθ .

Figure 3.10: CHT με γνωστή ακτίνα



Εάν γνωρίζουμε την ακτίνα του κύκλου που θα ανιχνευθεί στην εικόνα, η παράμετρος για αναζήτηση μειώνεται σε ένα ζεύγος (a, b) και ο χώρος H είναι 2 διαστάσεων. Θεωρούμε έναν κύκλο ακτίνας R και κέντρο τα σημεία a , b , ο μετασχηματισμός για κάθε σημείο a,b,x,y στο διάστημα I αποδίδει έναν κύκλο στο χώρο H που έχει ένα κέντρο a,b,x,y και ακτίνα R. Για κάθε σημείο του κύκλου σχεδιάζεται ένας κύκλος με την αντίστοιχη ακτίνα. Οι διασταυρώσεις που θα σχηματιστούν θα αποτελούν και ψήφους για το υποψήφιο κέντρο. Το σημείο με τις περισσότερες ψήφους είναι και το τελικό κέντρο του κύκλου.

Figure 3.11: CHT με άγνωστή ακτίνα



Σε περίπτωση που δεν είναι γνωστή η ακτίνα τότε η απεικόνιση του μετασχηματισμού Hough θα είναι τρισδιάστατη. Αυτό συμβαίνει διότι η εύρεση του πιθανού κέντρου, πάλι θα γίνει με ψηφοφορία για τις διασταυρώσεις. Παρόλα αυτά αντί να χρησιμοποιηθούν κύκλοι θα χρησιμοποιηθούν κώνοι για να αποδοθεί η αναπαράσταση των διαφορετικών ακτινών. Το ύψος

των κώνων υποδηλώνει σε ποια ακτίνα βρισκόμαστε. Έτσι η ψηφοφορία γίνεται με τις διασταυρώσεις που υπάρχουν για κάθε μία πιθανή ακτίνα. Η ομάδα ακτίνας και κέντρου που θα πάρει τις περισσότερες ψήφους δίνουν την τελική πληροφορία για την τοποθεσία του κύκλου πάνω στην εικόνα. Με αυτόν τον τρόπο και με την εφαρμογή του CHT αποκτάμε εν τέλει τον τελικό κύκλο που περιγράφει το μέρος της εικόνας που αποτελεί την κόρη του ματιού και έχουμε την απαραίτητη γνώση για το που βρίσκεται το κέντρο και πόση είναι η ακτίνα του.

Ένας σχολιασμός που είναι αναγκαίος για τις παραπάνω μεθόδους είναι ότι και οι δύο σε καταστάσεις όπου δεν έχουμε ακραία παραδείγματα , όπως είναι θολές φωτογραφίες ή με ισχυρές αντανακλάσεις έχουν αρκετά καλή απόδοση και ποσοστά επιτυχίας στην τελική εύρεση του ζητουμένου.

Η υλοποίηση τους στην Matlab έβγαλε χρήσιμα συμπεράσματα για την δημιουργία της δικιάς μας μοντελοποίησης. Παρ όλα αυτά, αν θέλαμε να μεταφέρουμε αυτούς τους συγκεκριμένους αλγόριθμους στο Hardware θα ήταν αρκετά πολύπλοκο τόσο επειδή χρησιμοποιούν αρκετές συναρτήσεις της Matlab που δεν είναι το ίδιο εφικτές στο Hardware, όσο και επειδή η απόδοσή τους σε χρόνο μπορεί να βελτιωθεί σε μεγάλο βαθμό.

Βασική επιδίωξη της δικιάς μας μοντελοποίησής, είναι να απλοποιήσει την πολυπλοκότητα. Οι δύο παραπάνω μέθοδοι βασίζονται, πρώτον σε πολλαπλές προσπελάσεις της ίδιας εικόνας προκειμένου να φιλτραριστεί και να ελαχιστοποιηθεί η πληροφορία της εικόνας για να εντοπιστεί ο κύκλος και δεύτερον η βέλτιστη μέθοδος εντοπισμού κύκλου είναι ο Circular Hough Transform, ο οποίος στην ατόφια του εκδοχή έχει μεγάλη πολυπλοκότητα. Η Υλοποίησή μας φιλοδοξεί να λύσει τα δύο αυτά προβλήματα.

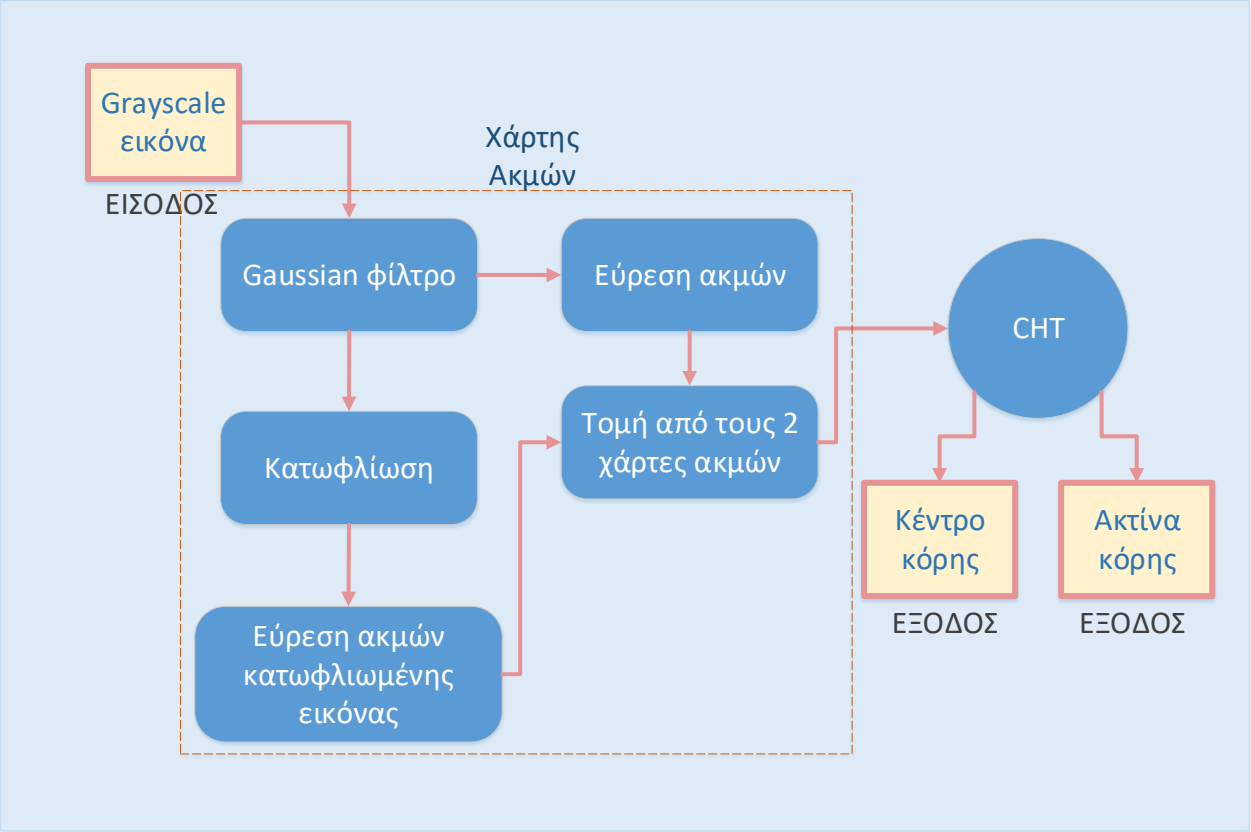
3.3 Ο Προτεινόμενος Αλγόριθμος

Αξιοποιώντας τα συμπεράσματα που βγάλαμε από την ενασχόλησή μας με τις προηγούμενες μεθόδους προχωρήσαμε στην υλοποίηση του προτεινόμενου αλγόριθμου. Βασική κατεύθυνση του αλγορίθμου μας είναι να καταφέρουμε να περιορίσουμε το κόστος και την πολυπλοκότητά του, δίχως να χάσουμε ιδιαίτερη ακρίβεια και απόδοση, στην επιτυχία της εύρεσης της κόρης του ματιού. Για τον σκοπό αυτό, προχωρήσαμε σε αλλαγές οι οποίες θα έκαναν τον αλγόριθμο πιο εύκολα μεταβιβάσιμο προς το Hardware αλλά και πιο «ελαφρύ».

Οι τεχνικές που μελετήσαμε είχαν ένα κοινό στοιχείο πάνω στο οποίο βασιστήκαμε. Και οι δύο χωρίζονται σε δύο βασικά μέρη. Στην εύρεση του χάρτη ακμών με βασικό στόχο να περιοριστούν όσο το δυνατό περισσότερο οι ακμές που θα παρέμειναν στην εικόνα και στον εντοπισμό του κυκλικού μέρους της κόρης πάνω στον χάρτη ακμών με κάποιο κριτήριο κυρτότητας των ακμών ή με κάποιο αλγόριθμο εντοπισμού κύκλου.

Στον δικό μας αλγόριθμο διατηρήσαμε αυτή την μεθοδολογία, παρ’ όλα αυτά κάναμε τις απαραίτητες μετατροπές για να περιορίσουμε το κόστος. Πιο συγκεκριμένα ο αλγόριθμός μας έχει και αυτός ένα πρώτο μέρος επεξεργασίας και εύρεσης του χάρτη ακμών και ένα δεύτερο στον οποίο εφαρμόζουμε Circular Hough Transform αυτή την φορά παραποιημένο, προκειμένου να ελαχιστοποιηθεί η αρκετά μεγάλη πολυπλοκότητα.

Figure 3.12: Διάγραμμα ροής του προτεινόμενου αλγόριθμου



3.3.1 Εύρεση του Χάρτη Ακμών

Όσον αφορά την εύρεση του χάρτη ακμών ξεκινήσαμε με βάση το εξής: Παρατηρήσαμε ότι οι μεθοδολογίες που εξετάσαμε βασίζονται είτε σε κατωφλίωση είτε σε κάποια μέθοδο εύρεσης ακμών. Επόμενο βήμα είναι το φιλτράρισμα τον παραπάνω αποτελεσμάτων καθώς οι εικόνες που προκύπτουν έχουν μεγάλη πληροφορία. Το πρόβλημα με αυτό είναι ότι το φιλτράρισμα θέλει πολλές προσπελάσεις της εικόνας για να μειωθεί η πληροφορία και με βασική προϋπόθεση να έχει υλοποιηθεί το πρώτο βήμα, πρώτού ξεκινήσει το φιλτράρισμα. Στην δικιά μας περίπτωση

σκεφτήκαμε ότι ένας συνδυασμός δύο τεχνικών που ξεκινούν από διαφορετική αφετηρία και μπορούν να συνδυαστούν θα έλυνε αυτό το πρόβλημα.

Στην προτεινόμενη τεχνική, ο τελικός χάρτης ακμών για την ανίχνευση των κορών, εντοπίζεται συνδυάζοντας δύο χάρτες ακμών: (1) τον χάρτη ακμών που προκύπτει από την εφαρμογή της ανίχνευσης ακμής στην εικόνα του οφθαλμού όπως φαίνεται στο σχήμα 3.12 και 2) ο χάρτης ακμής που λαμβάνεται εφαρμόζοντας κατώτατο όριο στην εικόνα του ματιού για την κατάτμηση της περιοχής της κόρης, ακολουθούμενη από την ανίχνευση ακμής.

Δεδομένου ότι και οι δύο αυτοί χάρτες ακμών περιέχουν την κόρη από κοινού, συνδυάζονται σε ένα μόνο χάρτη χρησιμοποιώντας τη λειτουργία τομής, η οποία ελαχιστοποιεί τις ψευδείς ακμές λόγω θορύβου, όπως βλεφαρίδες, αντανακλάσεις φωτισμού κ.λπ. Η λειτουργία τομής σε αυτούς τους δύο χάρτες ακμών αφαιρεί την επίδραση, τόσο των ανακλάσεων όσο και του σκοτεινού φωτισμού. Οι επεξεργασίες που υφίσταται η εικόνα αναλύονται παρακάτω.

3.3.1.1 Gaussian Filtering

Η εξομάλυνση της αρχικής εικόνας του ματιού απομακρύνει το τυχαίο θόρυβο και τις άνισες εντάσεις που μπορεί να καταλήγουν σε περιττές ψευδείς ακμές στην ανιχνευμένη εικόνα. Η αρχική εικόνα του ματιού εξομαλύνεται χρησιμοποιώντας ένα Gaussian φίλτρο μεγέθους 3 x 3. Το μέγεθος και η τιμή σίγμα επιλέγεται, για να γίνει έπειτα η υλοποίηση του Gaussian φίλτρου στο Hardware, όσο το δυνατόν πιο αποδοτικά και απλά.

3.3.1.2 Sobel Edge Detection

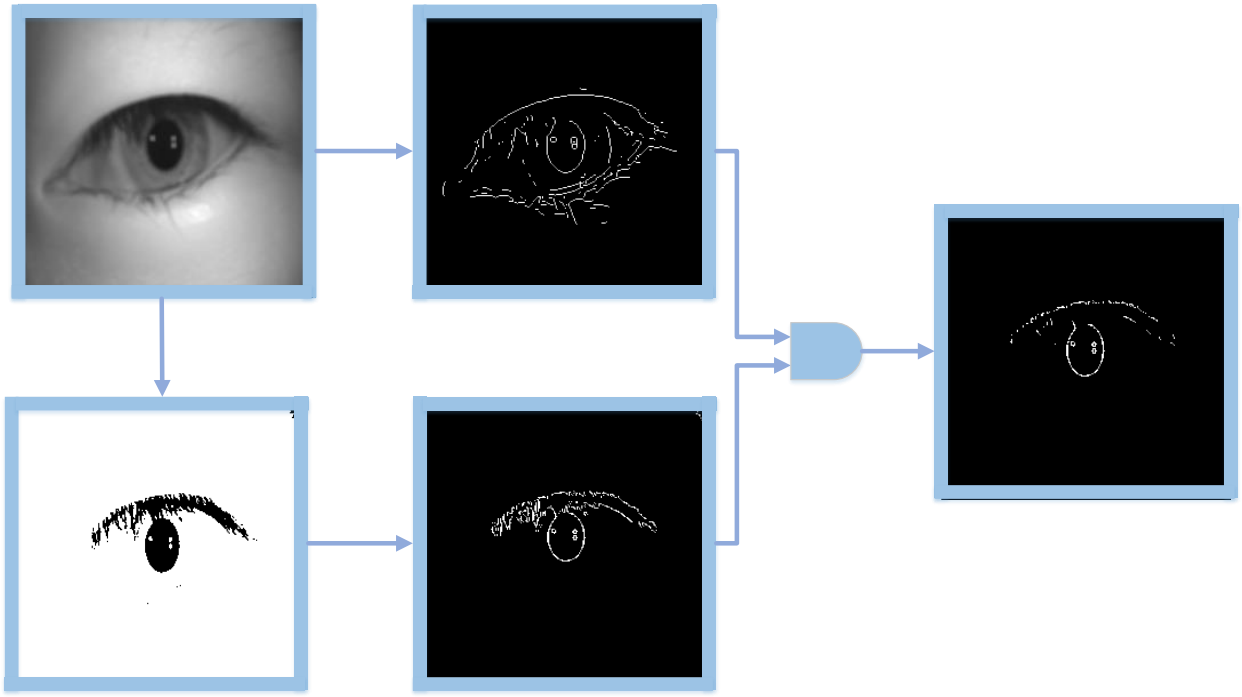
Ο ανιχνευτής ακμής Sobel εφαρμόζεται στην εξομαλυμένη εικόνα και επιλέχτηκε σε σχέση με άλλους ανιχνευτές λόγω της αποτελεσματικότητάς του σε συνδυασμό με την απλή πολυπλοκότητά (ιδιαίτερα σε σχέση με τον ανιχνευτή Canny). Δύο 3 x 3 μάσκες φίλτρων Sobel χρησιμοποιούνται για την εξεύρεση των Gradient x και Gradient y της εικόνας, προκειμένου να υπολογιστεί το συνολικό Gradient. Έπειτα επιλέγεται μια κατάλληλη τιμή κατωφλίου έτσι ώστε οι ακμές, οι οποίες είναι οι ισχυρές ακμές της εικόνας, να ανιχνεύονται ενώ οι αχνές ακμές όχι, δημιουργώντας τον πρώτο χάρτη ακμών.

3.3.1.3 Binarization

Η κόρη είναι η σκοτεινή περιοχή στην εικόνα που περιέχει την ίριδα. Έτσι, για να το εκμεταλλευτούμε αυτό, χρησιμοποιείται κατωφλίωση για τον κατακερματισμό της εικόνας και την εύρεση της ευρύτερης περιοχής της κόρης, μαζί με ανεπιθύμητη πληροφορία. Εφαρμόζεται κατώφλι (T) στην Gaussian εικόνα της ίριδας για να δημιουργηθεί η δυαδική εικόνα. Έπειτα στο αποτέλεσμα της κατωφλιωμένης εικόνας εφαρμόζεται εκ νέου φίλτρο Sobel, αποκτώντας ένα καινούριο χάρτη ακμών.

Τέλος οι δύο χάρτες ακμών συνδυάζονται και πιο συγκεκριμένα εντοπίζονται οι ακμές που εμφανίζονται και στους δύο χάρτες και οι οποίες αποτελούν εν τέλει τον τελικό χάρτη ακμής.

Figure 3.13:Ευρεση του χάρτη ακμών



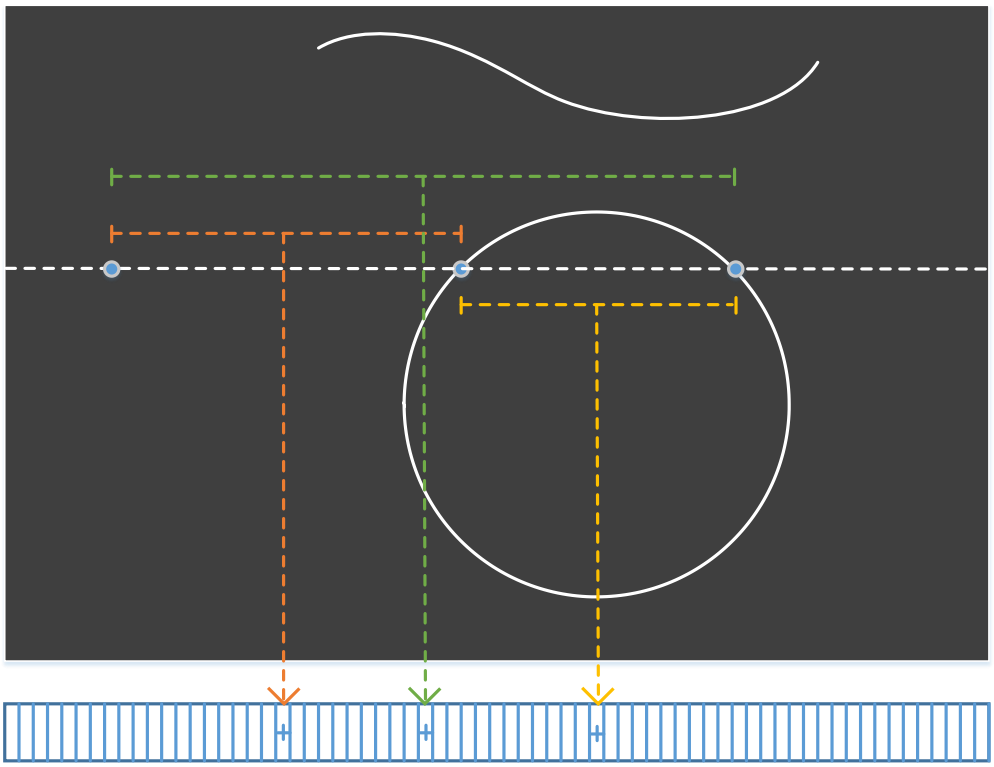
3.3.2 Circular Hough Transform

Επόμενο και βασικό βήμα του αλγόριθμου είναι η υλοποίηση του κυκλικού μετασχηματισμού Hough κατά τον οποίο θα προκύψει η θέση του κύκλου που περιγράφει την κόρη. Η μεγάλη πολυπλοκότητα του CHT μας αναγκάζει να κάνουμε και σε αυτόν μετατροπές για να την περιορίσουμε.

