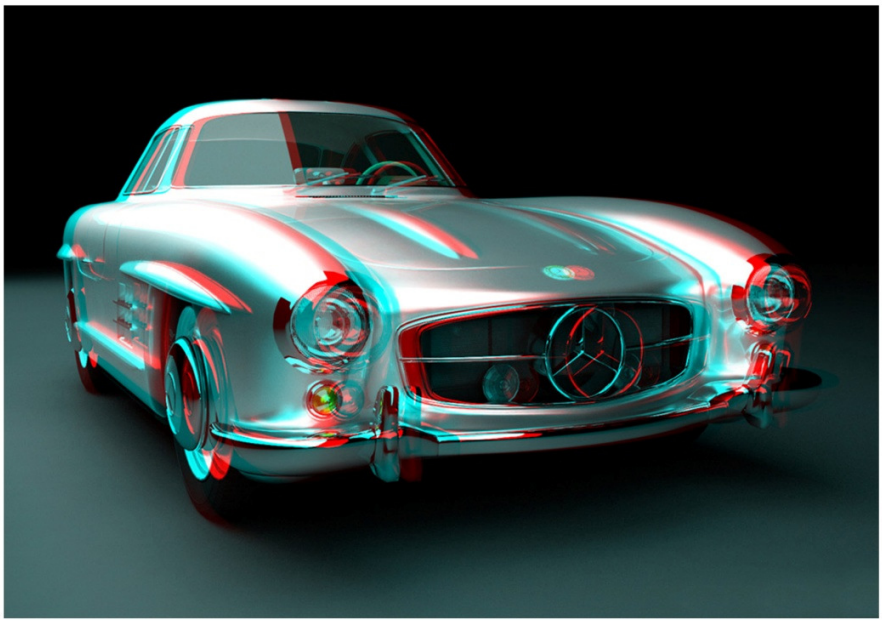
Τμήμα
Μηχανικών
Πληροφορικής τ.ε.
Τεχνολογικό Εκπαιδευτικό Ίδρυμα
Δυτικής Ελλάδας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και υλοποίηση σε υλικό, αλγορίθμου για καταγραφή και ανάλυση εικόνας

Διαμαντόπουλος Νικόλαος
Α.Μ. : 1181



Αντίριο, Ιανουάριος 2017

Περιεχόμενα

Πρόλογος

Η παρακάτω ερευνητική εργασία , βασίζεται στη μελέτη της στερεοσκοπικής όρασης . Ξεκινώντας από την ανάλυση της , έννοιες της και γυρνώντας πίσω στο χρόνο για μια ιστορική αναδρομή , προχωράμε για να συναντήσουμε έννοιες όπως ¨ μονόφθαλμη όραση ¨ και ¨ τρισδιάστατη αντίληψη στο χώρο ¨. Καθώς εξετάζουμε τη στερεοσκοπική όραση μέσα από κώδικες , τους ενσωματώνουμε στο εργαλείο Matlab, επεξεργάζοντας τη χρήση, τη λειτουργία και το αποτέλεσμα που μας αποφέρει. Εν συνεχεία επιλέγουμε τον βέλτιστο κώδικα , τον μετατρέψαμε σε γλώσσα περιγραφής υλικού VHDL έτσι ώστε να είναι εφικτή η εφαρμογή του , σε συστήματα FPGAs.

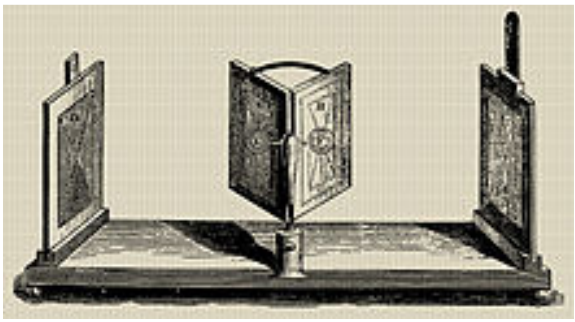
Κεφάλαιο 1ο

Στερεοσκοπική όραση

1.1 Ορισμός του στερεογράμματος

Ένα αυτοστερεόγραμμα είναι ένα στερεόγραμμα μιας εικόνας, στα αγγλικά single-image stereogram (SIS), το οποίο είναι σχεδιασμένο να «ξεγελά» τον ανθρώπινο εγκέφαλο έτσι ώστε να αντιλαμβάνεται μία τρισδιάστατη εικόνα (3D) σε μία δισδιάστατη απεικόνιση. Για να γίνουν αντιληπτά τα τρισδιάστατα σχήματα, ο εγκέφαλος πρέπει να παρακάμψει την κανονικά αυτόματη λειτουργία της εστίασης και της ταυτόχρονης κίνησης και των δύο οφθαλμών.

Το 1838, ο Βρετανός επιστήμονας, Charles Wheatstone εξέδωσε μία ερμηνεία της διόφθαλμης όρασης που σχετιζόταν με την αντίληψη του βάθους από τους ανθρώπους. Στην εργασία του αυτή, ο Wheatstone έφτιαξε στερεοσκοπικές εικόνες και δημιούργησε ένα στερεοσκόπιο βασισμένο σε έναν συνδυασμό καθρεφτών, ώστε να μπορεί κάποιος να δει τρισδιάστατες εικόνες από δισδιάστατα σχήματα.



Εικόνα 1.1: Το στερεοσκόπιο που κατασκεύασε ο Charles Wheatstone.

Μεταξύ του 1849 και του 1850, ο Sir David Brewster, Σκώτος επιστήμονας, βελτίωσε το στερεοσκόπιο του Wheatstone χρησιμοποιώντας φακούς αντί για καθρέφτες, μειώνοντας έτσι το μέγεθος της συσκευής. Ο Brewster παρατήρησε ότι το επίμονο κοίταγμα επαναλαμβανόμενων σχεδίων σε ταπετσαρίες, μπορούσε να μπερδέψει το μάτι, το οποίο συνδύαζε ζεύγη που ταίριαζαν μεταξύ τους, δημιουργώντας έτσι την ψευδή αντίληψη ενός

εικονικού επιπέδου πίσω από τον τοίχο. Αυτή είναι και η βάση των στερεογραμμάτων τοιχοστρωσίας (τα οποία είναι γνωστά και ως στερεογράμματα μιας εικόνας).

1.2 Ιστορική Αναδρομή

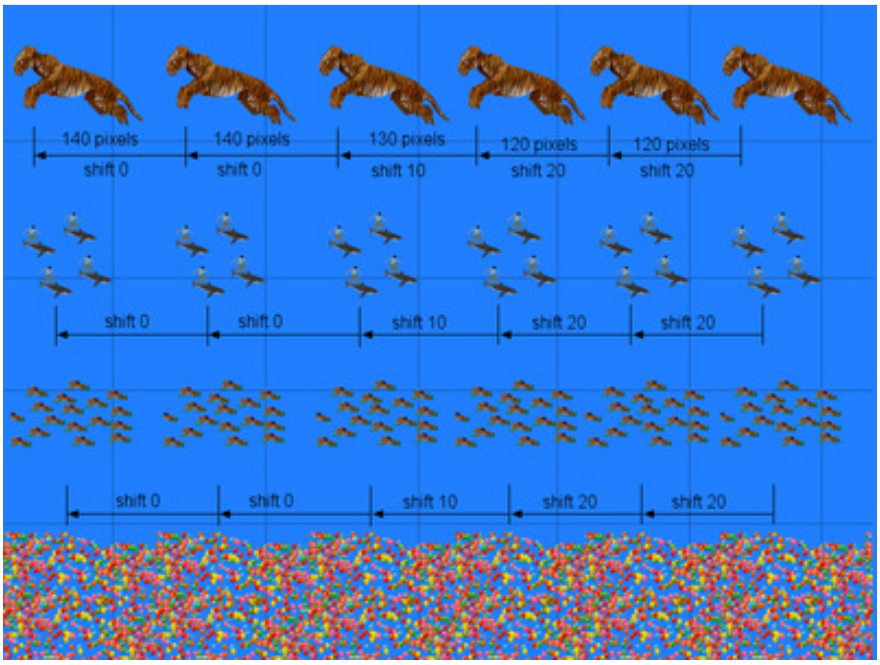
Το 1959, ο Béla Julesz, ένας επιστήμονας της όρασης, ψυχολόγος και υπότροφος του Ιδρύματος ΜακΆρθουρ, ανακάλυψε το στερεόγραμμα τυχαίας κουκκίδας καθώς εργαζόταν στα εργαστήρια Μπελ, πάνω στην αναγνώριση καμουφλαρισμένων αντικειμένων από αεροφωτογραφίες που τραβήχτηκαν από κατασκοπευτικά αεροπλάνα. Εκείνη την περίοδο, οι επιστήμονες της όρασης, πίστευαν ακόμη πως η αντίληψη του βάθους συνέβαινε στο ίδιο το μάτι, ενώ σήμερα είναι γνωστό πως είναι μια πολύπλοκη νευρολογική διεργασία. Ο Julesz χρησιμοποίησε έναν υπολογιστή για να δημιουργήσει στερεοσκοπικά ζεύγη εικόνων τυχαίας κουκκίδας τα οποία όταν κοιτάζονταν υπό στερεοσκοπίου, προκαλούσαν τον εγκέφαλο να απεικονίσει τρισδιάστατα σχήματα. Αυτό απέδειξε ότι η αντίληψη βάθους είναι νευρολογική διεργασία.