Η προτεινόμενη μετατροπή του Hough αλλάζει τον τρόπο που γίνεται η ψηφοφορία για τον

εντοπισμό του κύκλου. Στον πρότυπο Hough η ψηφοφορία γίνεται συνολικά ενώ εδώ την σπάμε σε δύο κομμάτια α) την ψηφοφορία για τον εντοπισμό του κέντρου και β) την εύρεση της ακτίνας. Αυτό μας εξασφαλίζει το να αποφύγουμε την εφαρμογή της περίπλοκής εξίσωσης του κύκλου πάνω στις ακμές της εικόνας.

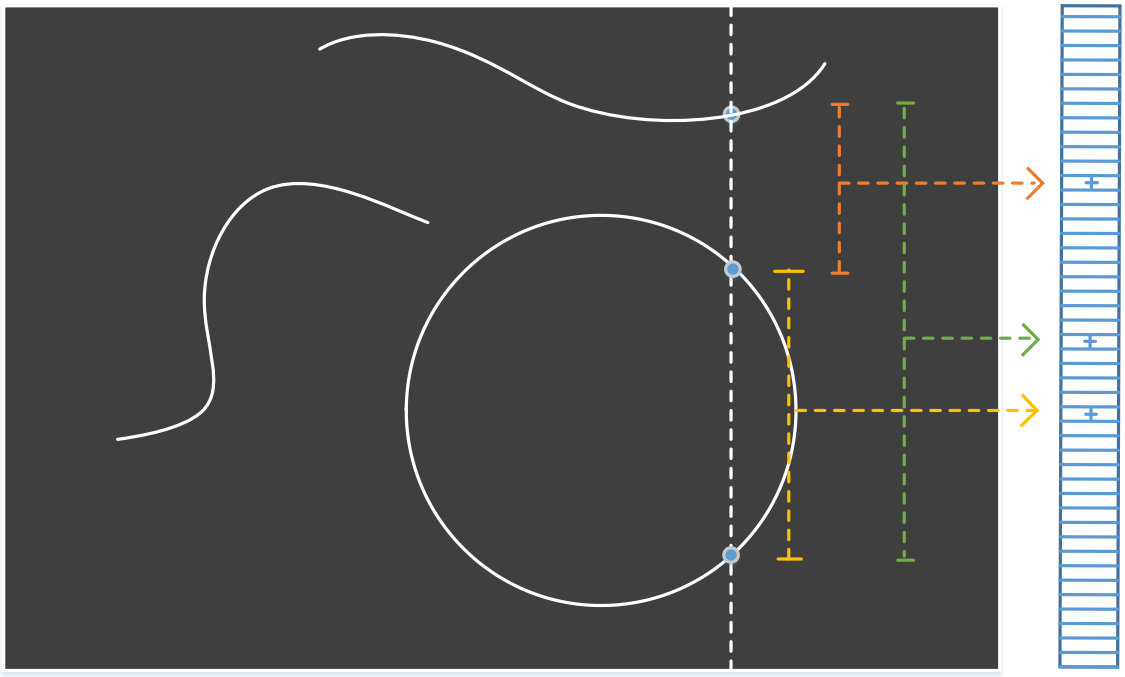
Figure 3.13:Ψηφοφορία στον άξονα χ για την εύρεση του κέντρου b.



Η ψηφοφορία για την εύρεση του κέντρου βασίζεται στην σάρωση της εικόνας μόνο δύο φορές, την μία για τον χ άξονα και την άλλη για τον y άξονα. Κατά την σάρωση αναζητούνται τα λευκά pixel που υποδηλώνουν ότι εντάσσονται σε μία ακμή και καταγράφονται όλες οι αποστάσεις μεταξύ αυτών τον pixel. Στο μέσο κάθε απόστασής εκχωρείται μία ψήφος. Η διαδικασία για τον χ άξονα στο τέλος θα αποδώσει τους περισσότερους ψήφους στο σημείο του κέντρου y. Στην

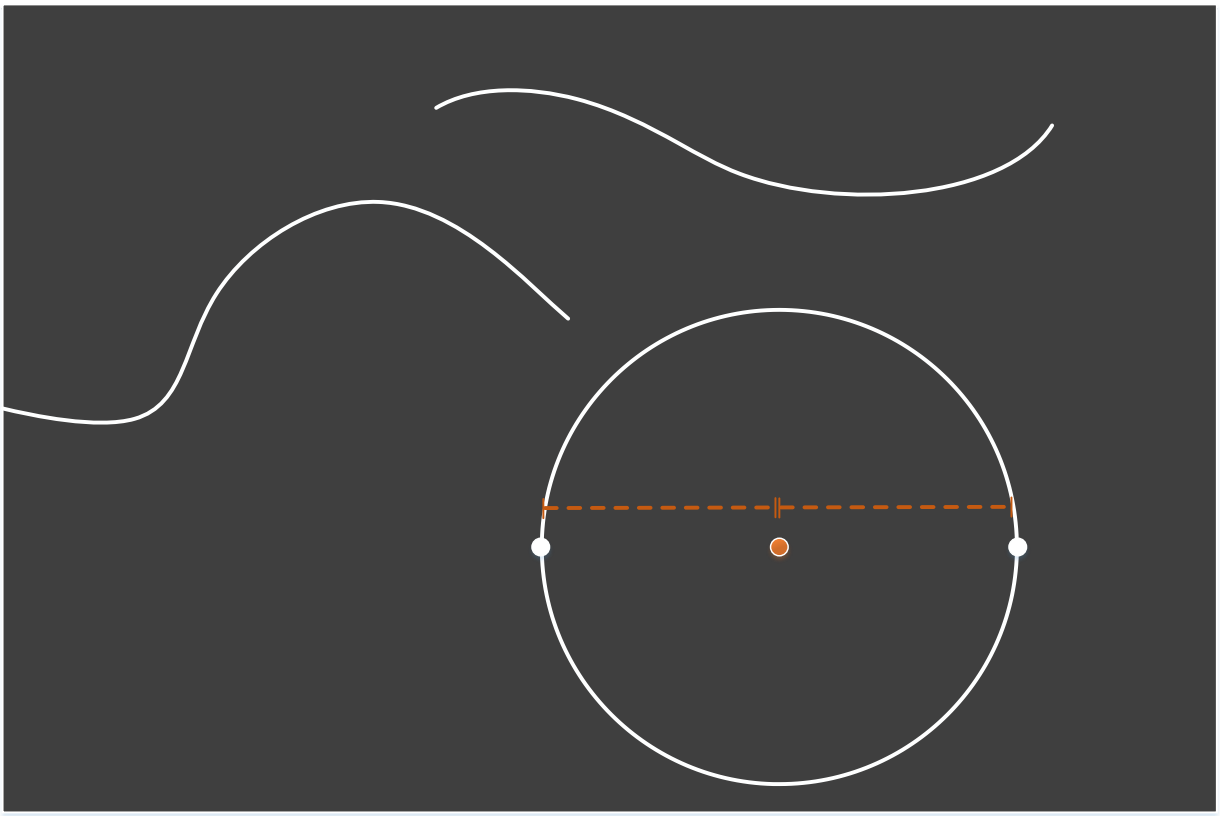
συνέχεια επαναλαμβάνεται για τον y άξονα εντοπίζοντας το σημείο χ. Με αυτόν τον τρόπο εντοπίζεται με πολύ λιγότερη πολυπλοκότητα το κέντρο του κύκλου όπως φαίνεται και στο σχήμα.

Figure 3.14:Ψηφοφορία στον άξονα y για την εύρεση του κέντρου a.



Παράλληλα ο εύκολος εντοπισμός του κέντρου, διευκολύνει και την αναζήτηση της ακτίνας. Η οποία, εφόσον το κέντρο είναι γνωστό μπορεί να περιοριστεί στην αναζήτηση συμμετρικών εμφανίσεων ακμών αριστερά και δεξιά του κέντρου. Η καλή επεξεργασία στον χάρτη ακμών μας δίνει ένα καλό αποτέλεσμα και για αυτό τον λόγο δεν χρειάζεται να αναζητηθούν πολλές ακτίνες. Ενδεικτικά εντοπίζονται 3 για τον άξονα χ και 3 για τον άξονα y και η τιμή που εμφανίζεται πιο συχνά είναι το τελικό αποτέλεσμα.

Figure 3.15: Υπολογισμός της ακτίνας



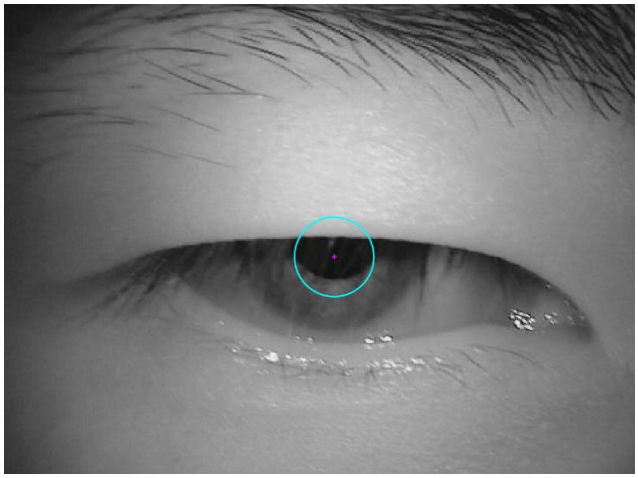
Εν κατακλείδι η τελική μας μοντελοποίηση μας εξασφαλίζει εξίσου αποτελεσματικότητα με τις παραπάνω μεθόδους που εξετάστηκαν στην Matlab με πολύ καλύτερα αποτελέσματα στο ζήτημα του χρόνου πράγμα και λογικό, εφόσον έγινε εκμετάλλευση των διάφορων παραγόντων που ρυθμίζονται από το σύστημά μας καθώς και της φύσης του Dataset, προκειμένου να ελαχιστοποιηθεί η πολυπλοκότητα.

Figure 3.16: Διάγραμμα μέσου χρόνου, των 3ων αλγορίθμων που υλοποιήσαμε στην Matlab

	SET	PUPIL LABS	ΠΡΟΤΕΙΝΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ
ΧΡΟΝΟΣ	345 ms	731 ms	180 ms

Ο προτεινόμενος αλγόριθμος που υλοποιήσαμε στηρίχθηκε σε στοιχεία των αλγόριθμων Set και Pupil Labs που όμως απλοποιήσαμε για να έχουν λιγότερη πολυπλοκότητα. Αντί για Canny Edge Detector χρησιμοποιούμε απλό φίλτρο Sobel που είναι υποσύνολο του Canny. Επιπλέον η υλοποίηση του CHT που εφαρμόσαμε δεν συγκρίνει κάθε σημείο ακμής με όλα τα υπόλοιπα προκειμένου να διευκρινιστεί αν ικανοποιούν την εξίσωση του κύκλου, όπως συμβαίνει στους αλγόριθμους Set και Pupil Labs. Επιπλέον εκτελέστηκε για 150 εικόνες και είχε επιτυχημένη εύρεση στις 140. Οι εικόνες που δεν ήταν επιτυχημένη η εύρεση περιέχουν εικόνες με κομμένη πάνω από 50% την κόρη όπως το παρακάτω παράδειγμα.

Figure 3.17: Αποτυχημένη εύρεση από τον προτεινόμενο αλγόριθμο, που υλοποιήσαμε στην Matlab.



4 ΥΛΟΠΟΙΗΣΗ ΣΕ HARDWARE

Όπως έχει ήδη αναφερθεί, στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη πάνω στις ποικίλες τεχνικές εύρεσης της τοποθεσίας της κόρης του ματιού και αξιοποιώντας την να παρουσιαστεί μια λειτουργική σχεδίαση στο Hardware, που θα υλοποιεί την παραπάνω λειτουργικότητα. Οι αλγόριθμοι που υλοποιήθηκαν στην Matlab και παρουσιάστηκαν αναλυτικά στα προηγούμενα κεφάλαια, διέπονται από την λογική της αξιοποίησής τους και μεταφοράς τους στο Hardware. Υπό αυτό το πρίσμα εξετάστηκαν και αξιολογήθηκαν πάνω σε ζητήματα όπως είναι οι πράξεις, ο χρόνος και φυσικά το ποσοστό επιτυχίας. Συλλέγοντας όλη αυτή την πληροφορία και συγκρίνοντας τις επιδόσεις κάθε επιμέρους τεχνικής, καταλήγουμε στο συμπέρασμα ότι ο δικός μας αλγόριθμος έχει λιγότερο κόστος για το σύστημα που θέλουμε να υλοποιήσουμε χωρίς όμως να υπολείπεται σε ποσοστό επιτυχημένης εύρεσης της κόρης.

Επιλέξαμε τον συγκεκριμένο αλγόριθμο να τον μεταφέρουμε στο Hardware. Για να το επιτύχουμε αυτό χρησιμοποιήσαμε τα εργαλεία που μας δίνει η Matlab και η Xilinx και πιο συγκεκριμένα το Simulink και το Xilinx System Generator, για να φτιάξουμε μία σχεδίαση την οποία μπορούμε να την εξετάσουμε και σε Simulation αλλά και να την τρέξουμε στην FPGA.

4.1 Xilinx System Generator

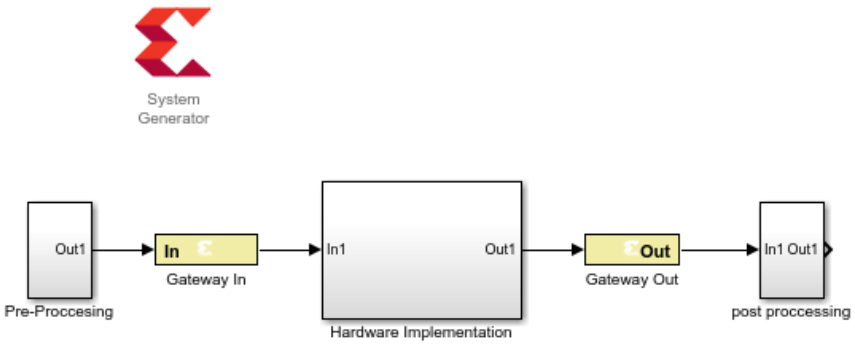
Το Xilinx System Generator είναι ένα εργαλείο σχεδίασης συστημάτων μέσω του περιβάλλοντος Simulink της MATLAB με στόχο την υλοποίηση της σχεδίασης σε FPGA. Η σχεδίαση ενός τέτοιου συστήματος έχει τη λογική του Block Diagram (όπως ακριβώς το simulink), δηλαδή την κατάλληλη διασύνδεση μονάδων-components που εκτελούν κάποια συγκεκριμένη λειτουργία. Κάθε υποσύστημα φυσικά έχει τις δικές του εισόδους και εξόδους. Αυτά τα σχέδια (designs) υλοποιούνται με μπλοκ που η ίδια η Xilinx έχει δημιουργήσει.

Το εργαλείο παρέχει τη δυνατότητα για πλήρη εξομοίωση του συστήματος που έχουμε σχεδιάσει, έλεγχο, αποτύπωση των αποτελεσμάτων είτε στο workspace της MATLAB είτε με οπτική αποτύπωση στην οθόνη. Φυσικά, για κάθε σχεδίαση μπορεί να παραχθεί και ο αντίστοιχος κώδικας VHDL που την περιγράφει πλήρως.

Τα blocks που μπορούν να χρησιμοποιηθούν παρέχονται από τη Xilinx στις βιβλιοθήκες του Simulink, αφού συνδεθεί η έκδοση του Vivado με αυτή της Matlab. Σε αυτά περιλαμβάνονται τα βασικότερα στοιχεία που μπορεί να αποτελούν μια εφαρμογή επεξεργασίας σήματος, αθροιστές, πολλαπλασιαστές καταχωρητές και άλλα. Επίσης περιλαμβάνονται και ένα σύνολο από πιο πολύπλοκα συστήματα όπως φίλτρα και μνήμες. Πέρα όμως από τα ήδη υπάρχοντα που παρέχονται από τις βιβλιοθήκες του System Generator υπάρχει η δυνατότητα ο σχεδιαστής να δημιουργήσει το δικό του System Generator block που μπορεί να εκτελεί ακόμα και ιδιαίτερα περίπλοκους αλγορίθμους. Ένα τέτοιο block μπορεί να αποτελεί ένα υποσύστημα της σχεδίασης και μπορεί να συνεργάζεται με τα ήδη διαθέσιμα μπλοκ. Παρ όλα αυτά στην παρούσα σχεδίαση δεν έχει χρησιμοποιηθεί κάτι τέτοιο αλλά την σχεδίασή μας την έχουμε οργανώσει σε Subsystems.

Μία σχεδίαση ενός συστήματος στο System Generator πρέπει πάντα να ξεκινά με τον προσδιορισμό των "ορίων" της FPGA. Τα δύο μπλοκ Gateway in και Gateway out μας δίνουν αυτή τη δυνατότητα. Στην ουσία, ό,τι υπάρχει ανάμεσα σε αυτά τα δύο μπλοκ αποτελεί το "περιεχόμενο" της FPGA, και ότι είναι έξω από αυτά χρησιμοποιείται είτε για να δώσει μία είσοδο στο σύστημά από την Matlab ή το Simulink, όπως πχ. μπορεί να είναι μία μεταβλητή από το workspace της Matlab, ή και να επιστρέψει κάποια έξοδο αντίστοιχα. Όλα τα υπόλοιπα μπλοκ της Xilinx, πρέπει οπωσδήποτε να παίρνουν δεδομένα εισόδου/εξόδου από και προς άλλα block της Xilinx.

Figure 4.1: Ο τρόπος σχεδίασης στο Xilinx System Generator. Τα blocks ανάμεσα στο Gateway in και στο Gateway out είναι αυτά που τρέχουν στην FPGA.



Μία σχεδίαση ενός τέτοιου συστήματος επίσης δεν μπορεί να δουλέψει αν δε θέσουμε κάποιες παραμέτρους. Αυτές λοιπόν τις παραμέτρους της σχεδίασης τις ρυθμίζει το System Generator Token. Το συγκεκριμένο block δε συνδέεται με το υπόλοιπο σύστημα, αλλά περιέχει πληροφορίες για το πώς αυτό θα τρέξει. Μέσω αυτού του μπλοκ ρυθμίζεται μεταξύ άλλων το clocking της

εξομοίωσης, ή, σε περίπτωση που θελήσουμε να κατεβάσουμε αυτή τη σχεδίαση σε κάποια FPGA, ρυθμίζει τα Mhz στα οποία θα τρέχει, κάνει generate τον VHDL κώδικα σε συγκεκριμένο directory και για συγκεκριμένο μοντέλο FPGA. Όταν ολοκληρώνουμε τη σχεδίασή μας μέσω της επιλογής Generate του System Generator Token δημιουργούνται τα απαραίτητα αρχεία για την υλοποίηση στην FPGA μέσω του προγράμματος Vivado.

Στην δικιά μας περίπτωση θέλουμε να χρησιμοποιήσουμε το Xilinx System Generator για την επεξεργασία εικόνας. Για αυτό τον σκοπό θα γίνεται μία προ επεξεργασία της εικόνας προκειμένου να μετατραπεί η εικόνα σε σειρά και μία επεξεργασία μετά την έξοδο της FPGA προκειμένου να υπάρχει απεικόνιση των αποτελεσμάτων.