Το 1979, ο Κρίστοφερ Τάιλερ του Ινστιτούτου Σμιθ-Κέτλγουελ (Smith-Kettlewell Institute), μαθητής του Julesz και οπτικός ψυχοφυσικός συνδύασε τις θεωρίες των στερεογραμμάτων μιας εικόνας (τοιχοστρωσίας) και των στερεογραμμάτων τυχαίας κουκίδας και δημιούργησε το πρώτο αυτοστερεόγραμμα τυχαίας κουκκίδας (επίσης γνωστό ως μιας εικόνας, τυχαίας κουκκίδας στερεόγραμμα) το οποίο επέτρεπε στον εγκέφαλο να αντιληφθεί μία τρισδιάστατη εικόνα από δισδιάστατη απεικόνιση, χωρίς να χρειάζεται κάποιο οπτικό βοήθημα

1.3 Η στερεοσκοπική όραση

Η στερεοσκοπία ή στερεοσκοπική όραση είναι η οπτική ανάμιξη δύο πανομοιότυπων αλλά όχι ίδιων εικόνων σε μία, με αποτέλεσμα την οπτική αντίληψη της στερεότητας και του βάθους.[8] Στον ανθρώπινο εγκέφαλο, η στερεοσκοπία προκύπτει από ένα πολύπλοκο σύνολο μηχανισμών που σχηματίζουν μία τρισδιάστατη αντίληψη μέσω της συσχέτισης κάθε σημείου (ή συνόλου σημείων) στο ένα μάτι κάποιου, με ένα αντίστοιχο σημείο (ή σύνολο σημείων) στο άλλο μάτι. Έτσι λοιπόν, προσδιορίζονται οι θέσεις των σημείων στον ανέκφραστο οπτικά, άξονα z (βάθους).

Όταν ο εγκέφαλος αντιμετωπίζει μια εικόνα ενός επαναλαμβανόμενου σχεδίου, όπως μιας τοιχοστρωσίας (ταπετσαρίας), αντιμετωπίζει δυσκολίες ως προς την ακριβή θέση υπό την οποία το κάθε μάτι θα προσαρμόζεται στην εικόνα. Κοιτώντας ένα οριζοντίως επαναλαμβανόμενο σχέδιο, αλλά συγκλίνοντας τα δύο μάτια σε ένα νοητό σημείο πίσω από το σχέδιο, είναι πιθανό να "ξεγελάσουμε" τον εγκέφαλο και έτσι να συνδυάσει ένα στοιχείο όπως αυτό φαίνεται από το αριστερό μάτι με ένα άλλο (παρόμοιο οπτικά), δίπλα από το πρώτο, όπως φαίνεται από το δεύτερο μάτι. Αυτό δίνει την ψευδαίσθηση ενός επιπέδου που περιλαμβάνει το ίδιο σχέδιο αλλά φαίνεται να βρίσκεται πίσω από τον πραγματικό τοίχο. Η απόσταση στην οποία το επίπεδο φαίνεται πίσω από τον τοίχο εξαρτάται μόνο από τον χώρο μεταξύ των ίδιων στοιχείων.



Εικόνα 1.2: Τα σχέδια σε αυτό το αυτοστερεόγραμμα θα εμφανίζονται σε διαφορετικό βάθος κατά μήκος κάθε σειράς.

Τα αυτοστερεογράμματα χρησιμοποιούν αυτήν την εξάρτηση του βάθους με την απόσταση παρομοίων σχεδίων, ώστε να δημιουργούν τρισδιάστατες εικόνες. Αν πάνω από κάποια περιοχή της εικόνας το σχέδιο επαναλαμβάνεται ανά μικρότερες αποστάσεις, η περιοχή αυτή θα εμφανιστεί πιο κοντά από το επίπεδο του υποβάθρου. Αν η απόσταση των επαναλήψεων είναι μεγαλύτερη σε κάποια περιοχή, τότε αυτή η περιοχή θα εμφανίζεται πιο μακριά (σαν τρύπα στο επίπεδο).

Ο άνθρωποι που δεν μπόρεσαν ποτέ να δουν τις τρισδιάστατες εικόνες που κρύβονται πίσω από ένα αυτοστερεόγραμμα θεωρούν δύσκολη την κατανόηση προτάσεων όπως: "η τρισδιάστατη εικόνα θα εμφανιστεί από το υπόβαθρο αν την κοιτάξετε για αρκετή ώρα" ή "η τρισδιάστατη εικόνα θα αναδυθεί από το υπόβαθρο". Επομένως θεωρείται ότι θα βοηθούσε κάποια απεικόνιση του πως οι τρισδιάστατες εικόνες "αναδύονται" από το υπόβαθρο από την οπτική γωνία ενός ανεξάρτητου παρατηρητή. Αν τα εικονικά τρισδιάστατα αντικείμενα ενός στερεογράμματος, που ανασυντίθενται στον εγκέφαλο κάποιου παρατηρητή, ήταν αληθινά αντικείμενα, ένας ανεξάρτητος θεατής που θα παρατηρούσε τη σκηνή από δίπλα θα έβλεπε τα αντικείμενα αυτά να αιωρούνται πάνω από την εικόνα του υποβάθρου.

Τα τρισδιάστατα εφέ στο αυτοστερεόγραμμα του παραδείγματος δημιουργούνται από την επανάληψη της εικόνας του καβαλάρη της τίγρης επαναλαμβάνεται κάθε 140 εικονοστοιχεία, του καβαλάρη του καρχαρία κάθε 130 εικονοστοιχεία και της εικόνας της τίγρης κάθε 120 εικονοστοιχεία. Όσο πιο κοντά μεταξύ τους, σε οριζόντια απόσταση, είναι ένα σύνολο εικόνων, τόσο πιο ψηλά φαίνονται σε σχέση με το επίπεδο του υποβάθρου. Αυτή η απόσταση επανάληψης αναφέρεται ως το βάθος ή η τιμή του άξονα των z. Στην αγγλόφωνη ορολογία και στην ορολογία των γραφικών υπολογιστή αυτή η τιμή είναι γνωστή ως τιμή z-buffer.

Ο εγκέφαλος είναι ικανός στο να ταυτίζει εκατοντάδες επαναλαμβανόμενα σχέδια σε διαφορετικές αποστάσεις μεταξύ τους, έτσι ώστε να επαναδημιουργεί τη σωστή πληροφορία βάθους για κάθε σχέδιο. Ένα αυτοστερεόγραμμα μπορεί να περιλαμβάνει 50 τίγρεις διαφόρων μεγεθών, που επαναλαμβάνονται ανά διαφορετικά διαστήματα, σε ένα περίπλοκο επαναλαμβανόμενο υπόβαθρο. Εντούτοις, παρά τη φαινομενικά χαοτική χωροθέτηση των σχεδίων, ο εγκέφαλος είναι ικανός να θέσει κάθε εικόνα τίγρης στο κατάλληλο βάθος.

1.4 Μονόφθαλμη όραση

Στη μονόφθαλμη, στα αγγλικά Monocular vision από το ελληνικό μονός, όραση τα δύο μάτια χρησιμοποιούνται ξεχωριστά. Χρησιμοποιώντας τα μάτια έτσι με αυτόν τον τρόπο, αντίθετα με τη διόφθαλμη όραση, ο χώρος έκτασης που μπορούμε να δούμε αυξάνεται, ενώ το βάθος μειώνεται. Με αυτόν τον τρόπο λειτουργεί η όραση πολλών ζώων και έτσι μπορούν να αντιληφθούν δύο αντικείμενα ταυτόχρονα.

1.5 Αντίληψη του τρισδιάστατου

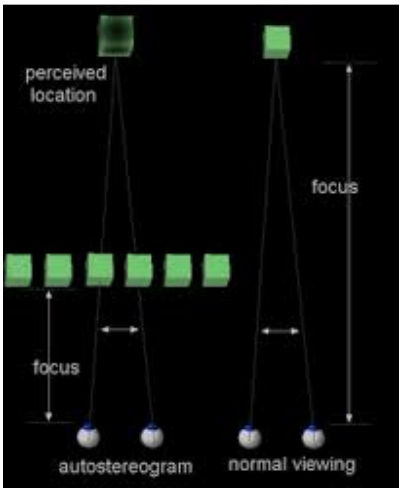
Η αντίληψη του βάθους προκύπτει από πολλά μονοφθάλμια και διοφθάλμια στοιχεία. Για τα αντικείμενα που είναι σχετικά κοντά στα μάτια, η διοφθάλμια όραση είναι αυτή που παίζει σημαντικό ρόλο στην αντίληψη του βάθους. Επίσης, αυτή είναι που επιτρέπει στον εγκέφαλο να δημιουργήσει μια κυκλώπεια εικόνα και να συσχετίσει ένα επίπεδο βάθους σε κάθε σημείο σε αυτήν.

Ο εγκέφαλος χρησιμοποιεί τη μετατόπιση συντεταγμένων (γνωστή και ως παράλλαξη) των σημείων που σχετίζονται οπτικά μεταξύ τους, ώστε να αναγνωρίσει το βάθος αυτών των σημείων. Το επίπεδο βάθους στη συνδυασμένη εικόνα, μπορεί να αντιπροσωπεύεται από ένα εικονοστοιχείο στην κλίμακα του γκρι για τη δισδιάστατη εικόνα, προς όφελος του αναγνώστη. Όσο πιο κοντά εμφανίζεται ένα σημείο στον εγκέφαλο, τόσο πιο φωτεινό απεικονίζεται στο σχήμα. Έτσι, ο τρόπος σύμφωνα με τον οποίο ο εγκέφαλος αντιλαμβάνεται το βάθος με τη διοφθάλμια όραση, μπορεί να απεικονιστεί από μια κλίμακα βάθους που χρωματίζεται ανάλογα με την μετατόπιση των συντεταγμένων.

Το μάτι προσαρμόζει τον εσωτερικό του φακό ώστε να λάβει μία σαφή, εστιασμένη εικόνα. Τα δύο μάτια συγκλίνουν προς ένα αντικείμενο. Το ανθρώπινο μάτι λειτουργεί όπως μία φωτογραφική μηχανή. Έχει μια ρυθμιζόμενη ίριδα, που ανοίγει (ή κλείνει) για να επιτρέψει περισσότερο (ή λιγότερο) φως να εισέλθει στο μάτι. Όπως σχεδόν κάθε φωτογραφική μηχανή, χρειάζεται να εστιάσει τις ακτίνες του φωτός που εισέρχονται μέσω της ίριδας, ώστε η εστίαση να γίνεται σε ένα σημείο του αμφιβληστροειδούς για να παραχθεί μια ευκρινής εικόνα. Το μάτι το πετυχαίνει αυτό, προσαρμόζοντας έναν φακό πίσω από τον κερατοειδή ώστε να διαθλά το φως σωστά για αυτή τη λειτουργία.

Όταν κάποιος κοιτά ένα αντικείμενο, οι δύο κόρες περιστρέφονται δείχνοντας προς αυτό, ώστε το αντικείμενο να εμφανίζεται στο κέντρο της εικόνας που δημιουργείται από τον αμφιβληστροειδή χιτώνα. Όταν κάποιο αντικείμενο βρίσκεται κοντά, γίνεται σύγκλιση των ματιών προς το αντικείμενο. Για να δούμε κάποιο μακρινό αντικείμενο τα δύο μάτια αποκλίνουν το ένα από το άλλο έως ότου σχεδόν παραλληλίζονται οι ακτίνες κατά τις οποίες η κάθε κόρη κοιτά.

Η στερεοσκοπική όραση βασίζεται στην παράλλαξη που επιτρέπει στον εγκέφαλο να υπολογίζει τα βάθη των αντικειμένων σε σχέση με το σημείο σύγκλισης. Η γωνία σύγκλισης είναι αυτή που δίνει στον εγκέφαλο την απόλυτη τιμή του βάθους αναφοράς για το σημείο σύγκλισης. Από αυτό υπολογίζονται όλα τα απόλυτα βάθη για τα υπόλοιπα σημεία που εκφράζουν την εικόνα των αντικειμένων.



Εικόνα 1.3: Αντίληψη του τρισδιάστατου από τον ανθρώπινο οφθαλμό.

Κεφάλαιο $2^{ο}$:

Αλγόριθμοι στερεοσκοπικής όρασης

2.1 Παρουσίαση του εργαλείου Matlab

Η MATLAB [1](matrix laboratory) είναι ένα περιβάλλον αριθμητικών υπολογισμών και μίας τέταρτης γενιάς γλώσσα προγραμματισμού η οποία αναπτύχθηκε από την εταιρεία MathWorks. Η Matlab επιτρέπει πολλαπλασιασμό πινάκων, δημιουργία γραφημάτων από συναρτήσεις ή δεδομένα, υλοποίηση αλγορίθμων, δημιουργία διεπαφών χρήστη και διασύνδεση με άλλα προγράμματα τα οποία είναι γραμμένα σε άλλες γλώσσες όπως C, C++, Java, Fortran και Python.

Μετά το 2004 το εργαλείο αυτό βρήκε εφαρμογή σε πολλά εκατομμύρια χρήστες στη βιοτεχνία και τα ακαδημαϊκά ιδρύματα. Στους χρήστες του εργαλείου αυτού εμπεριέχονται μηχανικοί, επιστήμονες και οικονομολόγοι.

Στη συνέχεια παρουσιάζουμε δύο βασικές εργαλειοθήκες τα οποία θα χρειαστούμε χρησιμοποιώντας τη Matlab, το Signal Processing Toolbox και το Computer Vision System Toolbox.[2]

Εργαλειοθήκες

Το Signal Processing μας δίνει τη δυνατότητα να χρησιμοποιούμε συναρτήσεις και εφαρμογές για δημιουργία, μέτρηση, μετασχηματισμό, φιλτράρισμα και οπτικοποίηση σημάτων. Η εργαλειοθήκη αυτή περιλαμβάνει αλγορίθμους για τη δειγματοληψία, την εξομάλυνση, των συγχρονισμό σημάτων, καθώς επίσης το σχεδιασμό και την ανάλυση φίλτρων. Επίσης για τον υπολογισμό του φάσματος, του εύρους και της παραμόρφωσης. Η εργαλειοθήκη περιέχει ακόμη παραμετρικά και γραμμικά μοντέλα πρόβλεψης αλγορίθμων.