Τα πλεονεκτήματα του Xilinx System Generator είναι ποικίλα. Προφανώς και για έναν σχεδιαστή Hardware ενδείκνυται να χρησιμοποιήσει κάποιο άλλο εργαλείο καθώς θα του δίνεται η δυνατότητα να πετύχει καλύτερα αποτελέσματα, δίχως τους επιμέρους περιορισμούς του Xilinx System Generator. Παρ όλα αυτά για μηχανικούς που θέλουν να διερευνήσουν κάποιους αλγόριθμους και να κάνουν έρευνα, στο πως θα μεταφερθούν αποδοτικά στο Hardware είναι πολύ αποτελεσματικό καθώς δεν υστερεί και δίνει την δυνατότητα να εμφανίζονται αποτελέσματα στο Simulation εξασφαλίζοντας χρόνο στον σχεδιαστή. Επιπλέον για περιπτώσεις όπως είναι η επεξεργασία εικόνας, όπως και στην δικιά μας, είναι ιδιαίτερα χρήσιμο γιατί δίνεται η δυνατότητα στον σχεδιαστή να έχει οπτικά αποτελέσματα και να μπορεί να επεξεργαστεί την σχεδίασή του πολύ πιο εύκολα και με περισσότερη επιτυχία. Τέλος υπάρχει η δυνατότητα εύκολα η σχεδίαση να κατέβει στην Fpga δίχως να υστερεί σε αποτελεσματικότητα σε μεγάλο βαθμό.

4.2 Η Σχεδίαση του Προτεινόμενου Αλγόριθμου

Όλες οι τεχνικές τις οποίες εξετάσαμε αποτελούνται από δύο ξεχωριστά κομμάτια. Την επεξεργασία της αρχικής εικόνας προκειμένου να καταλήξουμε στον τελικό χάρτη ακμών ο όποιος θα περιλαμβάνει τον κύκλο, που περιγράφει την κόρη του ματιού, εντός του και μετέπειτα η επεξεργασία της δεύτερης εικόνας με κάποιο κριτήριο κυρτότητας ή έναν αλγόριθμο εύρεσης κύκλου, προκειμένου να εντοπίσουμε και τον κύκλο αλλά και πληροφορία όπως είναι οι συντεταγμένες του κέντρου του και η ακτίνα του. Η δικιά μας τεχνική ακολουθεί ακριβώς αυτό το μοτίβο, όπως έχει περιγράφει σε προηγούμενο κεφάλαιο. Ο στόχος της σχεδίασης είναι να γίνεται η βασική επεξεργασία στην FPGA και το σύστημά μας, παίρνοντας σαν είσοδο μία εικόνα, να βρίσκει την τοποθεσία της κόρης του ματιού και πιο συγκεκριμένα το κέντρο του κύκλου που την περιγράφει.

Παρ όλα αυτά για την σχεδίαση στο Hardware διαφοροποιούνται κάποιοι παράγοντες που μέχρι τώρα ήταν δεδομένοι και για τους οποίους ο αλγόριθμος πρέπει να προσαρμοστεί. Πρώτη βασική διαφορά είναι ότι η επεξεργασία πρέπει να γίνει πλέον Pixel by Pixel. Η είσοδος στο σύστημα δεν μπορεί να είναι ένας πίνακας αλλά αντίθετα θα είναι η ακολουθία των Pixel, από τα οποία αποτελείται η εικόνα μας και η επεξεργασία τους θα γίνεται σειριακά. Σε αυτό το σημείο αξίζει να σημειωθεί ότι στην σχεδίαση μας η οποία έχει κατεύθυνση την FPGA, θεωρεί ότι έχει συγκεκριμένο μέγεθος η εικόνα, άρα και κατ' επέκταση και η ακολουθία των Pixel. Επομένως πριν μπει η εικόνα σαν είσοδος έχει υλοποιηθεί ένα μικρό κομμάτι προ- επεξεργασίας στο οποίο περιλαμβάνεται και η αλλαγή μεγέθους.

Πιο συγκεκριμένα η σχεδίαση μας θα λειτουργεί λαμβάνοντας υπόψιν ότι έχει σαν είσοδο μία εικόνα 256*256 στοιχείων. Αυτό δεν αποτελεί πρόβλημα καθώς τα αποτελέσματά της επεξεργασίας στην FPGA, δηλαδή η τοποθεσία του κέντρου της κόρης μαζί με την ακτίνα της, θα υποστούν μία μικρή επεξεργασία μετά την Fpga, προκειμένου αναλογικά με την αλλαγή στο μέγεθος που έγινε στην προ-επεξεργασία της αρχικής εικόνας, ώστε να βρεθεί και να υπολογιστεί εύκολα το τελικό αποτέλεσμα.

Μια ακόμα βασική διαφορά με την οποία ερχόμαστε αντιμέτωποι στο Hardware, είναι ότι πρέπει να αντιμετωπιστεί και το πρόβλημα του συγχρονισμού. Το σύστημά μας θα διαβάζει και θα επεξεργάζεται τα Pixel σειριακά σε κάθε κύκλο του ρολογιού. Στο Hardware το πότε γίνεται το κάθε τι έχει τεράστια σημασία, επομένως δεν μπορούμε να το αγνοήσουμε αλλά αντίθετα κιόλας πρέπει να φροντίσουμε όσο το δυνατόν να γίνονται κομμάτια της επεξεργασίας παράλληλα προκειμένου να μειωθούν οι συνολικοί κύκλοι που χρειάζεται το σύστημα μας. Υπό αυτή την έννοια η σχεδίαση μας χωρίζεται σε 6 στάδια.

Είναι αναγκαίο πριν προχωρήσουμε σε βάθος, να γίνει μία περιληπτική ανάλυση για το πως λειτουργεί η σχεδίασή μας. Αρχικά δίνεται σαν είσοδος η εικόνα σαν μία ακολουθία από Pixel. Τα δεδομένα λοιπόν που δίνονται στην αρχική είσοδο, αποθηκεύονται σε μία BRAM (εσωτερική μνήμη στην FPGA). Γνωρίζοντας εκ των προτέρων το μέγεθος που έχει μετασχηματιστεί η εικόνα, ξέρουμε πόσο χώρο καταλαμβάνει άρα και το πόσο μέγεθος θα έχει η BRAM και σε κάθε διεύθυνση αποθηκεύεται η τιμή του κάθε Pixel. Στην συνέχεια πρέπει να γίνει επεξεργασία εικόνας προκειμένου να παραχθούν οι φιλτραρισμένες ακμές, που υποσύνολό τους είναι και η κόρη του ματιού. Ο χάρτης ακμών που θα δημιουργηθεί θα χρησιμοποιηθεί για να γίνει η εφαρμογή του Circular Hough Transform. Πρώτο βήμα αυτής της διαδικασίας είναι να γίνει ψηφοφορία για να βρεθεί το κέντρο. Για κάθε ζευγάρι λευκών Pixel βρίσκεται το σημείο που ισαπέχει από τα δύο Pixel και καταχωρείται μία ψήφος για την θέση του. Αυτό γίνεται τόσο στον

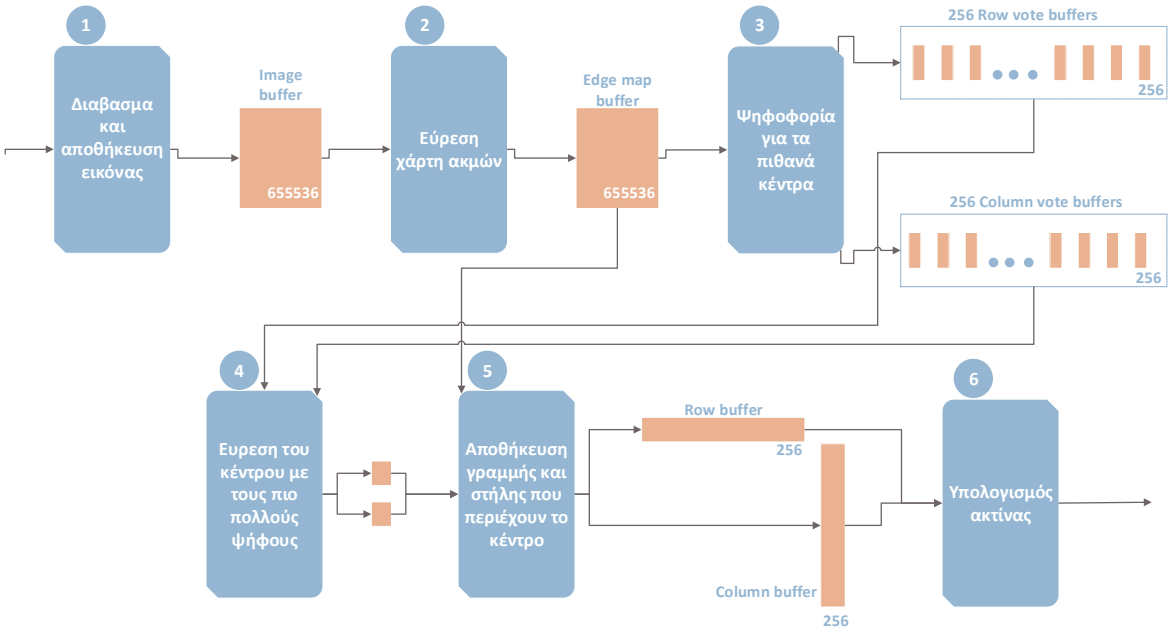
οριζόντιο όσο και στον κατακόρυφο άξονα. Αυτές οι ψήφοι θα καταμετρηθούν και στην συνέχεια θα προκύψει μία θέση στον κατακόρυφο και μία θέση στον οριζόντιο άξονα που έχει πάρει τους περισσότερους ψήφους. Το ζευγάρι των δύο αυτών θέσεων θα αποθηκευτεί σαν το κέντρο του κύκλου, ο οποίος αναζητάται με τον Circular Hough Transform. Έπειτα, γνωρίζοντας σε ποιο σημείο είναι το κέντρο θα απευθυνθούμε ξανά στο χάρτη ακμών προκειμένου να απομονωθούν η γραμμή και η στήλη που έχουν αναφορά στο κέντρο. Τέλος για την γραμμή και την στήλη θα γίνει ο υπολογισμός για να βρεθεί η υποψήφια ακτίνα στον οριζόντιο και στον κάθετο άξονα. Οι υποψήφιες ακτίνες και το κέντρο του κύκλου είναι η έξοδος από την FPGA.

Όπως αναφέρθηκε και προηγουμένως, αλλά και προκύπτει και από την περιγραφή που δόθηκε, η σχεδίαση μας αποτελείται από 6 βασικά στάδια. Αυτό προκύπτει από την ανάγκη, όσο προχωράει ο αλγόριθμος, να γίνεται κάποιο κομμάτι της επεξεργασίας, να αποθηκεύονται τα δεδομένα που προκύπτουν και να αρχίζει το επόμενο στάδιο επεξεργασίας παίρνοντας σαν είσοδο τα δεδομένα του προηγούμενου σταδίου. Για αυτό μετά από κάθε στάδιο, όπως φαίνεται και στο σχήμα, γίνεται αποθήκευση των δεδομένων σε Bram προκειμένου να μπορούν να αξιοποιηθούν μετέπειτα. Άρα είναι δεδομένο ότι για να αρχίσει κάποιο στάδιο να υλοποιείται πρέπει να έχει τελειώσει το προηγούμενο και οι συνολικοί κύκλοι που θα πάρει το σύστημά μας για να τερματίσει είναι το άθροισμα των κύκλων κάθε σταδίου. Παρόλα αυτά εντός των σταδίων όσο είναι εφικτό επιτυγχάνεται παραλληλία. Εντούτοις οι κύκλοι ρολογιού έτσι ώστε να ολοκληρωθεί κάποιο στάδιο είναι ανάλογοι με το μέγεθος της εικόνας. Αυτό δεν μπορεί να αποφευχθεί καθώς η επεξεργασία γίνεται σειριακά. Για παράδειγμα για να διαβαστεί μία εικόνα και να αποθηκευτεί χρειάζονται τόσοι κύκλοι όσο και το πλήθος των Pixel της. Τα 6 στάδια ακολουθούν και αναλύονται σε βάθος στην συνέχεια.

Στάδια της Σχεδίασης.

- Διάβασμα και αποθήκευση της εικόνας.
- Εύρεση του χάρτη ακμών.
- Ψηφοφορία για τα πιθανά κέντρα.
- Ευρεση του κέντρου με τις περισσότερες ψήφους.
- Αποθήκευση της γραμμής και της στήλης που περιέχουν το κέντρο.
- Υπολογισμός ακτίνας.

Figure 4.2: Το Datapath της σχεδίασής μας με τα 6 βασικά στάδια.



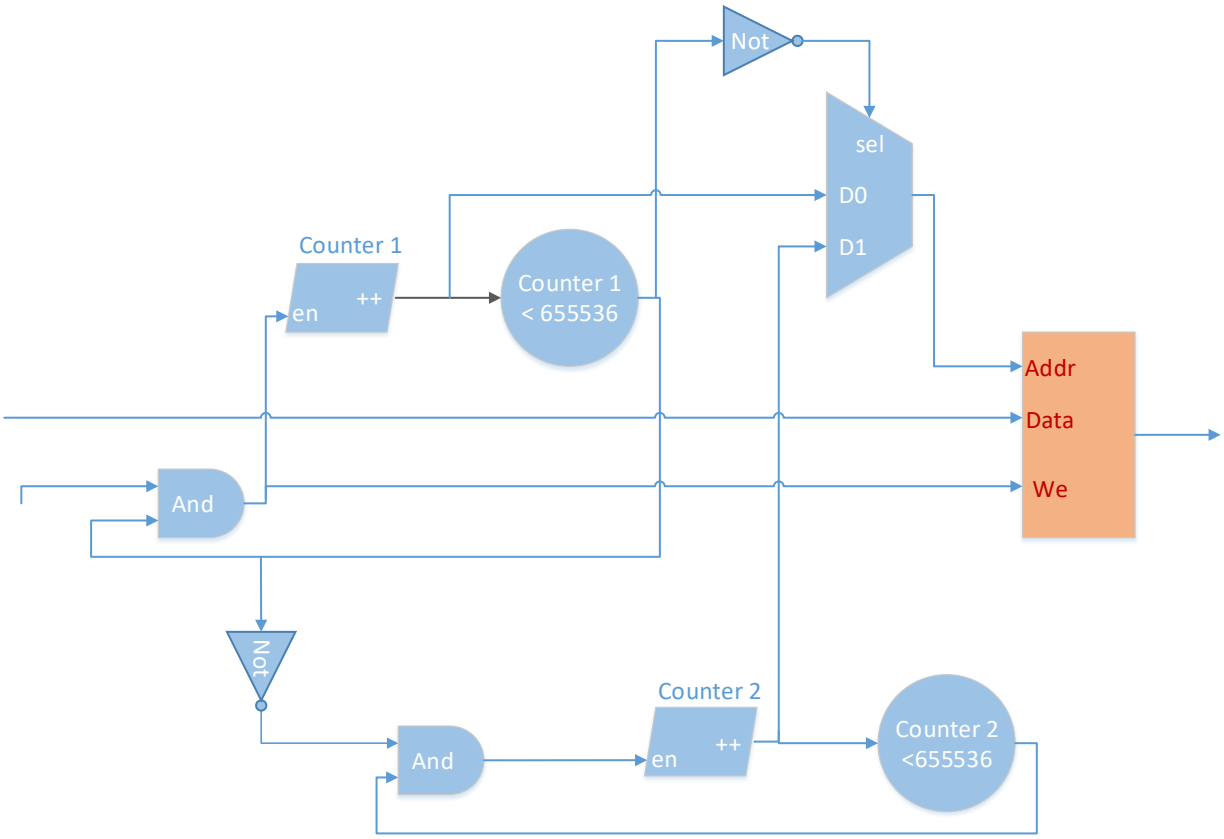
4.2.1 Διάβασμα και Αποθήκευση της Εικόνας

Όπως έχει αναφερθεί ήδη η σχεδίασή μας παίρνει σαν είσοδο την εικόνα μεγέθους 256*256 σειριακά και θα την επεξεργαστεί Pixel Pixel. Συνεπώς τα πρώτα δεδομένα που θα έχουμε σαν είσοδο θα είναι 65536 στοιχεία που το κάθε ένα θα έχει την τιμή ενός Pixel της εικόνας. Αυτά τα δεδομένα θα τα αποθηκεύσουμε σε μία Bram που θα είναι αντίστοιχου μεγέθους, δηλαδή 65536 θέσεων και σε κάθε θέση της θα αποθηκευτεί ένα Pixel.

Η παραπάνω διαδικασία γίνεται με τον εξής τρόπο. Στην σχεδίαση μας χρησιμοποιείται ένας μετρητής ο οποίος είναι αρχικοποιημένος στο 0 και χρειάζεται να φτάσει μέχρι το 65535. Ο μετρητής ενεργοποιείται μέσω του σήματος enable το οποίο παίρνει την τιμή 1 σαν είσοδο, όταν έχουμε δεδομένα που θέλουμε να αποθηκευτούν και η συνθήκη ότι ο μετρητής δεν έχει φτάσει στο όριο, δηλαδή το 65535 είναι αληθής. Για να το επιτευχθεί όλη η παραπάνω διαδικασία χρησιμοποιείται μία πύλη And η οποία παίρνει σαν είσοδο το σήμα valid input (δηλαδή ότι τα έχουμε δεδομένα) και το αποτέλεσμα της συνθήκης: counter1<65536 . Όταν και τα Δύο είναι αληθής το enable του μετρητή ενεργοποιείται και ο μετρητής αρχίζει ή συνεχίζει την μέτρηση.

Τον μετρητή τον χρειαζόμαστε για δύο λόγους. Ο πρώτος είναι για να γνωρίζουμε σε ποιο σημείο βρίσκεται η διαδικασία του διαβάσματος και της εγγραφής. Ο δεύτερος είναι για να αποτελεί η τιμή του μετρητή είσοδο στην BRAM και συγκεκριμένα την διεύθυνση στην οποία θα αποθηκευτούν τα δεδομένα που λαμβάνει. Έτσι σε κάθε κύκλο ρολογιού η BRAM παίρνει μία τιμή ενός Pixel και την τιμή του μετρητή σαν διεύθυνση και το αποθηκεύει στην αντίστοιχη θέση. Φυσικά χρειαζόμαστε και την ενεργοποίηση της εγγραφής προκειμένου να γράψουμε κάτι στην BRAM. Χρειαζόμαστε να έχουμε εγγραφή, όσο δουλεύει ο μετρητής, άρα όταν θέλουμε 1 στο enable του μετρητή θέλουμε και 1 στο write enable της Bram. Για αυτό το λόγο συνδέεται το write enable με το αποτέλεσμα της πύλης And που περιεγράφηκε προηγουμένως.

Figure 4.3: Ο τρόπος διαβάσματος και αποθήκευσης της εικόνας στην Bram.



Αφού τελειώσει η διαδικασία της εγγραφής θα υπάρχει αποθηκευμένη όλη η εικόνα στην BRAM. Αρά προκειμένου να προχωρήσουμε στο επόμενο στάδιο που είναι η επεξεργασία της εικόνας προκειμένου να παραχθεί ο χάρτης ακμών θα χρειαστεί να διαβάσουμε δεδομένα από την Bram. Σε αυτό το σημείο χρειάζεται να χρησιμοποιηθεί και ο απαραίτητος έλεγχος ότι έχει τελειώσει η διαδικασία της εγγραφής. Άρα μόλις τελειώσει ο πρώτος μετρητής ενεργοποιούμε το σήμα write image finished το οποίο στην ουσία ξεκινά το επόμενο στάδιο.

Κατά το επόμενο στάδιο θα χρησιμοποιήσουμε έναν δεύτερο μετρητή ο οποίος αντιπροσωπεύει την διεύθυνση από την οποία θα πάρουμε το περιεχόμενο. Επιπρόσθετα χρησιμοποιείται και ένας πολυπλέκτης, όπως φαίνεται και στο σχήμα 4.3, ο οποίος ρυθμίζει ανάλογα σε ποιο στάδιο είμαστε ποια διεύθυνση χρησιμοποιείται. Έτσι όταν βρισκόμαστε στο στάδιο παραγωγής του χάρτη ακμής θα χρησιμοποιείται σαν διεύθυνση ο δεύτερος μετρητής και σε κάθε κύκλο θα διαβάζεται ένα Pixel και θα δέχεται επεξεργασία η οποία θα περιγράφει στην συνέχεια.

4.2.2 Επεξεργασία και Εύρεση του χάρτη ακμών

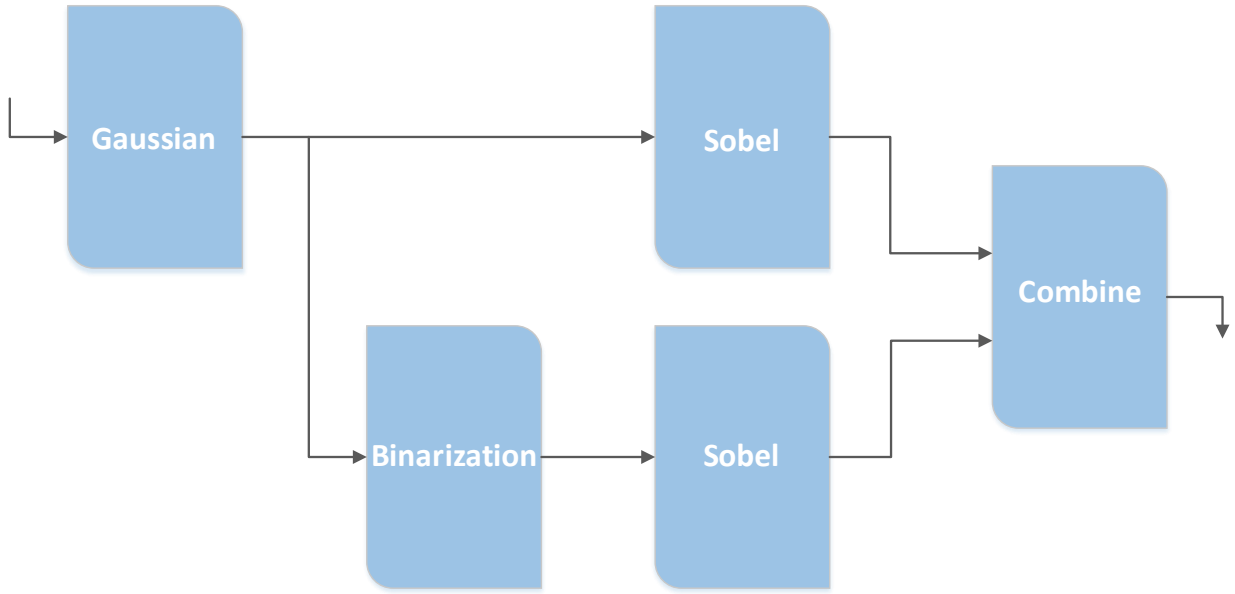
Αυτό το στάδιο της σχεδίασης μας αποτελεί έναν βασικό συντελεστή για κάθε αλγόριθμο εύρεσης της κόρης του ματιού. Αυτό συμβαίνει διότι σε αυτό το κομμάτι θα γίνει επεξεργασία έτσι ώστε να μειωθεί η πληροφορία που περιέχει η εικόνα, να δημιουργηθεί ένας χάρτης ακμών και στα καινούρια δεδομένα να γίνει η εύρεση της κυκλικής κόρης. Όσον αφορά το Hardware αποτελεί ένα σημαντικό στάδιο και για έναν επιπλέον λόγο γιατί μειώνοντας την πληροφορία οι πράξεις που θα χρειάζονται στα επόμενα στάδια θα είναι λιγότερες.

Όσον αφορά το αλγοριθμικό κομμάτι το οποίο έχει αναλυθεί εκτενώς σε προηγούμενο κεφάλαιο, σε αυτό το στάδιο η εικόνα θα φιλτραριστεί πρώτα με μία μάσκα Gaussian προκειμένου να ελαχιστοποιηθεί ο θόρυβος και έπειτα θα οδηγηθεί σε δύο επεξεργασίες.

Στην πρώτη γίνεται πρώτα κατωφλίωση της εικόνας και μετέπειτα με την εφαρμογή της μάσκας Sobel βρίσκονται οι ακμές του αποτελέσματος μετά την κατωφλίωση ενώ στην δεύτερη γίνεται απλή εφαρμογή του φίλτρου Sobel για την εύρεση ακμών. Στο τέλος θα συνδυαστούν τα αποτελέσματα των δύο αυτών επεξεργασιών προκειμένου να προκύψει ο τελικός χάρτης ακμών. Όλο αυτό είναι σημαντικό για την υλοποίηση σε Hardware καθώς με μικρές σε κόστος

επεξεργασίες μειώνουμε πολύ στον χάρτη ακμών τα κομμάτια που δεν αφορούν την κόρη. Επιπλέον είναι σημαντικό, όπως φαίνεται και στο σχήμα, ότι η σχεδίαση μας υλοποιεί τις δύο αυτές επεξεργασίες παράλληλα.

Figure 4.4: Η διαδικασία της εύρεσης του χάρτη ακμών.



Προκειμένου να φιλτραριστεί η εικόνα μέσω ενός Gaussian φίλτρου πρέπει να γίνει συνέλιξη της εικόνας με μία μάσκα Gaussian. Για αυτό το σκοπό επιλέχτηκε να δημιουργηθεί μία μάσκα 3x3, όπως φαίνεται και στο σχήμα. Αν ορίσουμε την εικόνα σαν πίνακα, η διαδικασία της συνέλιξης της εικόνας με την μάσκα, είναι ο πολλαπλασιασμός κάθε στοιχείου της μάσκας με το αντίστοιχο στοιχείο του αποτυπώματος της μάσκας πάνω στην εικόνα και το συνολικό άθροισμα όλων αυτών θα είναι η τιμή ενός pixel της συνέλιξης Αυτό επαναλαμβάνεται μεταφέροντας το αποτύπωμα της

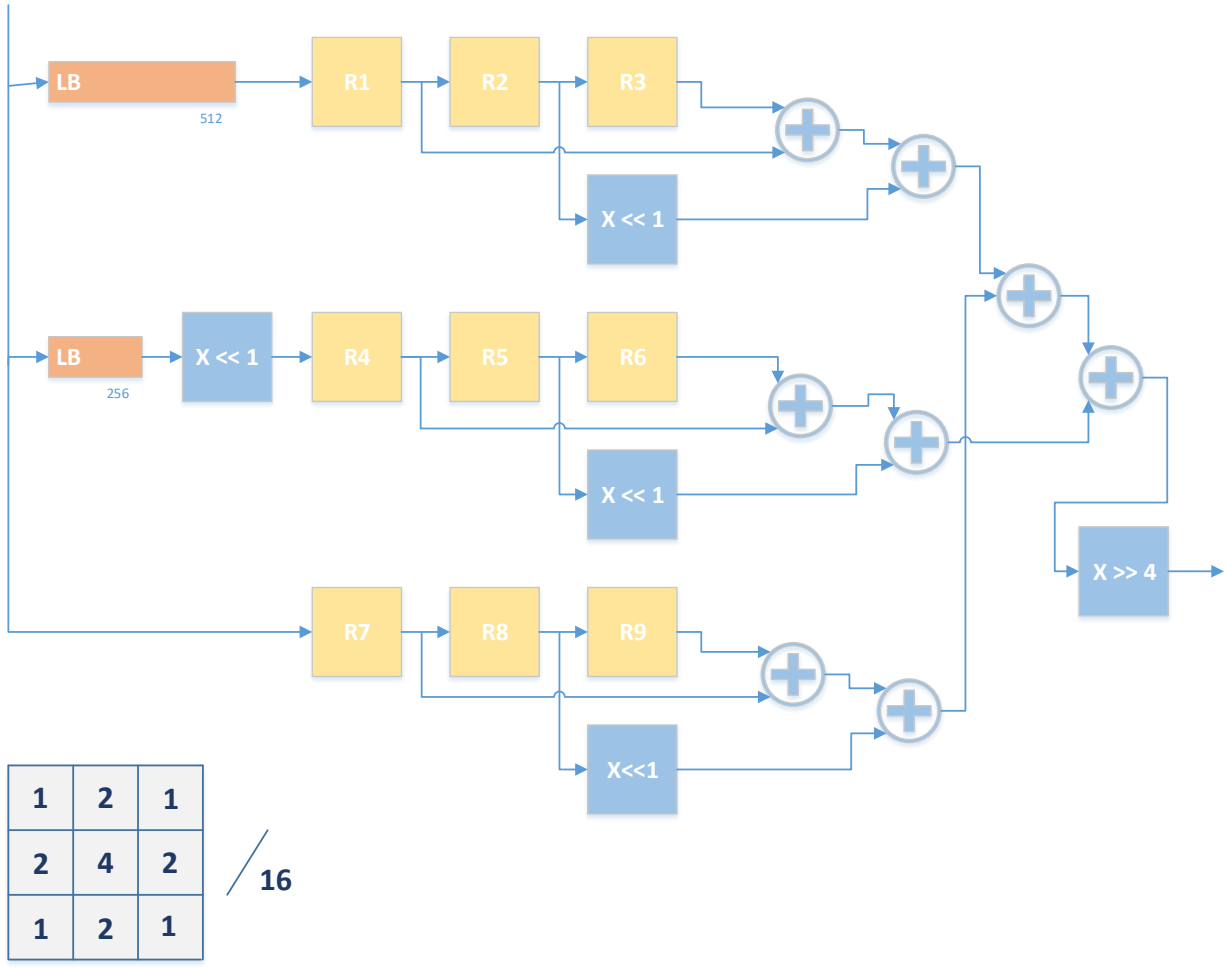
μάσκας σε όλη την εικόνα, αποκτώντας έναν καινούριο πίνακα ο οποίος αντιπροσωπεύει την φιλτραρισμένη εικόνα.

Στο Hardware όμως υπάρχει η διαφορά ότι τα δεδομένα υφίσταται επεξεργασία σειριακά. Για να προσομοιώσουμε τα αποτυπώματα της μάσκας σαν πίνακες ακολουθείται η εξής μεθοδολογία. Αρχικά χρησιμοποιούμε 9 Register οι οποίοι κρατάνε τις τιμές 9 στοιχείων της εικόνας για ένα αποτύπωμα της μάσκας. Οι 9 αυτοί Register χωρίζονται σε 3 ομάδες και οι Register κάθε ομάδας είναι συνδεδεμένοι σε σειρά αντιπροσωπεύοντας 3 διαδοχικά pixel. Η τελευταία ομάδα λαμβάνει απευθείας δεδομένα, η δεύτερη λαμβάνει δεδομένα μετά από καθυστέρηση 256 κύκλων, χρησιμοποιώντας έναν Line Buffer 256 θέσεων, και η πρώτη μετά από 512 κύκλους με την χρησιμοποίηση ενός Line Buffer 512 κύκλων.

Εφόσον η εικόνα μας έχει μέγεθος 256x256 μετά από 512 κύκλους θα έχουμε καταχωρημένα στην πρώτη ομάδα τα 3 πρώτα Pixel της πρώτης γραμμής, στην δεύτερη τα 3 πρώτα pixel της δεύτερης γραμμής και στην τρίτη τα 3 πρώτα pixel της 3 γραμμής. Άρα θα έχουμε το πρώτο αποτύπωμα της μάσκας από το οποίο κάνοντας τις απαραίτητες πράξεις θα βγει το πρώτο pixel της συνελημμένης εικόνας. Από αυτό το σημείο και έπειτα σε κάθε κύκλο θα έχουμε σαν αποτέλεσμα και ένα pixel, με την παραδοχή ότι στο τέλος αυτού του σταδίου πρέπει να λάβουμε υπόψιν ότι για να πάρουμε το πρώτο σωστό αποτέλεσμα υπήρξε καθυστέρηση 512 κύκλων.

Αφού λοιπόν έχει επιτευχθεί η καταχώρηση των τιμών του αποτυπώματος στους αντίστοιχους Register το επόμενο μέλημα είναι να γίνουν οι πολλαπλασιασμοί με την μάσκα. Άρα έπειτα από κάθε Register ακολουθεί και ο αντίστοιχος πολλαπλασιασμός με ολίσθηση αριστερά όσες θέσεις χρειάζεται. Στο τέλος αθροίζουμε τα αποτελέσματα με την μορφή δέντρου και κάνουμε την απαραίτητη διαίρεση, με ολίσθηση δεξιά 4 θέσεων, εφόσον όλες οι τιμές τις μάσκας ήταν τις μορφής χ/16, όπου το χ είναι ακαριαίος.

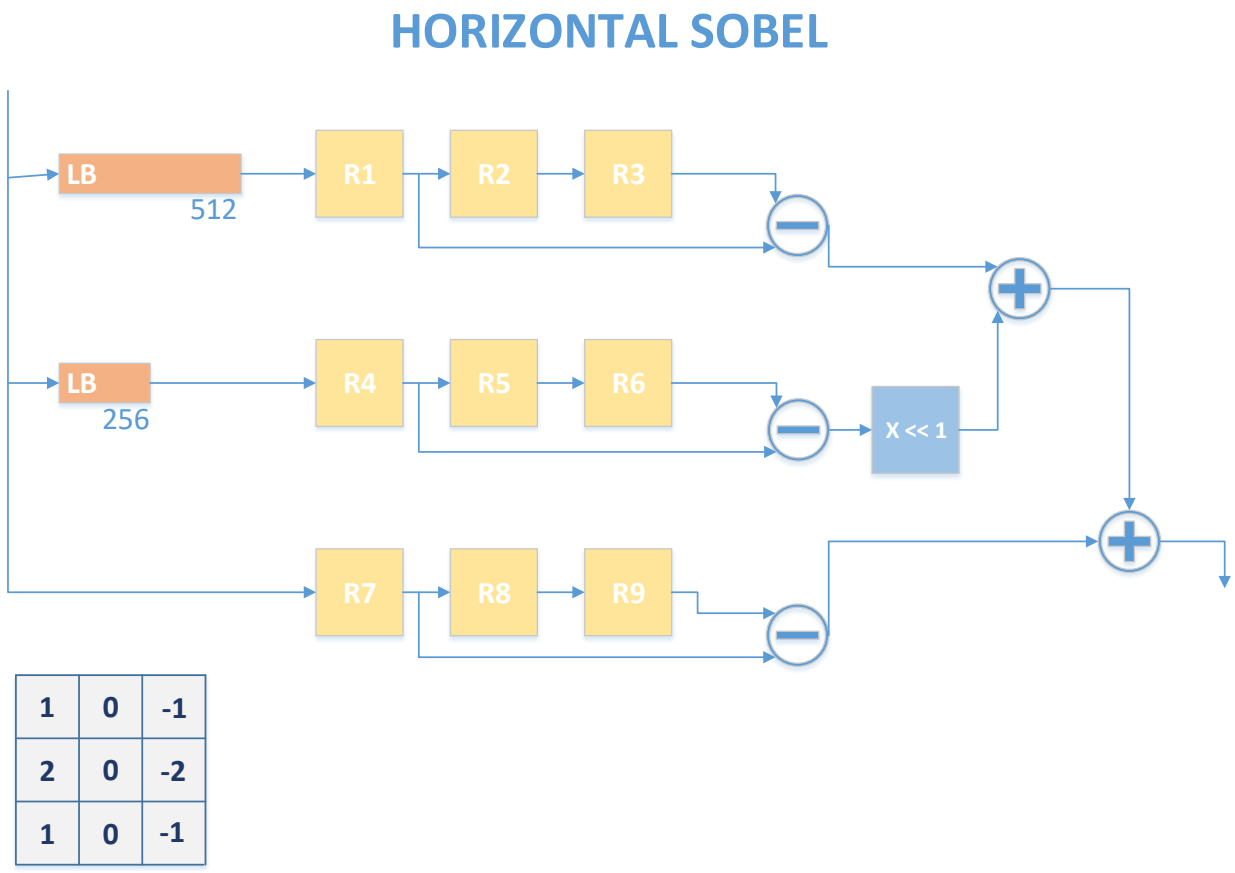
Figure 4.5: Gaussian filter 3x3.



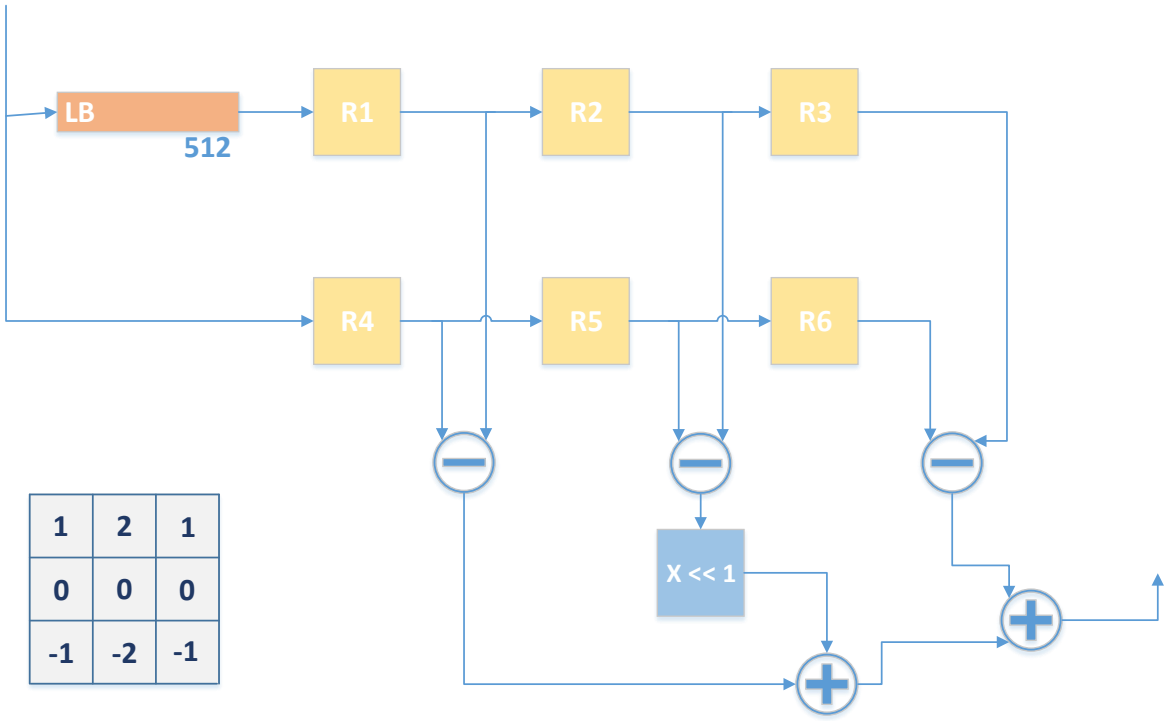
Έπειτα από το φιλτράρισμα της εικόνας και την αφαίρεση του θορύβου από αυτή, μέσω της συνέλιξης με την μάσκα Gaussian, τα επόμενα δύο βήματα θα γίνουν παράλληλα όπως φαίνεται και στο Σχήμα 1.4. Στο πρώτο κομμάτι θα υλοποιηθεί Edge detection με την εφαρμογή ενός φίλτρου Sobel. Για την εφαρμογή του Sobel έχουν δημιουργηθεί δύο μάσκες 3x3 από τις οποίες θα υπολογιστούν το Gradient x και Gradient y. Η συνέλιξη με τις μάσκες αυτές γίνεται με τον τρόπου που περιεγράφηκε για το Gaussian με την μόνη διαφοροποίηση να είναι οι τιμές των μασκών οι οποίες φαίνονται και στα ακόλουθα σχήματα. Οι μάσκες που χρησιμοποιούνται είναι

δυνάμεις του 2 προκειμένου για τους πολλαπλασιασμούς και τις διαιρέσεις να μπορεί να χρησιμοποιηθεί ολίσθηση.

Figure 4.6: Horizontal και Vertical Sobel.