Μια επίσης πολύ χρήσιμη εργαλειοθήκη είναι η Computer Vision System, η οποία μας παρέχει αλγορίθμους, εφαρμογές και συναρτήσεις για το σχεδιασμό και την προσομοίωση συστημάτων υπολογιστικής όρασης και επεξεργασίας βίντεο. Μπορούμε να εφαρμόσουμε ανίχνευση αντικειμένων, ανίχνευση και εξόρυξη υφής, συνδυασμό υφής, stereo απεικόνιση,

ανίχνευση κίνησης κ.α. Τέλος, η εργαλειοθήκη αυτή παρέχει επιπλέον δυνατότητες για την επεξεργασία και απεικόνιση βίντεο οι οποίες δεν θα μας απασχολήσουν στα πλαίσια αυτής της εργασίας.[3], [4]

2.1 Παρουσίαση των αλγορίθμων

Στα πλαίσια αυτής της εργασίας θα χρησιμοποιήσουμε δύο αλγορίθμους με τους οποίους μπορούμε να επεξεργαστούμε ζεύγη εικόνων και να κάνουμε στερεοσκοπική απεικόνιση των περιεχομένων τους. Τα ζεύγη των εικόνων που θα χρησιμοποιήσουμε θα είναι στο ίδιο χώρο με ολίσθηση της φωτογραφικής μηχανής κατά 6 εκατοστά προς τα αριστερά ώστε να αποτυπώσουμε την ίδια φυσική σκηνή με κάποιες επιπλέον πληροφορίες.

Η λογική που ακολουθείται κατά την ταξινόμηση των στέρεο αλγορίθμων έχουν ως βάση τα παρακάτω βήματα:

1. Ανάγνωση σετ εικόνων.
2. Συνδυασμός των μπλοκ.
3. Εκτίμηση των υπο-pixel.
4. Δυναμικός προγραμματισμός.
5. Χρήση μεθόδου πυραμίδας για τις εικόνες.
6. Συνδυασμός μεθόδου πυραμίδας και δυναμικού προγραμματιμού.
7. Προβολή του αποτελέσματος.

Στη συνέχεια παρουσιάζουμε τους αλγορίθμους που χρησιμοποιήσαμε.

Αλγόριθμος stereomatch

Σε αυτόν τον αλγόριθμο θα χρησιμοποιήσουμε δομημένο φωτισμό, έτσι ώστε κάθε εικονοστοιχείο να παίρνει μια μοναδική ταμπέλα σε κάθε σετ εικόνων. Με αυτόν τον τρόπο η ακρίβεια κάθε pixel μπορεί αυτόματα να παραχθεί με μεγάλη ακρίβεια. Σκοπός αυτού του αλγορίθμου, είναι χρησιμοποιώντας σετ πολύπλοκων εικόνων, οι οποίες περιέχουν αντικείμενα, να παράγει μια τιμή για την στέρεο-εικόνα, η οποία θα αντιστοιχεί στο αρχικό σετ. [3]

Για να μπορέσουμε να δώσουμε την επιθυμητή τιμή σε κάθε εικονοστοιχείο μπορούμε να προβάλουμε μια εικόνες, δομημένου φωτός, και να αποκωδικοποιήσουμε την ένταση κάθε

εικονοστοιχείου. Σε αυτόν τον αλγόριθμο όμως διαλέξαμε μια διαφορετική μέθοδο δομημένου φωτισμού, τη μέθοδο δυαδικού Gray-code και.

Μέθοδος Gray-code

Οι μέθοδοι Gray-code περιέχουν μόνο άσπρες και μαύρες τιμές εικονοστοιχείων, ακολουθώντας τη λογική on/off, οι οποίες τιμές ήταν και οι μόνες διαθέσιμες στους προηγούμενους LCD προβολείς. Χρησιμοποιώντας τέτοιου τύπου δυαδικές εικόνες απαιτεί $log_2(n)$ πράξεις για τη διάκριση μεταξύ n περιοχών. Η μέθοδος αυτή είναι αρκετά αποδοτική για τέτοιου είδους δυαδικές κωδικοποιήσεις γιατί μειώνει, σε μεγάλο βαθμό, τα λάθη που μπορούν να προκύψουν.

Αποκωδικοποιώντας το φωτισμό είναι σχετικά απλό να αποφασίσουμε για κάθε εικονοστοιχείο αν έχει τιμή μηδέν ή ένα. Βέβαια παρατηρούμε ότι ο μόνος τρόπος για να αποφασίσουμε με ασφάλεια για την κατωφλιωποίηση του εκάστοτε εικονοστοιχείου είναι να προβάλουμε και τις δύο κωδικοποιήσεις και στη συνέχεια να αποφασίσουμε με βάση τη φωτεινότητα. Βέβαια , το προφανές πρόβλημα παρατηρείται στο γεγονός του ότι παράγουμε διπλάσιο αριθμό εικόνων από αυτόν που χρειαζόμαστε με αποτέλεσμα την χρήση μεγαλύτερου τμήματος μνήμης, για το θέμα της μνήμης θα αναφερθούμε σε επόμενο κεφάλαιο. [4]

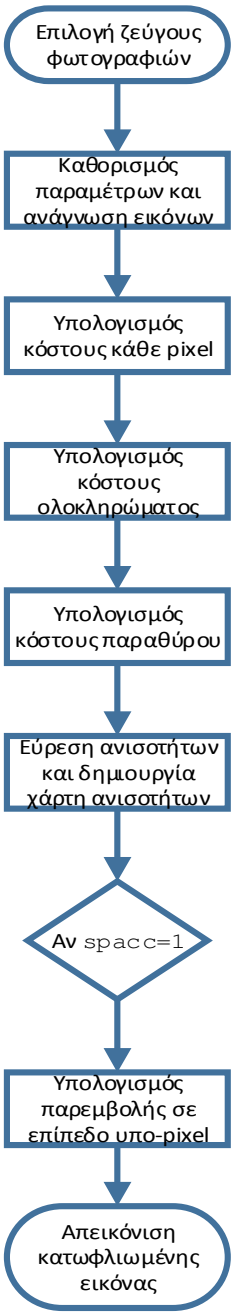
Εναλλακτικός αλγόριθμος stereo – vision.

Ο αλγόριθμος αυτός παράχθηκε κατά την προσπάθεια του δημιουργού του να πλοηγήσει ένα αυτόνομο ελικόπτερο με μία κάμερα. Η πλοήγηση θα χρησιμοποιούσε μονοδιάστατη όραση. Η κάμερα μπορεί να συλλάβει τον κόσμο στις δύο διαστάσεις, αλλά για την επιτυχή πλοήγηση απαιτείται και απόσταση μεταξύ του ελικοπτέρου και τον αντικειμένων. Για την μεταφορά λοιπόν της δισδιάστατης εικόνας στον τρισδιάστατο χώρο πρέπει να συνδυαστεί η στερεοσκοπική όραση με την μονόφθαλμη.

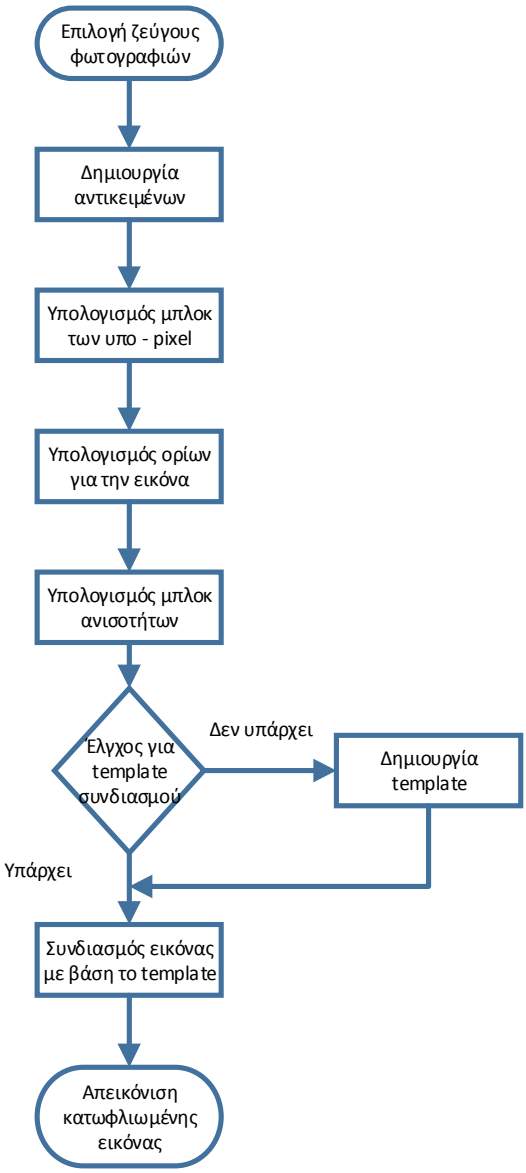
Την ίδια Gray-code κωδικοποίηση, όπως και παραπάνω, παράγει και αυτός ο αλγόριθμος, η εικόνα αποθηκεύεται στο αρχείο Dbasic. Ο αλγόριθμος αυτός παράγει πρακτικά ένα χάρτη, ο οποίος παρουσιάζει στο σύστημα RGB την απόσταση μεταξύ των αντικειμένων χρησιμοποιώντας την Gray-code εικόνα που προκύπτει.

Διαγράμματα Ροής

Στη συνέχεια παρουσιάζουμε τα διαγράμματα ροής των δύο αλγορίθμων:[5]



Σχέδιο 2.1: Flowchart για τον αλγόριθμο stereomatch.



Σχέδιο 2.2: Flowchart για τον εναλλακτικό αλγόριθμο stereo – vision.

2.2 Αποτελέσματα του αλγορίθμων

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τα αποτελέσματα των αλγορίθμων με βάση τα ζεύγη εικόνων που χρησιμοποιήσαμε.

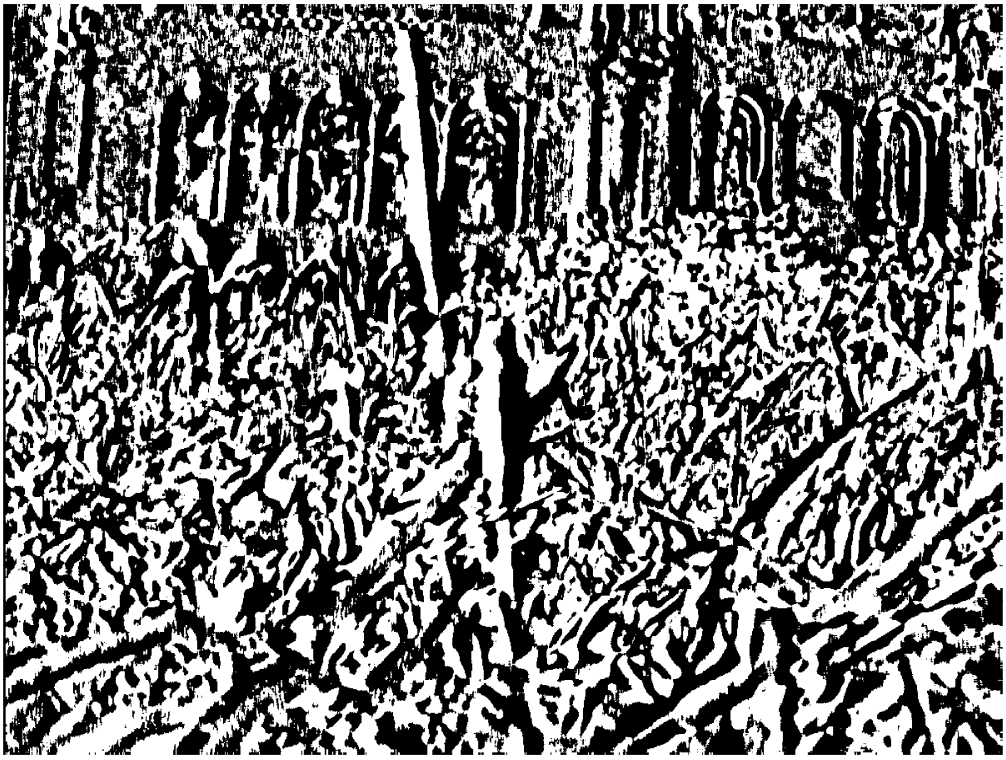
Αλγόριθμος Stereomatch

Για το **πρώτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.1: Η εικόνα Left.

Εικόνα 2.2: Η εικόνα Right.

Η εικόνα 3 παρουσιάζει την αριστερή λήψη της φυσικής σκηνής. Απεικονίσαμε σκηνές με πολλά αντικείμενα ώστε να ελέγξουμε την ορθότητα των αποτελεσμάτων. Στην εικόνα 4 παρουσιάζεται η δεξιά λήψη. Η απόσταση μεταξύ των δύο λήψεων είναι 6 εκατοστά. Στην εικόνα 4 παρουσιάζεται η Gray-code εικόνα που δημιουργείται από τον αλγόριθμο. Το αποτέλεσμα που παίρνουμε μετά τη χρήση του steromatch αλγορίθμου είναι:



Εικόνα 2.3: Gray-code απεικόνιση για το πρώτο σετ εικόνων.

Για το **δεύτερο ζεύγος** εικόνων έχουμε:

Εικόνα 2.4: Η εικόνα Left.

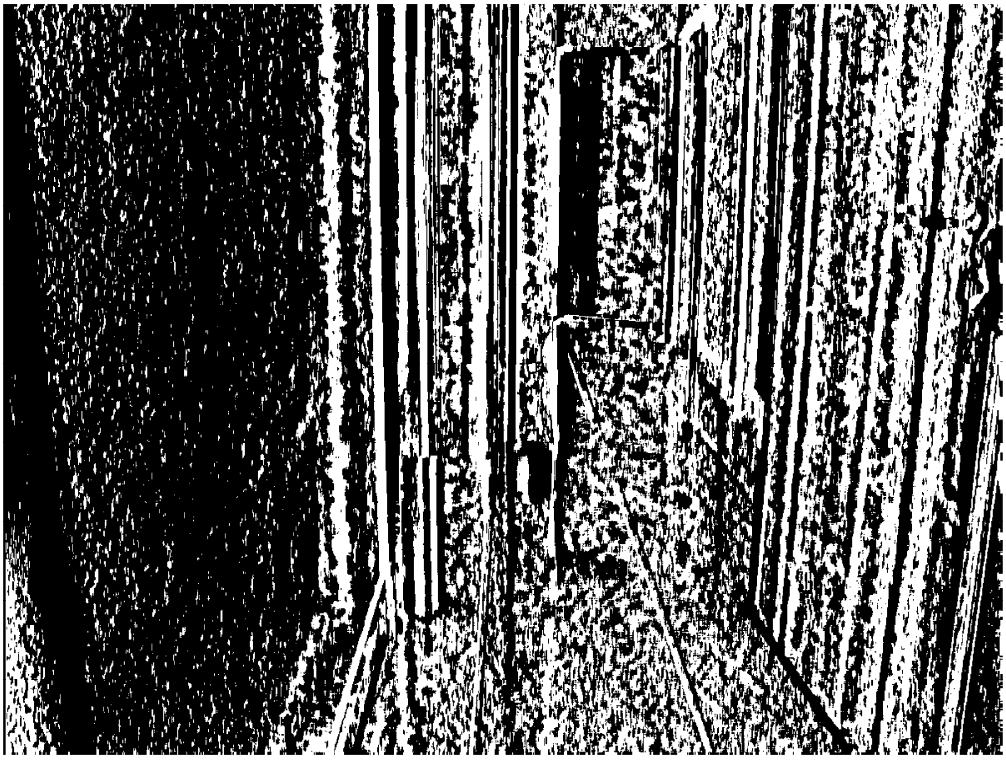
Εικόνα 2.5: Η εικόνα Right.