VERTICAL SOBEL



Το Gx θα είναι το αποτέλεσμα ύστερα από την συνέλιξη με την μάσκα Horizontal Sobel και το Gy ύστερα από την συνέλιξη με την μάσκα Vertical Sobel. Από το θεωρητικό κομμάτι που εξετάστηκε στα προηγούμενα κεφάλαια είναι γνωστό πως για να βρεθεί το τελικό Gradient πρέπει να γίνει ο υπολογισμός $\sqrt{\mathrm{Gx}^2 + Gy^2}$. Επομένως με δύο πολλαπλασιαστές βρήκαμε τα Gx^2 και $\mathrm{G}y^2$ και μετά χρησιμοποιήθηκε το block square root του Xilinx System Generator για τον υπολογισμό της ρίζας. Στο τέλος με την βοήθεια ενός πολυπλέκτη εάν η τιμή του Gradient είναι μεγαλύτερη ενός κατωφλιού, στην περίπτωσή μας 80, μετατρέπουμε το pixel σε λευκό 255, ενώ διαφορετικά σε μαύρο 0.

Για την δεύτερη επεξεργασία που γίνεται παράλληλα, η διαφορά είναι ότι θα υλοποιηθεί πρώτα κατωφλίωση στην εικόνα. Πάλι με την βοήθεια ενός πολυπλέκτη θα γίνει επεξεργασία στις τιμές του pixel και αν είναι μικρότερες από τις τιμές του κατωφλιού, στην περίπτωσή μας 30 θα μετατραπεί σε μαύρο 0 ενώ διαφορετικά θα του δώσουμε τιμή 255. Ύστερα θα εφαρμοστεί το φίλτρο Sobel με ίδιο τρόπο όπως και περιγράφεται και παραπάνω και θα καταλήξουμε στο τελικό αποτέλεσμα της δεύτερης επεξεργασίας. Τα δύο αποτελέσματα θα συνδυαστούν με ένα πολυπλέκτη και δυο πύλες And, δηλαδή κάθε φορά που οι δύο επεξεργασίες έχουν κοινή τιμή και αυτή είναι λευκή 255 θα μετατρέπεται το pixel σε λευκό pixel του χάρτη ακμών ενώ διαφορετικά θα μετατρέπεται σε μαύρο 0. Έτσι παράγεται ο τελικός χάρτης ακμών.

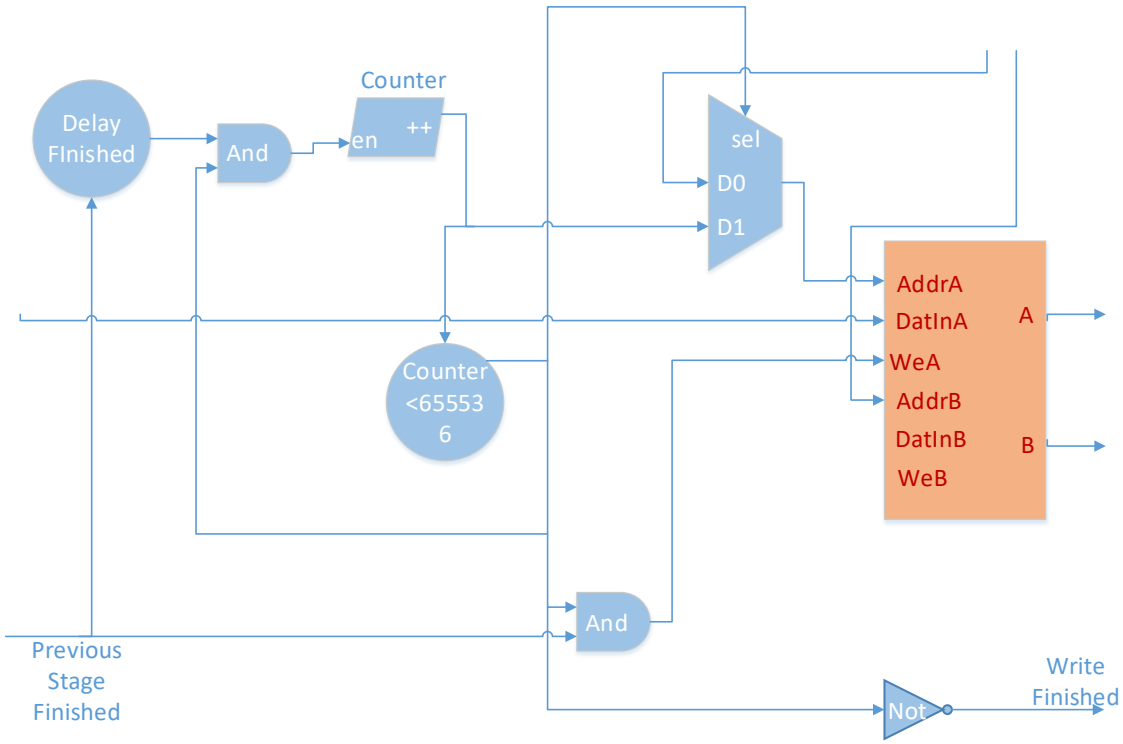
Βασική λειτουργία της σχεδίασης μας είναι ότι μετά από κάθε στάδιο, γίνεται η αποθήκευση του αποτελέσματος σε μία BRAM. Στο συγκεκριμένο σημείο θα αποθηκευτεί ο χάρτης ακμών. Επιπλέον επειδή αποτελεί ένα στάδιο, που και πριν από αυτό και έπειτα υπάρχει κάποιο άλλο, δίνεται η δυνατότητα να περιγράφει αναλυτικά η διαδικασία εγγραφής και πως επιτυγχάνεται ο συγχρονισμός ανάμεσα στα στάδια.

Το πρώτο πράγμα που πρέπει να αναφερθεί είναι ότι από το πρώτο στάδιο, δηλαδή αυτό της εγγραφής της αρχικής εικόνας στον Image Buffer μεταφέρεται το σήμα previous stage finished. Οπότε η εγγραφή του χάρτη ακμών στην δεύτερη BRAM, τον Edge map Buffer για να ξεκινήσει λαμβάνει υπόψιν ότι όλη η προηγούμενη διεργασία έχει τελειώσει. Όταν το previous stage finished είναι 1 ξεκινά ένας μετρητής για να υπολογιστεί η καθυστέρηση που έχει αναφερθεί προηγουμένως και αφορά τις μάσκες με τις οποίες έγινε συνέλιξη. Αυτή η καθυστέρηση για όλες τις μάσκες είναι 512 κύκλοι και θα έχουμε σωστά αποτελέσματα μετά το πέρας αυτών.
Με το που ο μετρητής της καθυστέρησης τερματίσει ξεκινά ένας δεύτερος μετρητής, ο οποίος, όπως και στο στάδιο 1 θα αναλάβει τον εξής ρόλο, αφενός αποτελεί διεύθυνση στην Bram και αφετέρου, όταν τερματίζει γίνεται γνωστό ότι τερμάτισε η εγγραφή του χάρτη ακμών στην Bram. Για αυτό τον σκοπό, όσο ο μετρητής είναι μικρότερος του μεγέθους του χάρτη ακμών δηλαδή

65536, το we της BRAM δέχεται 1, ενώ με το που φτάσει αυτό το όριο το σήμα write finished ενεργοποιείται για να ξεκινήσει με την σειρά του το στάδιο που ακολουθεί.

Σε αυτό το σημείο πρέπει να αναφερθεί μία επιπλέον ιδιαιτερότητα του Edge Map Buffer. Όπως θα διαπιστωθεί και θα εξηγηθεί και αργότερα το επόμενο στάδιο χρειάζεται δύο χάρτες ακμών, έναν κανονικό και έναν ανεστραμμένο κατά 90 μοίρες. Για αυτό το λόγο και ο Edge Map Buffer είναι dual port Bram. Κατά την εγγραφή χρησιμοποιείται η διεύθυνση A. Κατά το διάβασμα στο επόμενο στάδιο χρησιμοποιείται και η διεύθυνση A και η B. Για αυτό το σκοπό υπάρχει και ένας πολυπλέκκτης , για να ελέγχεται αν θα χρησιμοποιηθεί η διεύθυνση εγγραφής ή του διαβάσματος.

Figure 4.7: Εγγραφή στον Edge Map Buffer



4.2.3 Ψηφοφορία για τα Πιθανά Κέντρα

Σε αυτό το στάδιο χρησιμοποιείται ο χάρτης ακμών για να βρεθεί το κέντρο μέσω ψηφοφορίας ανάμεσα στα πιθανά κέντρα. Θα γίνει ψηφοφορία στον άξονα χ και στον άξονα y ξεχωριστά. Πιο συγκεκριμένα για κάθε ζευγάρι λευκών σημείων θα βρίσκεται το μέσο της απόστασής τους και θα καταχωρείται σε αυτή την θέση μία ψήφος. Εφόσον η διαδικασία γίνεται ξεχωριστά στον άξονα των χ και y εξασφαλίζεται παραλληλία των δύο αυτών διεργασιών, αλλά και επιπλέον χρειάζεται να διαβάζεται ο Edge Map Buffer από δύο διαφορετικές διευθύνσεις, μια για κάθε άξονα και κατ' επέκταση. Αυτός είναι και ο λόγος που χρησιμοποιείται dual port BRAM.

Πιο συγκεκριμένα, θα διαβαστεί η εικόνα δύο φορές, μία κανονικά και μία που προσομοιώνει περιστροφή κατά 90 μοίρες. Στην συνέχεια θα εφαρμοστεί και στις δύο εικόνες η ίδια επεξεργασία με την διαφορά ότι η δεύτερη εφόσον τα στοιχεία της εικόνας θα διαβάζονται σαν να έχει περιστραφεί κατά 90 μοίρες θα μας δίνει τους ψήφους για τον άξονα y, ενώ η κανονική για τον άξονα χ. Οι δύο ψηφοφορίες όπως έχει ήδη αναφερθεί θα γίνονται παράλληλα.

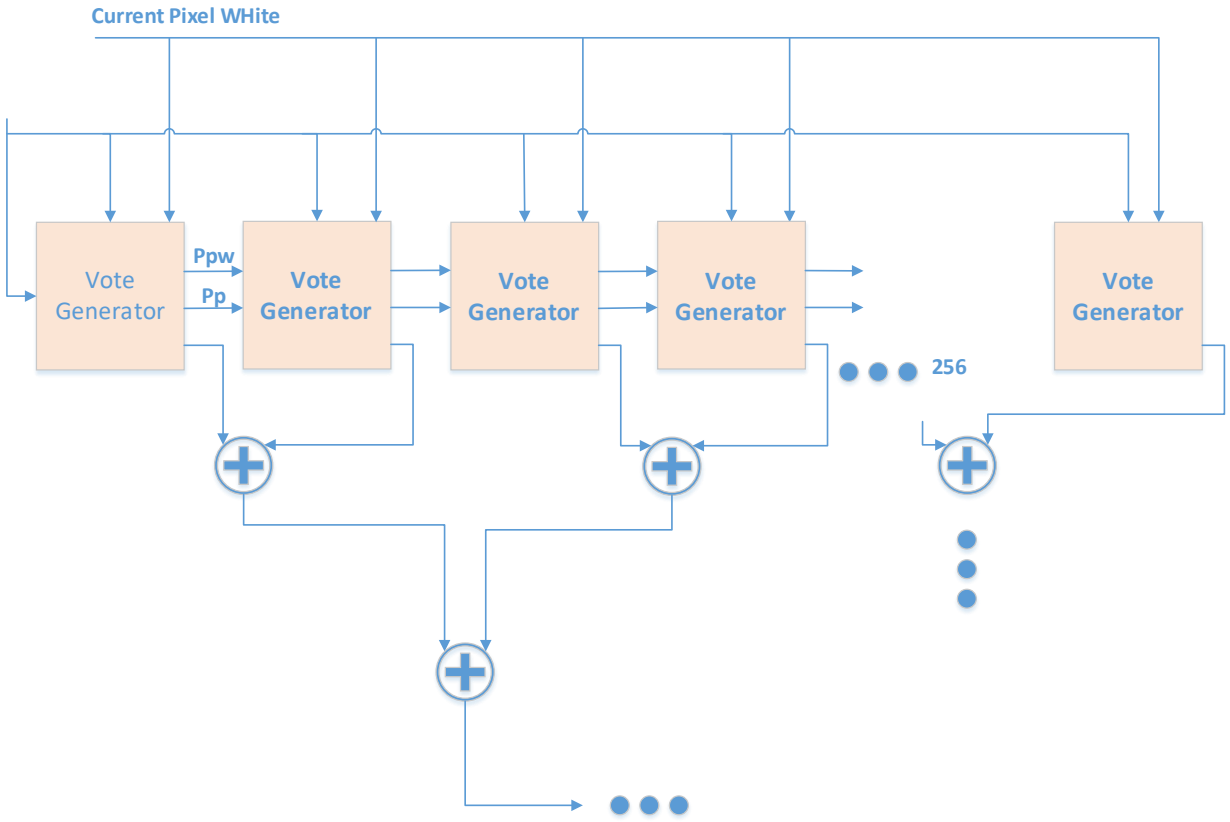
Για να επιτευχθεί η προσομοίωση της περιστροφής, χρησιμοποιούνται δύο μετρητές, ο μετρητής μεγάλης διάστασης i και ο μετρητής μικρής διάστασης j. Οι δύο μετρητές θα προσομοιώσουν την περιστροφή καθώς θα ρυθμίσουν την διεύθυνση με την οποία διαβάζουμε. Ο δεύτερος μετράει από το 0 μέχρι το 255. Όταν ο μετρητής μικρής διάστασης j γίνεται 255, ο μετρητής μεγάλης διάστασης i αυξάνεται κατά 1 και ο πρώτος γίνεται reset και αρχίζει από την αρχή την μέτρησή του. Για την κανονική εικόνα η λειτουργικότητα αυτή περιγράφεται ως εξής. Ο μετρητής j μετράει 1-1 τα στοιχεία μιας γραμμής και με το που γίνει 255 ο μετρητής i αυξάνεται για να προσδιοριστεί ότι περνάμε σε επόμενη γραμμή. Υπό αυτή την έννοια για να επεξεργαστούμε την κανονική εικόνα θα χρησιμοποιείται σαν διεύθυνση το i * 255 + j , ενώ για την εικόνα που θεωρούμε ότι έχει

περιστραφεί, χρησιμοποιείται σαν διεύθυνση το j * 255 + i. Έχοντας σαν είσοδο αυτές τις δύο διευθύνσεις, γίνεται ανάγνωση της εικόνας από τον Edge map Buffer.

Έπειτα η ψηφοφορία αν και γίνεται δυο φορές παράλληλα, με διαφορετική είσοδο κάθε φορά, είναι ίδια και για τους δύο άξονες. Για κάθε λευκό σημείο, κάθε γραμμής υπολογίζεται το μέσο της απόστασης του με τα υπόλοιπα λευκά στοιχεία της γραμμής. Το μέσο της απόστασης δύο σημείων περιγράφεται ως το άθροισμά των διευθύνσεων διαιρεμένο δια 2 και έχει σαν αποτέλεσμα ένα υποψήφιο σημείο κέντρου στο οποίο πρέπει να καταχωρηθεί μία ψήφος για τον συγκεκριμένο άξονα.

Για αυτό τον σκοπό χρησιμοποιούνται για κάθε άξονα 256 Vote Generator. Ο κάθε ένας έχει μέσα μία BRAM 256 θέσεων στις οποίες καταχωρούνται οι ψήφοι. Οι Vote Generator συνδέονται μεταξύ τους σε σειρά. Οι διευθύνσεις των Pixel εισέρχονται στον πρώτο Vote Generator μαζί με ένα σήμα για το αν σε αυτήν την διεύθυνση υπάρχει λευκό pixel ή όχι. Σε κάθε κύκλο ρολογιού μεταβιβάζονται αλυσιδωτά προς τους επόμενους Vote Generator, που ο κάθε ένας τους έχει μέσα δύο καταχωρητές για να κρατάνε αυτά τα δύο σήματα (Previous Pixel , Previous Pixel White). Παράλληλα κάθε καινούρια διεύθυνση μπαίνει σαν είσοδος σε όλους τους Vote Generator και συγκρίνεται με την τιμή που έχει μέσα. Αν και τα δύο pixel είναι λευκά υπολογίζεται η διεύθυνση που αντιστοιχεί στο μέσο της απόστασής τους και στη συγκεκριμένη θέση της BRAM αυξάνεται η τιμή κατά ένα καταμετρώντας μία ψήφο. Η διαδικασία επαναλαμβάνεται για κάθε καινούριο λευκό pixel της γραμμής τα οποία συγκρίνονται με όλα τα προηγούμενα λευκά και καταχωρούνται ψήφοι στην αντίστοιχη θέση. Είναι τυχαίο σε ποιους Vote Generator θα καταχωρηθούν οι ψήφοι για αυτό και στο τέλος υπολογίζεται για κάθε διεύθυνση το άθροισμα των ψήφων συνυπολογίζοντας τα αποτελέσματα από όλους τους Vote Generator. Το άθροισμα γίνεται με την μορφή δέντρου για να μειωθούν οι πράξεις που χρειάζονται. Η Διαδικασία αυτή γίνεται και στον άξονα χ και στον y.

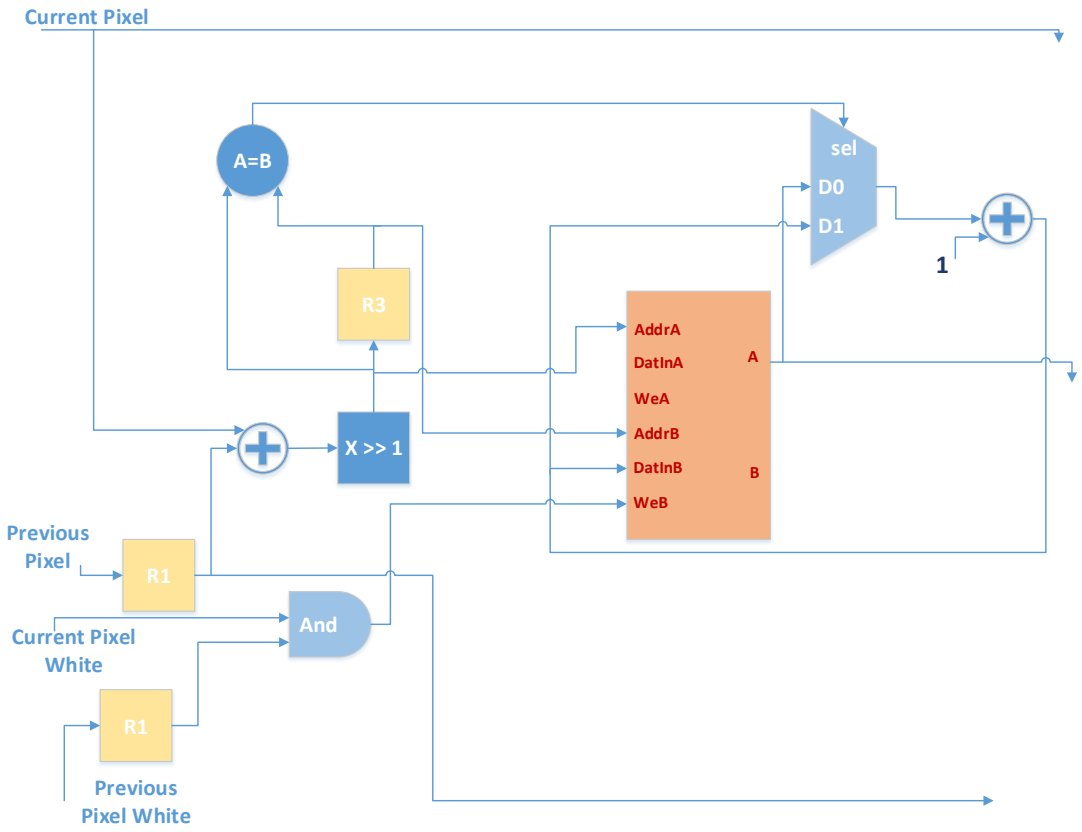
Figure 4.8: Σχεδιάγραμμα των Vote Generator και της λειτουργίας τους



Εσωτερικά οι Vote Generator περιέχουν τους καταχωρητές που κρατάνε τα σήματα Previous Pixel και Previous Pixel White. Αρχικά, όταν και το Pixel που έχει ο καταχωρητής μέσα και το καινούριο που έρχεται στον Vote Generator είναι και τα δύο λευκά, ενεργοποιείται η εγγραφή στον Vote Buffer ο οποίος είναι BRAM 256 θέσεων. Έπειτα γίνονται οι πράξεις με έναν αθροιστή και με ολίσθηση ένα bit δεξιά για να υλοποιηθεί η διαδικασία εύρεσής του μέσου της απόστασής τους. Το αποτέλεσμα αποτελεί την διεύθυνση από την οποία θα διαβαστεί το περιεχόμενό της, θα αυξηθεί κατά 1 και θα ξαναγραφτεί. Σε αυτό το σημείο χρειάζεται λίγη προσοχή, αν σε δύο

διαδοχικούς κύκλους ρολογιού χρειάζεται να προστεθεί ψήφος στην ίδια διεύθυνση. Για αυτό το λόγο χρησιμοποιείται ένας καταχωρητής για να γίνεται σύγκριση των δύο διευθύνσεων, αλλά και ένας πολυπλέκτης στο ενδεχόμενο που είναι ίδια να μην διαβαστεί το περιεχόμενο της Bram, γιατί δεν θα έχει προλάβει να γίνει η προηγούμενη εγγραφή στην συγκεκριμένη διεύθυνση, αλλά αντίθετα το αποτέλεσμα του προηγούμενου κύκλου να αυξηθεί κατά 1 και να εγγραφεί εκ νέου στον Vote Generator. Σημαντικό είναι ότι μετά το πέρας της επεξεργασίας σε μία γραμμή, δηλαδή μετά από 256 κύκλους πρέπει να αδειάσουν όλοι οι καταχωρητές που περιείχαν τα σήματα Previous Pixel και Previous Pixel White και η διαδικασία να ξεκινήσει από την αρχή, προστέθωντας καινούριους ψήφους στους ήδη υπάρχοντες που έχουν αποθηκευτεί στους Vote Buffer. Με αυτό τον τρόπο, όταν προσπελαστεί ολόκληρη η εικόνα το σύνολο των 256 Vote Buffer θα έχουν αποθηκευμένους τους ψήφους για κάθε σημείο-διεύθυνση.