Εικόνα 2.6: Gray-code απεικόνιση για το δεύτερο σετ εικόνων.

Για το **τρίτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.7: Η εικόνα Left.

Εικόνα 2.8: Η εικόνα Right.

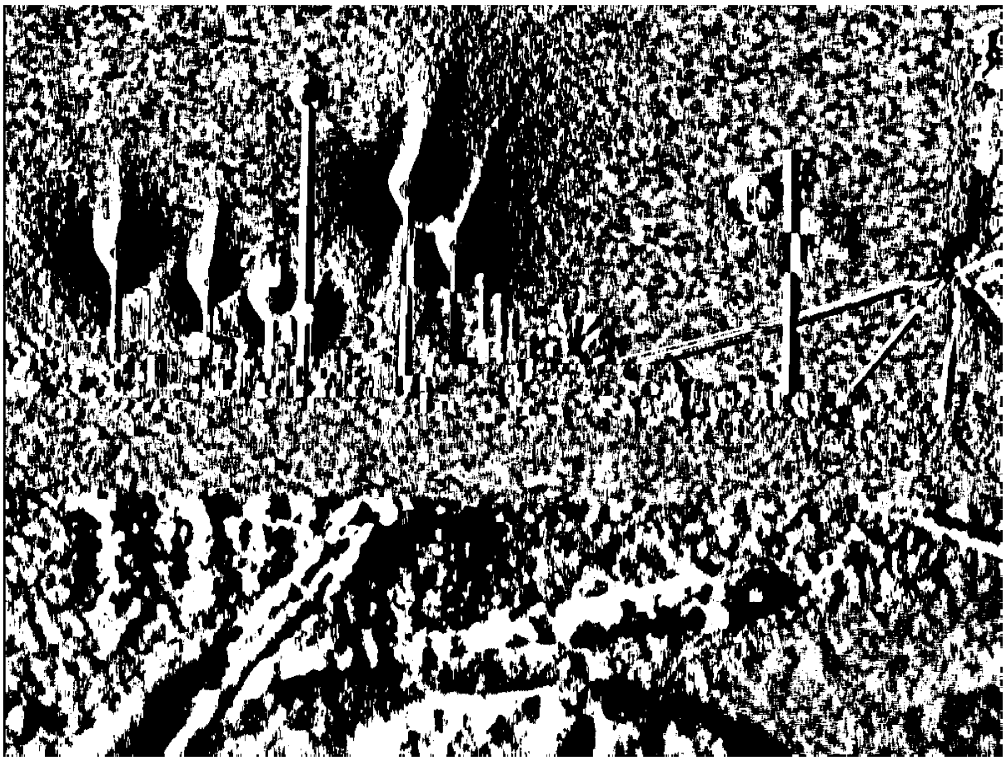


Εικόνα 2.9:Gray-code απεικόνιση για το τρίτο σετ εικόνων.

Για το **τέταρτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.10: Η εικόνα Left.

Εικόνα 2.11: Η εικόνα Right.

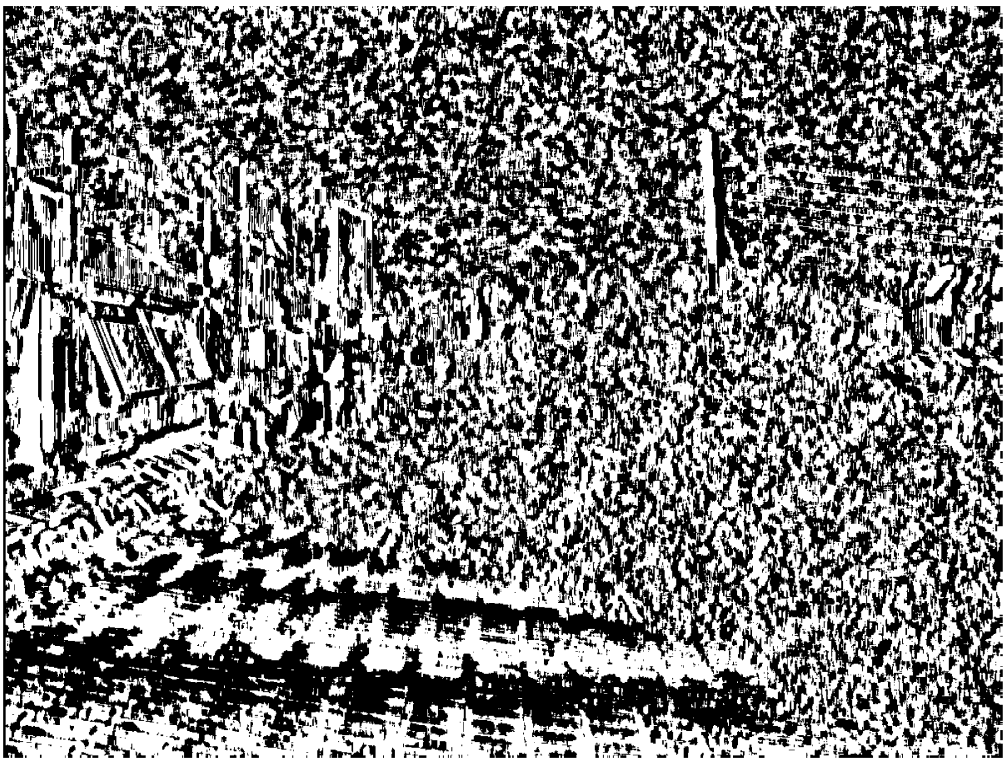


Εικόνα 2.12: Gray-code απεικόνιση για το τέταρτο σετ εικόνων.

Για το **πέμπτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.13: Η εικόνα Left.

Εικόνα 2.14: Η εικόνα Right.



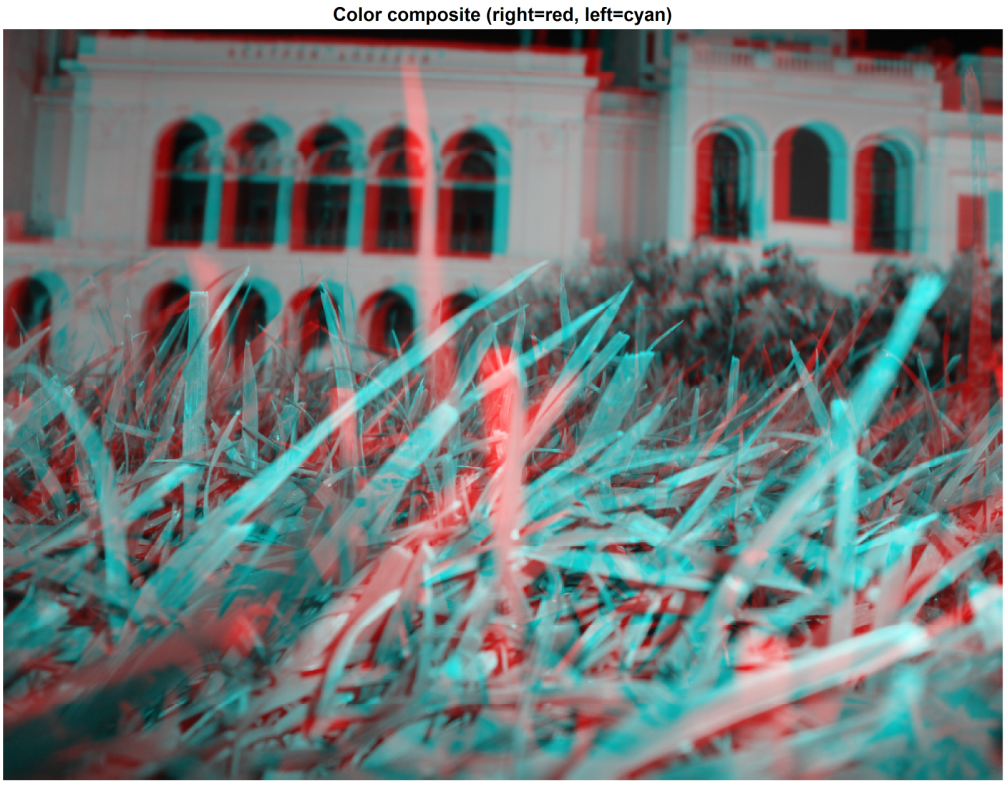
Εικόνα 2.15: Gray-code απεικόνιση για το πέμπτο σετ εικόνων.

Εναλλακτικός αλγόριθμος stereo – vision.

Στη συνέχεια θα παρουσιάσουμε τα αποτελέσματα της χρήσης του εναλλακτικού αλγορίθμου stereo – vision. Στην πρώτη εικόνα θα παρουσιάσουμε το αποτέλεσμα της σύνθεσης των δύο εικόνων, δεξιά και αριστερή λήψη, και στη συνέχεια την Grey – code εικόνα, με τη μέθοδο που περιγράφηκε νωρίτερα.

Εικόνα 2.1: Η εικόνα Left.

Εικόνα 2.2: Η εικόνα Right.



Εικόνα 2.18: Η εικόνα με αναπαράσταση των δύο αρχικών για το πρώτο σετ. Η δεξιά με μπλε και η αριστερή με κόκκινο χρώμα.

Εικόνα 2.19: Gray-code απεικόνιση για το πρώτο σετ εικόνων με χρήση του εναλλακτικού stereo – vision αλγορίθμου.

Για το **δεύτερο ζεύγος** εικόνων έχουμε:

Εικόνα 2.4: Η εικόνα Left.

Εικόνα 2.5: Η εικόνα Right.

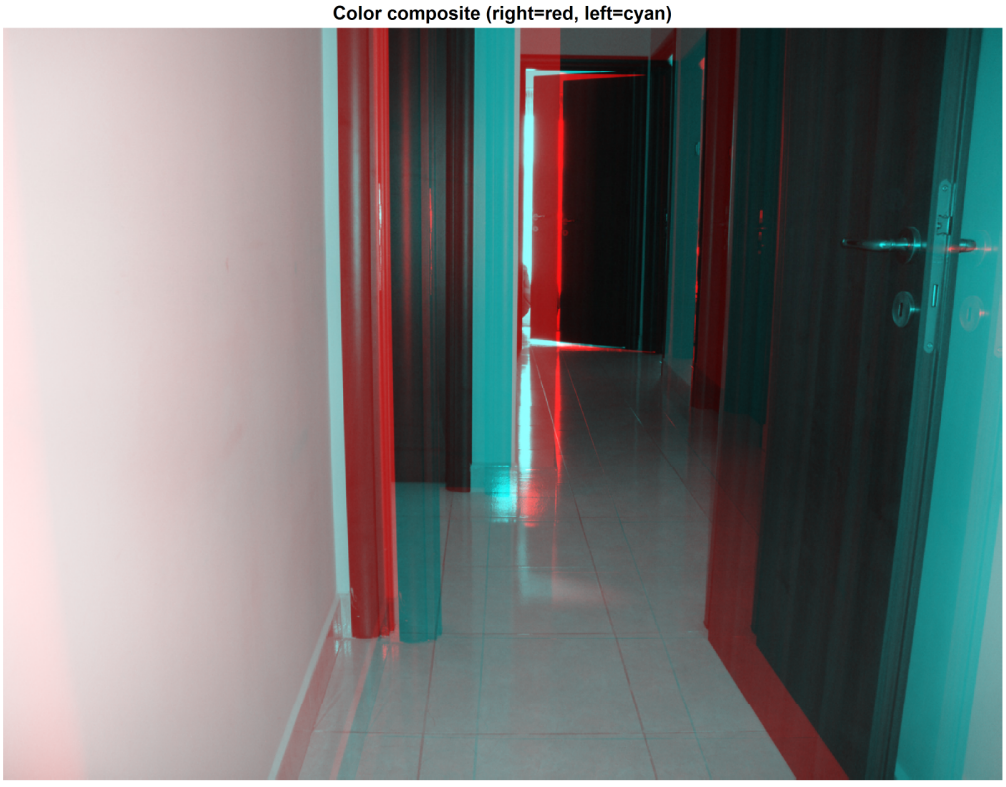
Εικόνα 2.22: Η εικόνα με αναπαράσταση των δύο αρχικών για το δεύτερο σετ. Η δεξιά με μπλε και η αριστερή με κόκκινο χρώμα.

Εικόνα 2.23: Gray-code απεικόνιση για το δεύτερο σετ εικόνων με χρήση του εναλλακτικού stereo – vision αλγορίθμου.

Για το **τρίτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.7: Η εικόνα Left.

Εικόνα 2.8: Η εικόνα Right.



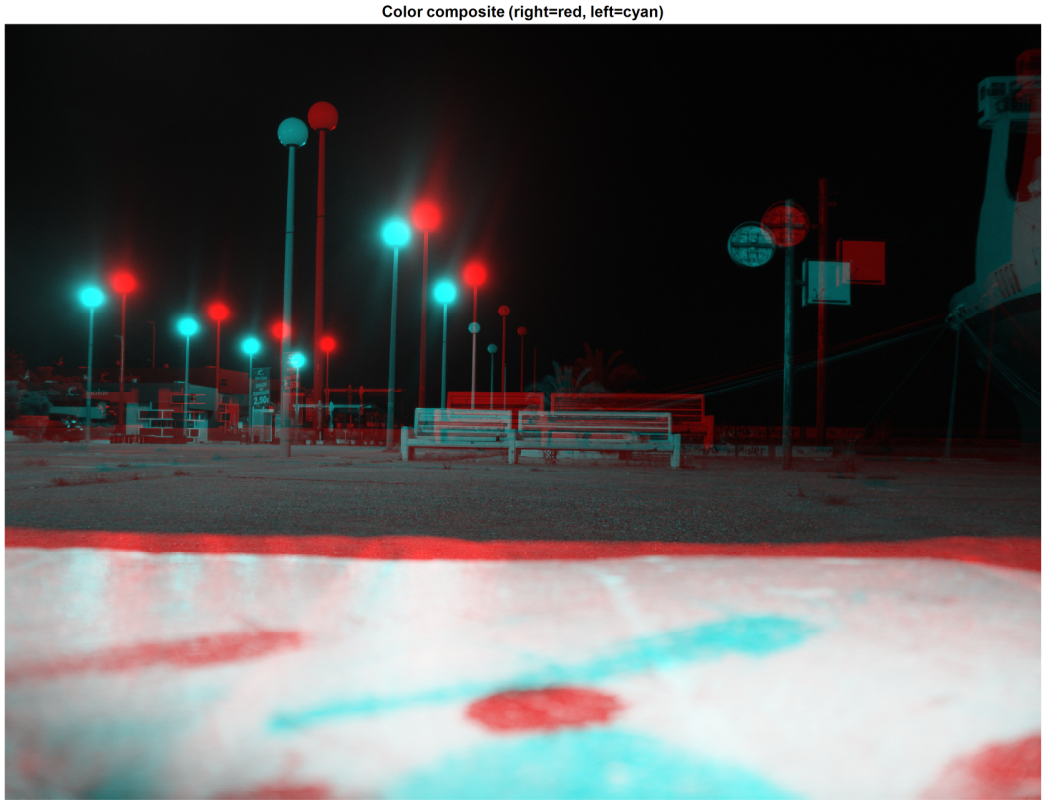
Εικόνα 2.26: Η εικόνα με αναπαράσταση των δύο αρχικών για το τρίτο σετ. Η δεξιά με μπλε και η αριστερή με κόκκινο χρώμα.