Figure 4.9: Το εσωτερικό ενός Vote Generator.



4.2.4 Εύρεση του Υποψήφιου Κέντρου με τις Περισσότερες Ψήφους

Όπως περιεγράφηκε προηγουμένως οι 256 Vote Buffer x και οι 256 Vote Buffer y κρατάνε τις ψήφους για τους αντίστοιχους άξονες. Αφού τερματίσει λοιπόν το στάδιο της ψηφοφορίας, σε αυτό το στάδιο με την βοήθεια ενός μετρητή ο οποίος μετράει από το 0 έως το 256 ζητείται από τους Vote Buffer το αποτέλεσμα για την αντίστοιχη διεύθυνση, που είναι η τιμή του μετρητή.

Χρησιμοποιώντας τους αθροιστές που προσθέτουν τις τιμές όλων των Vote Buffer με την μορφή δέντρου, προκύπτουν οι ολικοί ψήφοι για την συγκεκριμένη διεύθυνση.Αυτό επαναλαμβάνεται για όλες τις διευθύνσεις. Για την συνέχεια τις επεξεργασίας χρησιμοποιούνται δύο καταχωρητές. Ο ένας για τις διευθύνσεις και ο άλλος για την τιμή των ψήφων. Κάθε φορά που έρχεται ένα καινούριο αποτέλεσμα συγκρίνεται η τιμή του με την τιμή του καταχωρητή. Εάν το καινούριο αποτέλεσμα είναι μεγαλύτερο της ήδη υπάρχουσας τιμής τότε αποθηκεύεται η καινούρια τιμή στον καταχωρητή. Σε όλο αυτό εξυπηρετεί η χρήση ενός πολυπλέκτη που ρυθμίζει ποια τιμή θα αποθηκευτεί ανάλογα με την σύγκριση.

Επιπλέον αυτό που ενδιαφέρει την σχεδίασή μας είναι κρατηθεί η τιμή της διεύθυνσης γιατί μας ενδιαφέρει, όχι το πόσες είναι οι περισσότερες ψήφοι αλλά το ποιο σημείο τις συγκέντρωσε. Έτσι στον άλλον καταχωρητή κάθε φορά που γίνεται εναλλαγή της τιμής καταχωρείται και η καινούρια διεύθυνση.

Αυτή η διαδικασία γίνεται τόσο για τις ψήφους του άξονα χ όσο και για τους ψήφους του άξονα y. Έτσι στο τέλος της θα είναι αποθηκευμένες σε δύο καταχωρητές το σημείο που συγκέντρωσε τις περισσότερες ψήφους στον χ άξονα και το σημείο που συγκέντρωσε τις περισσότερες ψήφους στον y άξονα.

4.2.5 Αποθήκευση της Γραμμής και της Στήλης που Αντιστοιχούν στους Βέλτιστους Υποψήφιους.

Για Τον υπολογισμό της ακτίνας που είναι το επόμενο στάδιο θα χρειαστεί να επιστρέψουμε στον Edge Map Buffer και να πάρουμε από την κανονική εικόνα ολόκληρη τη γραμμή και τη στήλη που μας ενδιαφέρουν σύμφωνα με το σημείο που έχει προκύψει σαν κέντρο. Την γραμμή θα την αποθηκεύσουμε στον Row Buffer ενώ την στήλη θα την αποθηκεύσουμε στον Column Buffer. Και οι δύο αποτελούνται από BRAM 256 θέσεων.

Προκειμένου να γίνει αυτό θα χρειαστεί να δημιουργηθεί μία διεύθυνση η οποία θα εξυπηρετεί στο να γίνει η ανάγνωση από τον Edge Map Buffer. Έτσι χρησιμοποιείται ένας μετρητής (στην εξίσωση περιγράφεται ως a) ο οποίος μετράει μέχρι το 255 , παρ' όλα αυτά, σε αντίθεση με προηγούμενα στάδια δεν αποτελεί αυτή καθ' αυτή η τιμή του, την διεύθυνση. Αντίθετα η τιμή του a είναι η διεύθυνση που σε κάθε χτύπο του ρολογιού θα εγγράφεται η πληροφορία στους Row και Column Buffer. Για την διεύθυνση που διαβάζεται από τον Edge Map Buffer χρησιμοποιείται για τις γραμμές η τιμή a * 256 + y, όπου y είναι το σημείο του κέντρου στον y άξονα και για τις στήλες η τιμή a * 256 + χ, όπου χ είναι το σημείο του κέντρου στον χ άξονα.

Έτσι σε κάθε κύκλο ρολογιού διαβάζεται το αντίστοιχο pixel από τον Edge Map και αποθηκεύεται στους Row και Column Buffer. Εδώ πρέπει να αναφερθεί ότι ο κάθε ένας, στην πραγματικότητα αποτελείται από δύο ίδιες Dual BRAM 256 θέσεων καθώς για την εύρεση της ακτίνας χρειάζεται να εξεταστούν διάφορες περιπτώσεις για τις οποίες χρειάζονται 4 έξοδοι από τις BRAM.

4.2.6 Εύρεση της Ακτίνας.

Για να υπολογιστεί η ακτίνα, η διαδικασία που ακολουθείται είναι η εξής. Για κάθε άξονα χρησιμοποιείται το σημείο που έχει αναφορά στο κέντρο. Έπειτα με την βοήθεια ενός μετρητή c , ο οποίος ξεκινά την μέτρησή του, όταν τελειώνει το προηγούμενο στάδιο και που μετράει ως το 127 αρχίζουν να παράγονται διευθύνσεις που έχουν αναφορά δεξιά και αριστερά από το κεντρικό σημείο, προσθέτοντας σε κάθε κύκλο την αύξηση που προκύπτει από τον μετρητή. Για παράδειγμα εάν είμαστε στον άξονα χ οι δύο διευθύνσεις θα είναι x + c και x – c. Στόχος είναι, με τις συγκρίσεις που θα ακολουθήσουν, πάνω στην γραμμή να εντοπιστεί πότε εμφανίζονται συμμετρικά, δεξιά και αριστερά από το κέντρο δύο λευκά pixel.

Επιπλέον για να αποφευχθεί το πρόβλημα επειδή η εικόνα είναι αρκετά μικρή να μην εμφανίζονται τα σημεία ενός κύκλου ακριβώς συμμετρικά, αλλά να λαμβάνουμε υπόψη και τα διπλανά τους, έχουμε προσθέσει και τις διπλανές τους θέσεις, δηλαδή x –c - 1 και x +c + 1 αποκτώντας εν τέλει 4 διευθύνσεις. Οι συγκρίσεις που θα γίνουν για το αν υπάρχει συμμετρία ανάμεσα σε λευκά Pixel είναι εν τέλει 3 και συγκεκριμένα μεταξύ των παρακάτω διευθύνσεων: 1) Τιμή[x – c] = [x + c] 2) x – c = x + c + 1 3) x – c - 1 = x + c . Εάν οποιαδήποτε από αυτές τις συγκρίσεις είναι αληθής τότε αποθηκεύεται σε έναν καταχωρητή η τιμή της ακτίνας. Η σχεδίαση μας έχει προβλέψει να βρίσκει 3 πιθανές ακτίνες καθώς η πληροφορία έχει περιοριστεί από τα προηγούμενα βήματα και θεωρούμε ότι αρκούν. Επομένως το τελικό αποτέλεσμα είναι 3 πιθανές ακτίνες για τον άξονα χ και 3 πιθανές ακτίνες για τον άξονα y.

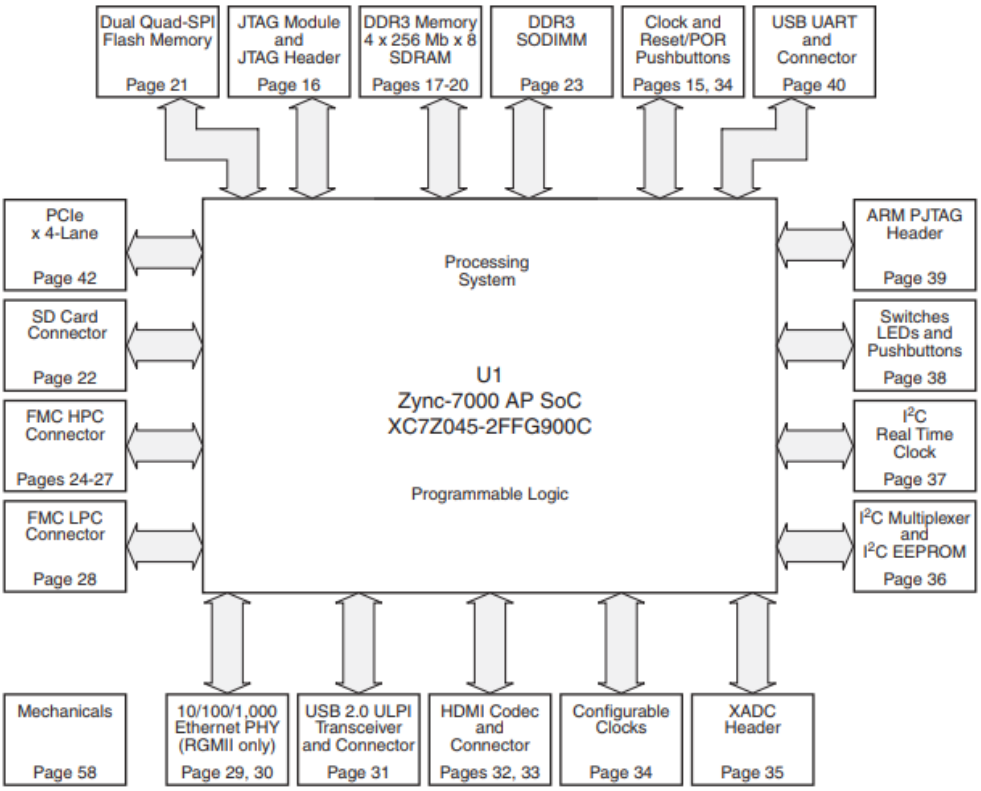
5 Επιβεβαιωση Λειτουργιας, Εξαγωγη Αποτελεσματων και Αξιολογηση

5.1 Εξαγωγή Αποτελεσμάτων

Το Vivado IDE είναι το γραφικό περιβάλλον χρήστη για το Vivado Design Suite. Όλα τα εργαλεία του Vivado Design Suite είναι γραμμένα σε native Tcl interface και όλες οι εντολές που περιλαμβάνονται είναι διαθέσιμες στο IDE είτε μέσω του γραφικού περιβάλλοντος είτε μέσω της Tcl κονσόλας. Κατά την διάρκεια της διαδικασίας της σχεδίασης είναι εφικτή ανάλυση της σχεδίασης καθώς και προσδιορισμός διάφορων περιορισμών, όπως οι εκτιμήσεις για τον χρόνο ή για την κατανάλωση ενέργειας, μετά τη διαδικασία του Synthesis. Επιπλέον είναι διαθέσιμες αρκετές άλλες επιλογές κατά την διάρκεια της εκτέλεσης της σχεδίασης, όπως το Simulation, εκτιμήσεις κατανάλωσης πόρων κ.α. που βοηθούν στην βελτιστοποίηση των αποτελεσμάτων μιας σχεδίασης.

Το Xilinx SDK είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης για την ανάπτυξη Software εφαρμογών που στοχεύουν σε ενσωματωμένους επεξεργαστές της Xilinx. Το SDK δουλεύει σε συνεργασία με σχεδιάσεις Hardware που έχουν δημιουργηθεί μέσω του Vivado Design Suite. Στο SDK είναι εφικτός ο προγραμματισμός σε γλώσσες προγραμματισμού C/C++ μέσω του σχετικά απλού Code Editor που διαθέτει. Με αυτόν τον τρόπο διευκολύνεται η υλοποίηση μίας επιπλέον λειτουργικότητας που θέλει να προσθέσει ο σχεδιαστής Hardware στην σχεδίασή του.

Figure 5.1: ZC706



Η ZC706 evaluation board για το XC7Z045 SoC παρέχει ένα περιβάλλον Hardware για την ανάπτυξη και την αξιολόγηση σχεδιάσεων που στοχεύουν σε Zynq®-7000 XC7Z045-2FFG900C SoC. Η ZC706 παρέχει όλα τα χαρακτηριστικά που είναι κοινά στα ενσωματωμένα συστήματα όπως DDR3 SODIMM και component memory, 4 γραμμών PCI Express® interface, Ethernet PHY, γενικού σκοπού I/O και 2 UART interfaces. Επιπλέον έχει τα ακόλουθα χαρακτηριστικά.

• Zynq-7000 XC7Z045-2FFG900C SoC
• 1 GB DDR3 memory SODIMM on the programmable logic (PL) side
• 1 GB DDR3 component memory (four [256 Mb x 8] devices) on the processing system (PS) side
• Two 128 Mb Quad-SPI (QSPI) flash memory (Dual Quad-SPI)
• USB 2.0 ULPI (UTMI+ low pin interface) transceiver with micro-B USB connector
• Secure Digital (SD) connector • USB JTAG interface via Digilent module with micro-B USB connector
• Clock sources:
° Fixed 200 MHz LVDS oscillator (differential)
° I 2C programmable LVDS oscillator (differential)
° Fixed 33.33 MHz LVCMOS oscillator (single-ended)
° Subminiature version A (SMA) connectors (differential)
° SMA connectors for GTX transceiver clocking (differential)

Στα Memory Mapped Axi (AXI3, AXI4, and AXI4-Lite) όλες οι προσπελάσεις διέπονται από την εξής λογική κατά την οποία, οι προσπελάσεις στοχεύουν με μία συγκεκριμένη διεύθυνση εντός του χώρου μνήμης του συστήματος και των δεδομένων που πρέπει να μεταφερθούν. Τα Memory Mapped συστήματα συχνά παρέχουν έναν πιο ομοιογενή τρόπο για προσπέλαση ενός συστήματος.

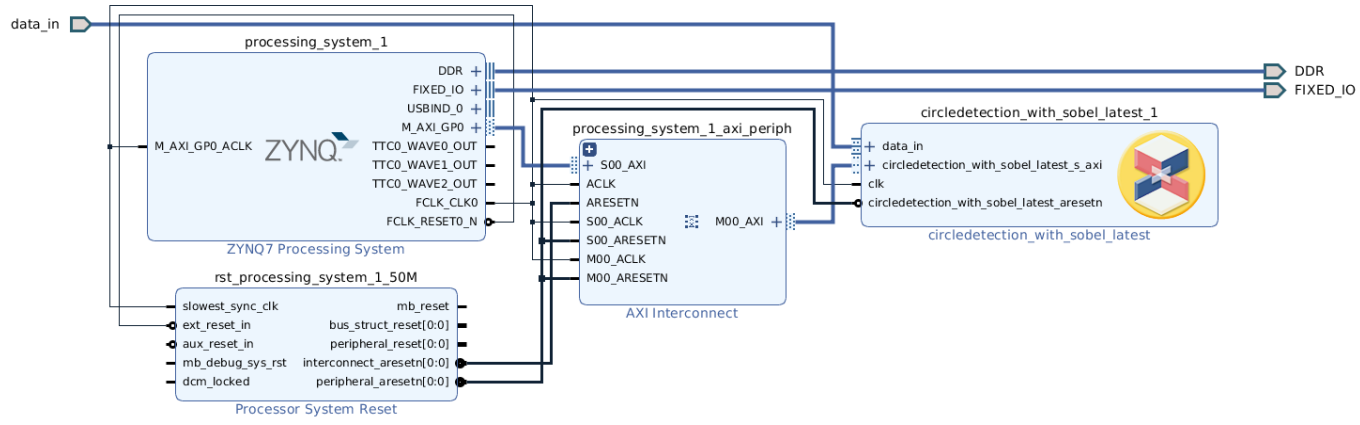
Το AXI4-Stream πρωτόκολλο χρησιμοποιείται για εφαρμογές που η λογική της διεύθυνσης δεν είναι παρούσα ή δεν είναι αναγκαία. Κάθε AXI4-Stream ενεργεί σαν ένα μοναδικό κανάλι για την λειτουργία ροής δεδομένων μεταξύ των μονάδων, με την μορφή χειραψίας. Καθ' αυτήν την λειτουργία, ο μηχανισμός μεταφοράς δεδομένων, είναι προκαθορισμένος και αποτελεσματικός, αλλά δεν περιέχει ενοποιημένο περιεχόμενο διευθύνσεων.

Μία άλλη προσέγγιση είναι η δημιουργία συστημάτων που συνδυάζουν AXI4-Stream και AXI memory mapped μαζί. Συχνά μία DMA μηχανή χρησιμοποιείται για την μεταφορά Stream από την μνήμη. Ουσιαστικά είναι ένα έτοιμο module που μας παρέχει η Xilinx που διευκολύνει την χρησιμοποίηση Axi Stream και την επικοινωνία μεταξύ μνήμης, επεξεργαστή και Accelerator.

Όσον αφορά την μεθοδολογία που ακολουθήσαμε για την μεταφορά της σχεδίασης μας στην FPGA, η αρχή έγινε υλοποιώντας την σχεδίασή μας μέσω του Xilinx System Generator στο Simulink της Matlab. Το Xilinx System Generator μας δίνει την δυνατότητα χρησιμοποιώντας block της Xilinx να φτιάξουμε μία σχεδίαση της οποίας μπορούμε να παραγάγουμε τον κώδικα προκειμένου να προγραμματιστεί ο Accelerator της πλακέτας μας. Προκειμένου να γίνει αυτό μας δίνεται η δυνατότητα να ρυθμίσουμε κάποιες παραμέτρους που αφορούν, ποια πλακέτα θέλουμε να προγραμματίσουμε, σε τι μορφή, χρονισμούς κ.λπ. Όπως φαίνεται και στο παρακάτω σχήμα, ανοίγοντας το παράθυρο του System Generator Token, επιλέγουμε την ZC706, ρυθμίζουμε τον τρόπο που θέλουμε να παραχθεί το Project μας (μας δίνεται η επιλογή για Co-Simulation, αλλά εμείς επιλέγουμε IP Catalog) και κάνοντας Generate παράγουμε τον κώδικα του Project μας που μπορούμε να το ανοίξουμε στο Vivado για να το επεξεργαστούμε περεταίρω.

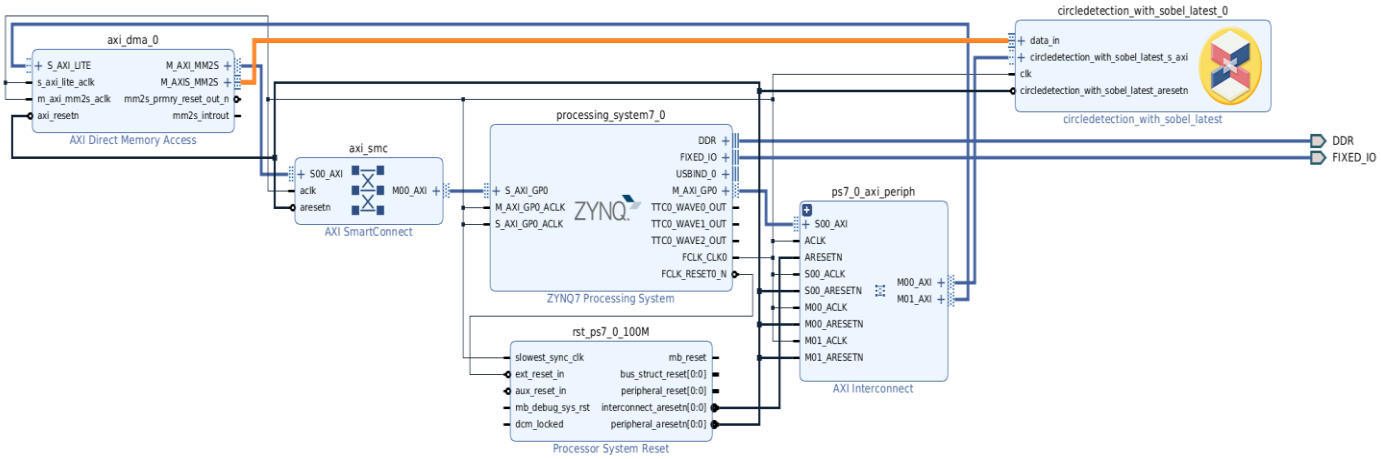
Figure 5.2: Η ρύθμιση του System Generator Token

Figure 5.3: Η εξαγωγή του Project από το Xilinx System Generator χωρίς μετατροπές



Ανοίγοντας το Project στο Vivado όπως φαίνεται και στο παραπάνω σχήμα διαπιστώνουμε αυτό που ήταν αναμενόμενο, ότι το Data in του Accelerator είναι external και δεν συνδέεται πουθενά. Αυτό προκύπτει διότι έχουμε επιλέξει εμείς να ρυθμίσουμε τον Accelerator να δέχεται μέσω Axi Stream τα δεδομένα της εικόνας (σε αντίθεση με τις εξόδους του που είναι σε μορφή Axi Memory Maped) προκειμένου να τα λαμβάνει πιο αποδοτικά και χωρίς καθυστερήσεις, εξασφαλίζοντας ότι θα έχουμε ένα Pixel σαν είσοδο σε κάθε κύκλο ρολογιού. Αντίθετα όμως ο επεξεργαστής δίνει τα δεδομένα σε μορφή Axi Memory Mapped. Είναι αναγκαίο για να λυθεί το παραπάνω πρόβλημα να χρησιμοποιηθεί Dma Module το οποίο όπως έχει περιγράφει προηγουμένως μπορεί να συνδυάσει Axi Memory Maped και Axi Stream. Κάνοντας τις απαραίτητες προσθέσεις καταλήγουμε εν τέλει στο παρακάτω Data Path του Project για το οποίο παράγουμε το Bitstream που θα κατεβάσουμε στην πλακέτα.

Figure 5.4: Το τελικό Project στο Vivado



Τελευταίο βήμα είναι να περάσουμε την εικόνα που πρόκειται να υποστεί επεξεργασία στην μνήμη. Σε αυτό το σημείο πρέπει να επισημανθεί ότι γίνεται μία προ επεξεργασία της εικόνας στην Matlab προκειμένου να την φέρουμε στην μορφή που χρειάζεται. Πιο συγκεκριμένα την

κάνουμε Resize σε συγκεκριμένο μέγεθος (256 x 256) και την αποθηκεύουμε σαν .pgm αρχείο. Επιπλέον φτιάξαμε μία εφαρμογή σε γλώσσα προγραμματισμού C για να μετατρέπει την εικόνα σε κείμενο χαρακτήρα τις μορφής .c το οποίο μπορούμε να το ενσωματώσουμε στο SDK προκειμένου αφού μπει η εικόνα στην Ram , να αξιοποιηθεί. Ειδικότερα ο κώδικας που γράψαμε στο SDK διαβάζει την εικόνα, δίνοντας συγκεκριμένη διεύθυνση και μέγεθος, από την Ram και την στέλνει στον Accelerator, έπειτα γίνεται η απαραίτητη αναμονή μέχρι να τελειώσει η επεξεργασία του Accelerator και εν τέλει διαβάζουμε τα αποτελέσματα. Τέλος αφού έχουμε τα αποτελέσματα τα εισάγουμε στην Matlab, προκειμένου να επιλεγεί η τελική ακτίνα και να υπάρχει και οπτική απεικόνιση των αποτελεσμάτων προκειμένου να μπορεί να αξιολογηθεί.

5.2 Αποτελέσματα

Πριν περάσουμε στην τελική παρουσίαση τον αποτελεσμάτων της σχεδίασής μας έχει αξία να γίνει αναφορά σε σχέση με το ποια είναι τα αποτελέσματα της απόδοσης της FPGA. Πιο συγκεκριμένα στα παρακάτω σχήματα φαίνονται οι αποδόσεις χρόνου με ρολόι 50 MHZ (μπορούμε με λίγες μετατροπές να βελτιώσουμε κατά πολύ το ρολόι), η κατανάλωση ενέργειας καθώς και οι πόροι που καταναλώνονται , όπως τα Dsp, το ποσοστό που καταλαμβάνεται από την διαθέσιμη BRAM κ.λπ.

Figure 5.5: Πληροφορίες σχετικά με την κατανάλωση ενέργειας.

Power	
Total On-Chip Power	**2.254 W**
Junction Temperature	29.0 C
Thermal margin	56.0 C (30.8 W)
Effective JA	1.8 C/W
Power Supplied To Off-Chip Devices	**0 W**

Figure 5.6: Οι πόροι της FPGA που χρησιμοποιήθηκαν.

Resource	Utilization	Utilization %	Available
Lut	12756	**5.84%**	**218600**
Lutram	175	0.25%	70400
Ff	9642	2.21%	437200
Bram	168.50	30.92%	545
Dsp	266	**29.56%**	**900**
Bufg	1	3.13%	32

Εκτεταμένη αναφορά πρέπει να γίνει και στην απόδοση χρόνου που έχει η FPGA. Για αυτό τον σκοπό όχι μόνο εξετάσαμε τον συνολικό χρόνο εκτέλεσης της σχεδίασής μας αλλά και εντρυφήσαμε σε πόσο χρόνο καταναλώνει κάθε επιμέρους στάδιο. Αυτό έγινε προκειμένου να μπορούμε να συγκριθούμε με την βιβλιογραφία και με αντίστοιχες δουλειές που μπορεί να μην είχαν σαν στόχο την αναγνώριση ίριδας συνολικά αλλά να υλοποιούν κάποιο επιμέρους στάδιο όπως είναι ο κυκλικός μετασχηματισμός Hough. Τα αποτελέσματα ήταν ικανοποιητικά καθώς η σχεδίασή μας κατά μέσο όρο τρέχει σε 3.9 ms, ενώ αν αναλογιστούμε τα επί μέρους στάδια το στάδιο της αποθήκευσης της εικόνας και το στάδιο της εύρεσης χάρτη ακμών, γίνονται στους ελάχιστους κύκλους, στους κύκλους που θα πάρει στο σύστημά μας να προσπελάσει μία φορά την εικόνα (Για εικόνα 256x256 έχουμε 256x256 κύκλους). Φυσικά ο τελικός χρόνος εκτέλεσης αυτών των σταδίων κρίνεται και από το ρολόι, όπου με βελτίωση του ρολογιού θα επιτευχθεί ακόμα μικρότερος χρόνος. Όσον αφορά τον κυκλικό μετασχηματισμό Hough οι υλοποιήσεις του στην βιβλιογραφία γίνονται σε 5 ms. Η διαφοροποίηση που εφαρμόσαμε προκειμένου να θυσιάσουμε λίγο απόδοση και να κερδίσουμε σε χρόνο κατέβασε τον χρόνο εκτέλεσής του στην δικιά μας υλοποίηση στα 1.3 ms. Παρακάτω παρουσιάζονται αναλυτικά οι πίνακες χρόνων.

Figure 5.7: Πληροφορίες σχετικά με τους θεωρητικούς κύκλους και τον χρόνο.

Στάδια Σχεδίασης	Θεωρητικοι Κυκλοι	Θεωρητικος Χρονος
Διαβασμα και Αποθηκευση	256x256	**1.31 ms**
Ευρεση Χαρτη ακμων	256x256+524	1.321 ms
Ψηφοφορια Για Κεντρο	256x256	1.31 ms
Ευρεση Του Κεντρου	256	0.005 ms
Αποθηκευση Κεντρου	256	0.005 ms
Υπολογισμος Ακτινας	256	0.005 ms

Figure 5.8: Πληροφορίες σχετικά με τους θεωρητικούς κύκλους και τον χρόνο.

Στάδια Χαρτη Ακμων	Θεωρητικοι Κυκλοι	Θεωρητικος Χρονος
Καθυστέρηση Line Buffer	256x2	**0.01 ms**
Καθυστερηση Register	12	~ 0 ms
Επεξεργασια Εικονας	256x256	1.31 ms

Figure 5.9: Πληροφορίες σχετικά με τους θεωρητικούς κύκλους και τον χρόνο.

Στάδια CHT	Θεωρητικοι Κυκλοι	Θεωρητικος χρονος
Ψηφοφορια Για Κεντρο	256x256	1.31 ms
Ευρεση Του Κεντρου	256	0.005 ms
Αποθηκευση Κεντρου	256	0.005 ms
Υπολογισμος Ακτινας	256	0.005 ms

Figure 5.10: Πληροφορίες σχετικά με τους θεωρητικούς κύκλους και τον χρόνο.

ΣΤΑΔΙΑ ΣΧΕΔΙΑΣΗΣ	ΘΕΩΡΗΤΙΚΟΙ ΚΥΚΛΟΙ	ΘΕΩΡΗΤΙΚΟΣ ΧΡΟΝΟΣ	ΠΡΑΓΜΑΤΙΚΟΣ ΧΡΟΝΟΣ	ΧΡΟΝΟΣ ΣΕ MATLAB
ΔΙΑΒΑΣΜΑ ΚΑΙ ΑΠΟΘΗΚΕΥΣΗ	256x256	**1.31 ms**		
ΕΥΡΕΣΗ ΧΑΡΤΗ ΑΚΜΩΝ	256x256+524	1.321 ms		
CHT	256x256+768	1.325 ms		
ΣΥΝΟΛΟ ΣΧΕΔΙΑΣΗΣ	256x256x3+1292	3.956 ms	3.982 ms	181.1 ms

Όσον αφορά τα αποτελέσματα της επιτυχημένης εύρεσης της κόρης, η υλοποίησή μας εξετάστηκε πάνω στο Dataset NIR εικόνων από μάτια, Casia Lamp. Το Dataset αυτό περιλαμβάνει εικόνες 415 ατόμων, του δεξιού και του αριστερού ματιού ξεχωριστά , κάτω από διαφορετικές συνθήκες φωτισμού, που μπορεί να μην φαίνονται στην αποτύπωση της εικόνας αλλά διαφοροποιούν το μέγεθος της κόρης σε κάθε συνθήκη φωτισμού. Όπως γίνεται κατανοητό ήταν αρκετά δύσκολο να εξετάσουμε όλες τις εικόνες και για αυτό επιλέξαμε εμείς 150 εικόνες για να τεστάρουμε τα αποτελέσματα. Η επιλογή έγινε με στόχο να καλυφθούν όσο το δυνατόν περισσότερες περιπτώσεις και συνθήκες που μπορεί να δυσκολεύουν την ανίχνευση.

Υπό αυτό το πρίσμα θα ήταν σχετικά αυθαίρετο να βγάλουμε ποσοστό επιτυχημένης εύρεσης καθώς θα μπορούσαμε να το ρυθμίσουμε εμείς ανάλογα με τις επιλογές που κάναμε. Αντίθετα τα αποτελέσματα τα χωρίζουμε σε καλά, μέτρια και κακά αποτελέσματα. Εδώ πρέπει να επισημανθεί, ότι η σχεδίασή μας αναζητά το κέντρο της κόρης και την ακτίνα της. Για αυτό το λόγο πρέπει να τονιστεί ότι μπορεί ένα αποτέλεσμα να έχει μπει στην κατηγορία «κακών αποτελεσμάτων», παρ. όλα αυτά ο εντοπισμός του κέντρου να δουλεύει αρκετά καλά και αυτό που να είναι αρκετά προβληματικό να είναι η ακτίνα.

Πιο συγκεκριμένα εξετάστηκαν οι περιπτώσεις απλής εικόνας, εικόνες με θόρυβο από βλεφαρίδες, εικόνες με γωνία του βλέμματος, εικόνες που η κόρη κόβεται, εικόνες που η εύρεση της ακτίνας ήταν αποτυχημένη λόγω της συμμετρίας των αντανακλάσεων και εικόνες που συνδύαζαν κάποιες από τις παραπάνω περιπτώσεις. Η υλοποίησή μας ανταποκρίθηκε σε μεγάλο βαθμό και είχε αρκετά καλά αποτελέσματα. Το μόνο που είναι αρκετά προβληματικό είναι η μη επιτυχημένη πρόβλεψη της ακτίνας όταν συνδυάζονται κάποια από τα προβλήματα.

Στην συνέχεια δίνεται ο Πίνακας (Figure 5.12) που δείχνει την διαφορά των αποτελεσμάτων της FPGA σε σχέση με αυτά της Matlab. Στον εν λόγω πίνακα προκύπτει ότι στα καλά αποτελέσματα η διαφορά είναι 1-2 Pixel ενώ στα πιο δύσκολα μπορεί να φτάσει και στα 5-6 Pixel πράγμα που δείχνει ότι η σχεδίαση μας χάνει κάποια λεπτομέρεια σε σχέση με την υλοποίηση στην Matlab. Αυτό είναι λογικό αν αναλογιστούμε ότι στην Matlab τα ίδια φίλτρα μπορούν να έχουν καλύτερη απόδοση σε σχέση με τα αντίστοιχα στο Hardware. Έτσι ανάλογα με την απόκλιση που υπάρχει σε σχέση με τα αποτελέσματα της Matlab, κατηγοριοποιήθηκαν τα αποτελέσματα σε καλά, μέτρια και κακά, όπως δείχνει ο πίνακας στο Figure 5.11.

Figure 5.11 Πίνακάς στον οποίον σύμφωνα με την απόκλιση των αποτελεσμάτων της FPGA σε σχέση με τα αποτελέσματα της Matlab κατηγοριοποιούνται τα αποτελέσματα.

	ΚΑΛΟ ΑΠΟΤΕΛΕΣΜΑ	ΜΕΤΡΙΟ ΑΠΟΤΕΛΕΣΜΑ	ΚΑΚΟ ΑΠΟΤΕΛΕΣΜΑ	
ΚΕΝΤΡΟ	Αποκλιση 1-2 P.	Αποκλιση 1-2 P.	Αποκλιση 1-2 P.	Αποκλιση 3-7 P.
ΑΚΤΙΝΑ	Αποκλιση 1-2 P.	Αποκλιση 3-7 P.	Καθόλου Εύρεση	Αποκλιση 3-7 P.

Figure 5.12 Πίνακάς σχέσης των αποτελεσμάτων σε κέντρο και ακτίνα της Fpga με την Matlab (για τα 7 Figure που παρουσιάζονται στην συνέχεια και άλλα 8 τυχαία παραδείγματα).

ΕΙΚΟΝΑ	MATLAB CENTER	MATLAB R	FPGA CENTER	FPGA R	ΚΑΤΗΓΟΡΙΑ ΑΠΟΤΕΛΕΣΜΑΤΟΣ
FIGURE 5.13	(340,243)	**38**	(342,244)	**37**	**Καλό**
FIGURE 5.14	(300,263)	28	(302,263)	27	**Καλό**
FIGURE 5.15	(433,152)	46	(435,152)	45	**Καλό**
FIGURE 5.16	(335,157)	29	(334,157)	23	Μέτριο
FIGURE 5.17	(480,184)	**40**	(474,187)	**33**	**Κακό**
FIGURE 5.18	(335,219)	60	(333,220)	-	Κακό
FIGURE 5.19	(338,219)	50	(337,220)	49	**Καλό**
	(341,270)	36	(341,270)	37	**Καλό**
	(322,151)	42	(321,150)	41	**Καλό**
	(421,178)	**46**	(423,179)	**40**	Μέτριο
	(302,210)	34	(301,212)	30	Μέτριο
	(390,193)	38	(392,193)	32	Μέτριο
	(288,176)	27	(288,178)	-	Κακό
	(312,199)	33	(307,194)	29	Κακό
	(428,176)	24	(433,181)	20	Κακό

Figure 5.13: Παράδειγμα καλού αποτελέσματος.

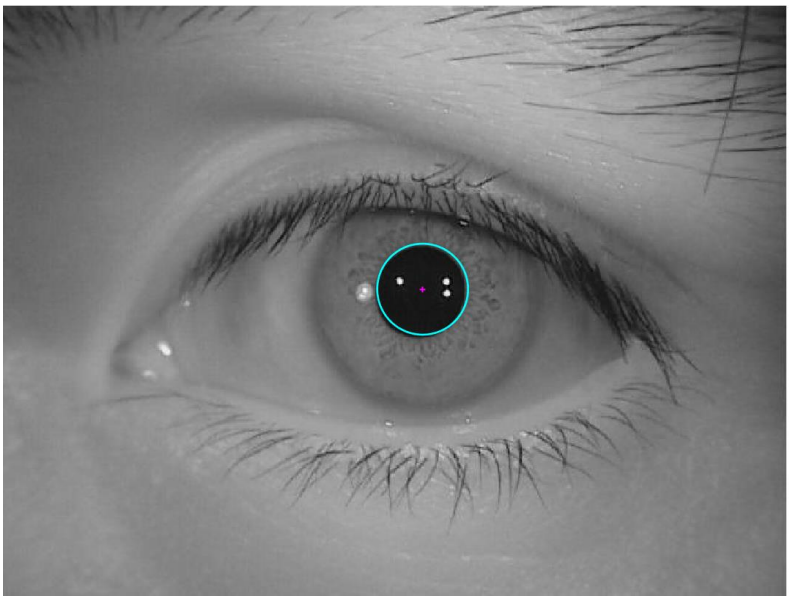


Figure 5.14: Παράδειγμα καλού αποτελέσματος, παρά την ύπαρξη σκοτεινών περιοχών.

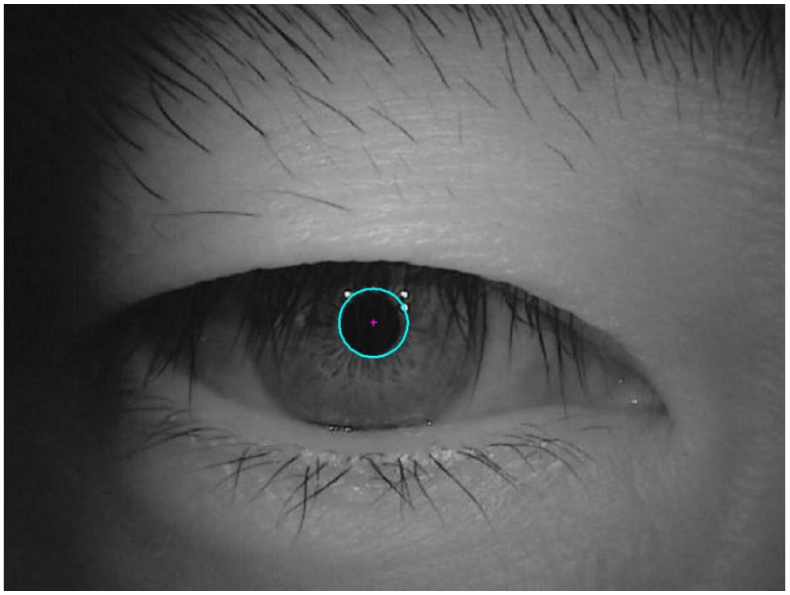


Figure 5.15: Παράδειγμα καλού αποτελέσματος παρά την γωνία του βλέμματος του ματιού.

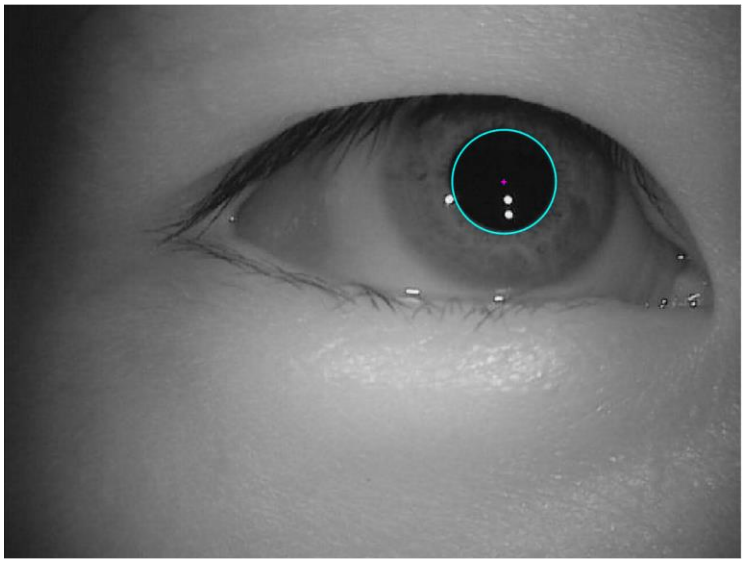


Figure 5.16: Παράδειγμα μέτριου αποτελέσματος εξαιτίας του κοψίματος της κόρης.

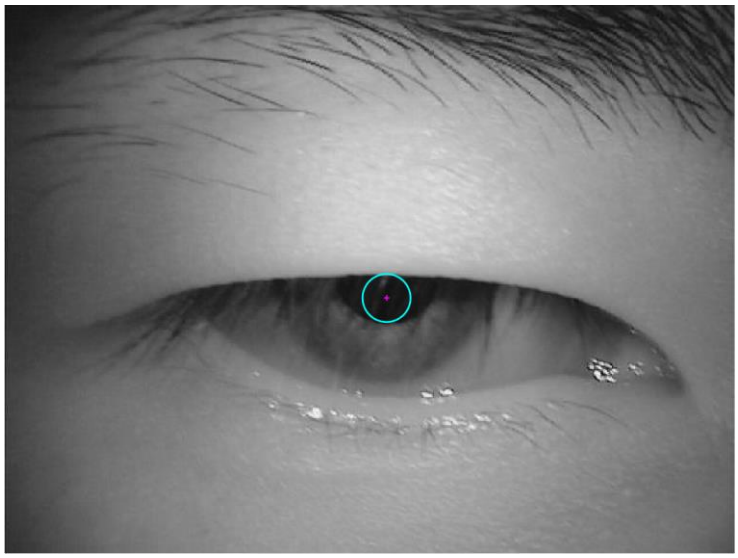
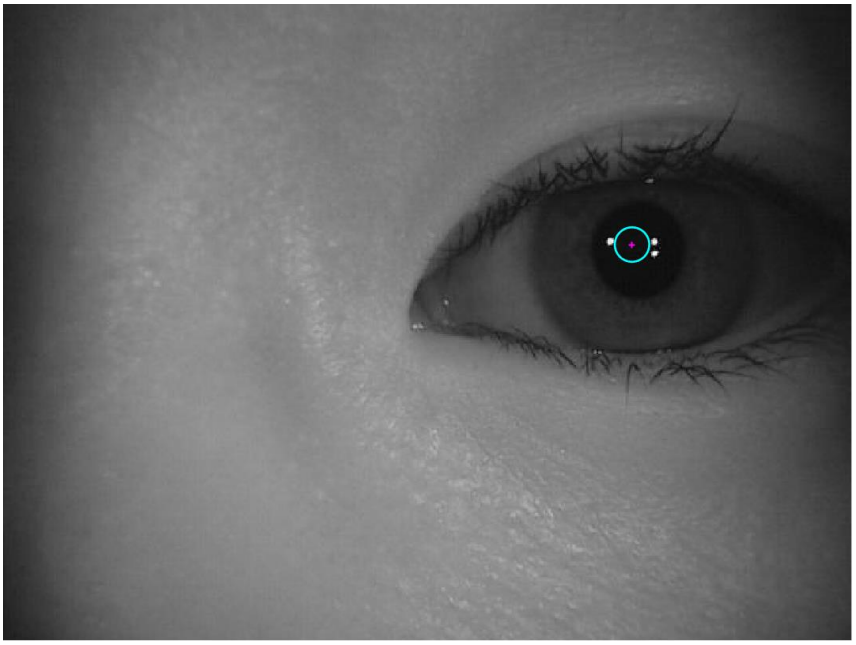


Figure 5.17: Παράδειγμα κακού αποτελέσματος εξαιτίας της σκοτεινής ίριδας και των συμμετρικών αντανακλάσεων ως προς το κέντρο.



Κάτι που πρέπει να επισημανθεί είναι πως η σχεδίαση μας αποτυγχάνει σε δύσκολες περιπτώσεις στον εντοπισμό της ακτίνας παρ' όλα αυτά έχει μεγάλη σημασία η βελτίωση του Hough Transform που έχουμε επιτύχει από άποψη χρόνου. Στην βιβλιογραφία η υλοποίηση του Hough σε Fpga έπαιρνε 5 ms ενώ σε εμάς η συνολική υλοποίηση παίρνε 3.9 ms, εκ των οποίων τα 1.3 ms χρειάζονται για την εγγραφή της εικόνας, τα 1.3 ms για την παραγωγή του χάρτη ακμών και τέλος ο Hough παίρνει μόνο 1.3 ms για εικόνες 256 x 256 (αντίστοιχες με αυτές τις βιβλιογραφίας). Επιπλέον παρατηρήσαμε ότι σε φωτογραφίες ίδιων ατόμων μπορεί να βελτιωθεί η ανίχνευσή μας με καλύτερο φωτισμό, κατά τον οποίον μικραίνει η κόρη και επομένως είναι πιο εύκολο να εντοπιστεί. Ακολουθεί ένα παράδειγμα που εξαιτίας της ρύθμισης του φωτισμού, από κακό αποτέλεσμα μετατράπηκε σε καλό.

Figure 5.18: Παράδειγμα κακού αποτελέσματος εξαιτίας κακού φωτισμού, μεγάλης κόρης που κόβεται και λοιπών παραμέτρων.

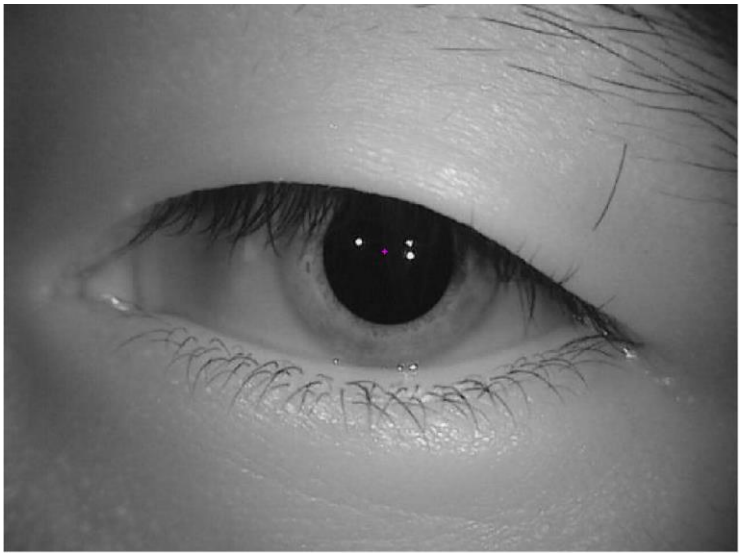
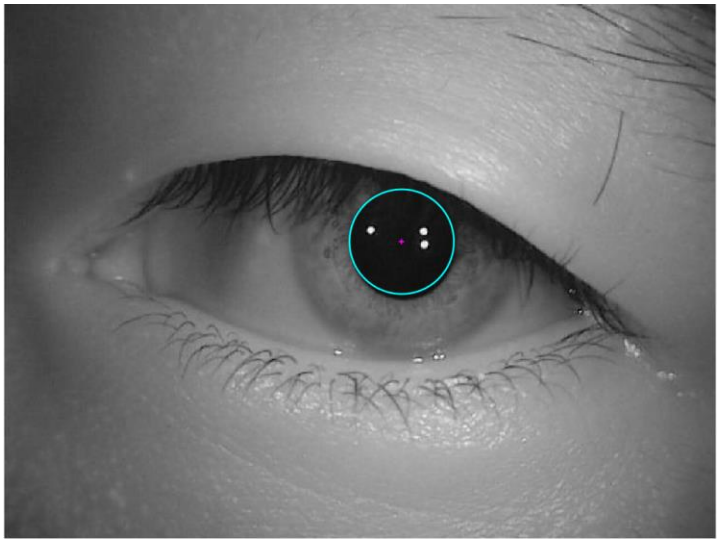


Figure 5.19: Μετατροπή του κακού αποτελέσματος σε καλό με την βελτίωση του φωτισμού και την σμίκρυνση της κόρης εξαιτίας αυτού.



6 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Στόχος της διπλωματικής ήταν η υλοποίηση μίας σχεδίασης hardware που δεν θα περιορίζεται στην εξυπηρέτηση μίας συγκεκριμένης εφαρμογής αλλά θα αντιμετωπίζει το ζήτημα της ανίχνευσης θέσης της κόρης, πιο σφαιρικά και θα καλύπτει μία πληθώρα μελλοντικών επεκτάσεων. Για αυτό τον λόγο εντάξαμε στην ανίχνευσή θέσης, τον εντοπισμό και του κέντρου και της ακτίνας. Αυτό αποκτά μεγάλη σημασία, αν ληφθεί υπόψιν ότι η θέση της κόρης αποτελεί την βάση (και με το μεγαλύτερο κόστος σε πολυπλοκότητα) για όλες τις εφαρμογές ανίχνευσης ίριδας και η πληροφορία για το κέντρο και την ακτίνα της είναι αρκετά σημαντική.

Σημαντικό επίτευγμα ήταν ο μικρός χρόνος που παίρνει η λειτουργία της σχεδίασής μας στην Fpga (3.98 ms), ειδικά αν αναλογιστούμε το σχετικά χαμηλό ρολόι των 50 Mhz, όπως επίσης και η σχετικά χαμηλή κατανάλωση ενέργειας. Με αυτά σαν δεδομένα, η σχεδόν βέβαια ανίχνευση του κέντρου σε συνδυασμό με την δυνατότητα καλύτερης ανίχνευσης της ακτίνας, με προσαρμογές ενός ενσωματωμένου συστήματος σε ζητήματα όπως ο φωτισμός, καθιστούν το έργο της διπλωματικής επιτυχημένο.

Ιδιαίτερη αναφορά πρέπει να γίνει και στην χρησιμοποίηση του Xilinx System Generator, καθώς είναι ένα εργαλείο που μπορεί να βοηθήσει στην υλοποίηση αλγορίθμων, που αφορούν την επεξεργασία εικόνας, με κατεύθυνση το Hardware εξαιτίας των πλεονεκτημάτων του σε τέτοιου είδους εργασίας , όπως εξηγήθηκε αναλυτικότερα στο κεφάλαιο 4. Έτσι η εκμάθησή του και η ενασχόλησή με αυτό ήταν ένα επιπλέον κέρδος της παρούσας διπλωματικής.

Όσον αφορά τις μελλοντικές επεκτάσεις, μπορούμε να τις χωρίσουμε σε δύο κατηγορίες.

Πρώτα από όλα η παρούσα διπλωματική εργασία αφήνει ανοιχτό το ενδεχόμενο, μελλοντικά, να εξειδικευτεί στην υλοποίηση μίας συγκεκριμένης εφαρμογής, έχοντας την παρούσα διπλωματική εργασία σαν βάση και αξιοποιώντας την επιτυχημένη εύρεση του κέντρου και της ακτίνας της κόρης. Για παράδειγμα μία εφαρμογή που θα προσφέρει βοήθεια σε άτομα με ειδικές ανάγκες μέσω της παρακολούθησης του βλέμματός τους θα χρησιμοποιήσει το κέντρο της κόρης, ενώ μία εφαρμογή που αφορά την βιομετρία της ίριδας θα χρησιμοποιήσει την ακτίνα. Σημαντική προσθήκη για τα παραπάνω θα ήταν η επεξεργασία βίντεο αντί για εικόνα και η αξιοποίηση της παρούσας διπλωματικής στην επεξεργασία κάθε frame.

Επιπλέον στόχος της διπλωματικής ήταν η αναζήτηση ενός αλγορίθμου που θα κατεβεί στο Hardware σε ικανοποιητικό βαθμό. Ο ικανοποιητικός βαθμός όμως είναι σχετικός καθώς από την δικιά μας σκοπιά πέτυχαμε στο να βρούμε μία αλγοριθμική ροή που να είναι προσιτή και αξιοποιήσιμη από το Hardware καθώς και να την μεταφέρουμε με έναν αποδοτικό τρόπο. Από εκεί και πέρα υπάρχει μία πληθώρα βελτιώσεων που ένας σχεδιαστής Hardware μπορεί να τις υλοποιήσει και να επιτύχει ακόμα καλύτερη απόδοση.

7 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] T. Kocejko, A. Bujnowski, and J. Wtorek, "Eye mouse for disabled," in *IEEE Conf. Human Syst. Interact*, 2008.

[2] B. Fu, R. Yang, "Display control based on eye gaze estimation," in *4th Int. CISP*, 2011.

[3] R. Lupu, R. Bozomitu, F. Ungureanu, V. Cehan, "Eye tracking based communication system for patient with major neuro-locomotor disabilities," in *IEEE 15th ICSTCC*, 2011.

[4] A. Pranith, C. R. Srikanth, "Iris recognition using corner detection," in *in Proc. 2nd ICISE*, 2010.

[5] R. Sundaram, B. Dhara, B. Chanda, "A fast method for iris Localization," in *Proc. 2nd Int. Conf. EAIT*, 2011.

[6] C. Calvi, M. Porta, D. Sacchi, "E5Learning, an e-learning environment," in *8th IEEE ICALT*, 2008.

[7] R. C. Coetzer, G. P. Hancke, "Eye detection for a real-time vehicle driver fatigue monitoring system," in *IEEE Intell. Veh. Symp.*, 2011.

[8] M. S. Devi, P. R. Bajaj, "Driver fatigue detection based on eye detection," in *1st Int. Conf. Emerg. Trends Eng. Technol*, 2008.

[9] W. Fuhl, T. Santini, T. Kubler, E. Kasneci, "Else: Ellipse selection for robust pupil detection in real-world environments," in *Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, 2016.

[10] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, E. Kasneci, "ExCuSe: Robust pupil detection in real-world scenarios," in *International Conference,Computer Analysis of Images*, 2015.

[11] L. Swirski, A. Bulling, N. Dodgson, "Robust real-time pupil tracking," in *Symposium on Eye Tracking Research & Applications*, 2012.

[12] D. Li, D. Winfield, D., Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and modelbased approaches," in *IEEE Computer Society Conference on CVPR*, 2005.

[13] A. Javadi,Z. Hakimi,M. Barati, V. Walsh, L. Tcheang, "Set: A pupil detection method using sinusoidal approximation," in *Front. Neuroeng*, 2015.

[14] M. Kassner, W.Patera, A. Bulling, "Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014.

[15] W. Fuhl, M. Tonsen, A. Bulling, E. Kasneci, "Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art," *Machine Vision and Applications*, vol. 27, no. 8, pp. 1275-1288, 2016.

[16] V. Raudonis, R. Simutis, G. Narvydas, "Discrete eye tracking for medical applications," in *2nd ISABEL*, 2009.

[17] J. Tang, J. Zhang, "Eye tracking based on grey prediction," in *1st Int. Workshop Educ. Technol. Comput. Sci.*, 2009.

[18] Y. Kuo, J. Lee, S. Kao, "Eye tracking in visible environment," in *5th Int. Conf. IIH-MSP*, 2009.

[19] H. Liu, Q. Liu, "Robust real-time eye detection and tracking for rotated facial images under complex conditions," in *6th ICNC*, 2010.

[20] C. Yang, J. Sun, J. Liu, X. Yang, D. Wang, W. Liu, "A gray difference-based pre-processing for gaze tracking," in *IEEE 10th ICSP*, 2010.

[21] X.Yang, J. Sun, J. Liu, J. Chu,W. Liu, Y. Gao, "A gaze tracking scheme for eye-based intelligent control," in *3rd Int. Conf. IEEE Intell. Inf. Hiding Multi-media Signal Process*, 2007.

[22] Y. Chen, K. Kubo, "A robust eye detection and tracking technique using gabor flters," in *3rd Int. Conf. IEEE Intell. Inf. Hiding Multi-media Signal Process*, 2007.

[23] W. Khairosfaizal, A. Nor'aini, "Eye detection in facial images using circular Hough transform," in *5th CSPA*, 2009.

[24] A. Spurou, "Android based embedded iris data processing system," Thesis, Technical Univercity of Crete, 2016.

[25] C. Siachamis, "Study of lighting methods of the eye iris and embedded Android application for iris identification," Thesis,Technical Univercity of Crete, 2018.

[26] M. López, J. Daugman, E. Cantó, "Hardware–software co-design of an iris recognition algorithm," in *IET Inf. Secur*, 2011.

[27] H. Ngo, R. Rakvic, R. Broussard, R. Ives, "Resource-aware architecture design and implementation of Hough transform for a real-time iris boundary detection," in *IEEE Trans. Consum. Electron*, 2014.

[28] A. Elhossini, M. Moussa, "A memory efficient FPGA implementation of Hough transform for line and circle detection," in *25th Canadian*, 2012.

[29] Z.-H. Chen, A.W.Y. Su, M.-T. Sun, "Resource-efficient FPGA architecture and implementation of Hough transform," in *IEEE Trans. Very Large Scale Integr.*, 2012.

[30] V. Kummar, "Iris Localization in Iris Recognition System: Algorithms and Hardware Implementation," Birla Institute of technology & science Pilani, 2016.

[31] J. Avey, "An FPGA-based hardware accelerator for iris segmentation," Thesis, Iowa State University, 2018.

[32] K. Anil, M. Vijay, "Implementation of Image Processing Lab Using Xilinx System Generator," *AIVP*, vol. 2, no. 5, 2015.

[33] S. Mohapatra, B, Ranjan, "Optimized Approach of Sobel Edge Detection Technique Using Xilinx System Generator," in *ICECS*, 2015.

[34] H, Veena, G.Kamala, "Different Edge Detection Techniques using System Generator," *IJAREEIE*, vol. 5, no. 5, 2016.

[35]