Εικόνα 2.27: Gray-code απεικόνιση για το τρίτο σετ εικόνων με χρήση του εναλλακτικού stereo – vision αλγορίθμου.

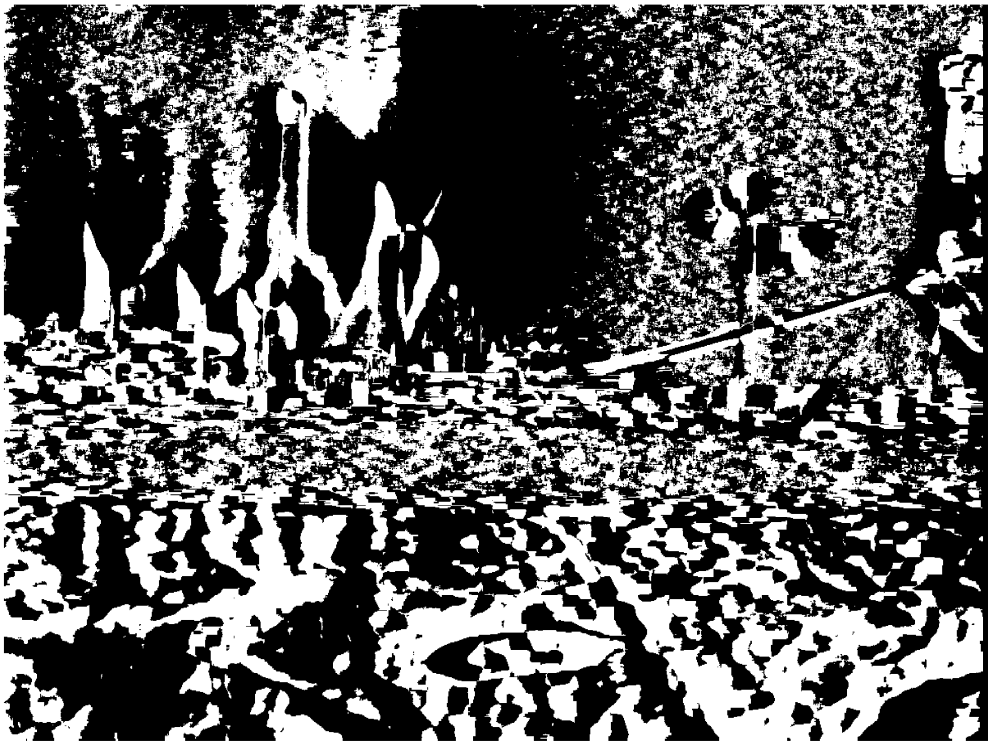
Για το **τέταρτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.10: Η εικόνα Left.

Εικόνα 2.11: Η εικόνα Right.



Εικόνα 2.30: Η εικόνα με αναπαράσταση των δύο αρχικών για το τέταρτο σετ. Η δεξιά με μπλε και η αριστερή με κόκκινο χρώμα.

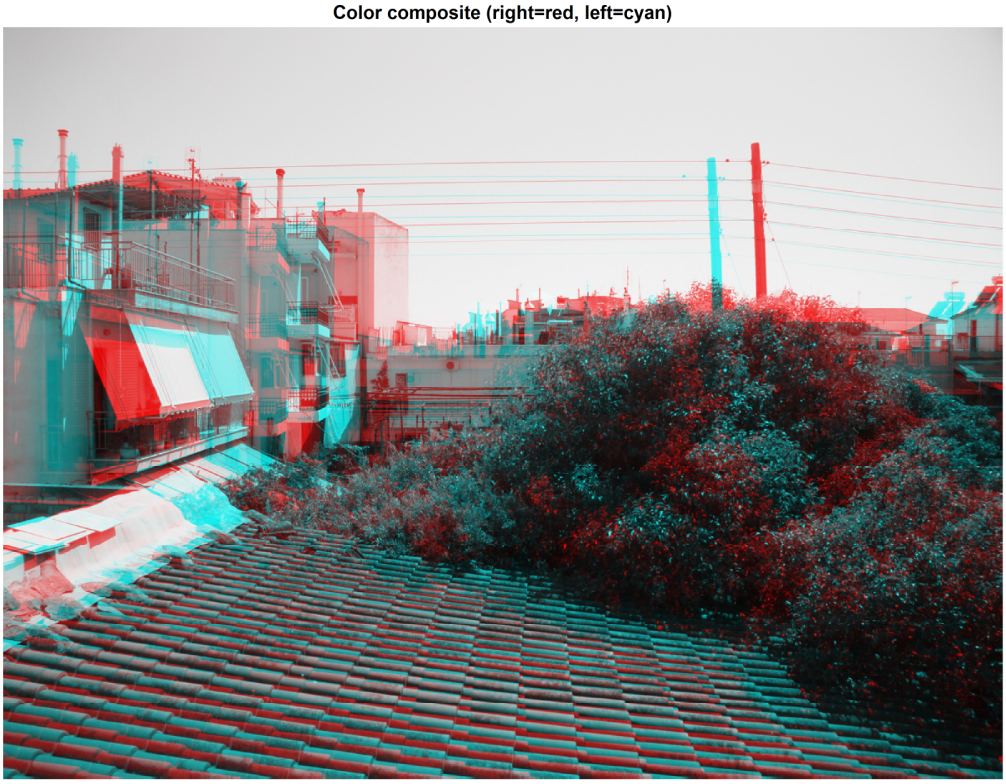


Εικόνα 2.31 Gray-code απεικόνιση για το τέταρτο σετ εικόνων με χρήση του εναλλακτικού stereo – vision αλγορίθμου.

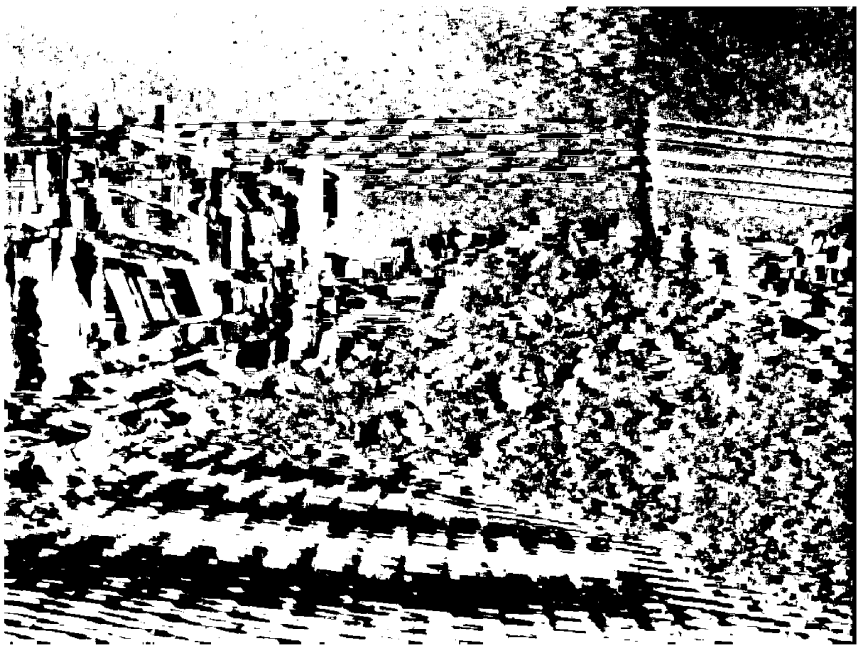
Για το **πέμπτο ζεύγος** εικόνων έχουμε:

Εικόνα 2.13: Η εικόνα Left.

Εικόνα 2.14: Η εικόνα Right.



Εικόνα 2.34: Η εικόνα με αναπαράσταση των δύο αρχικών για το τέταρτο σετ. Η δεξιά με μπλε και η αριστερή με κόκκινο χρώμα.



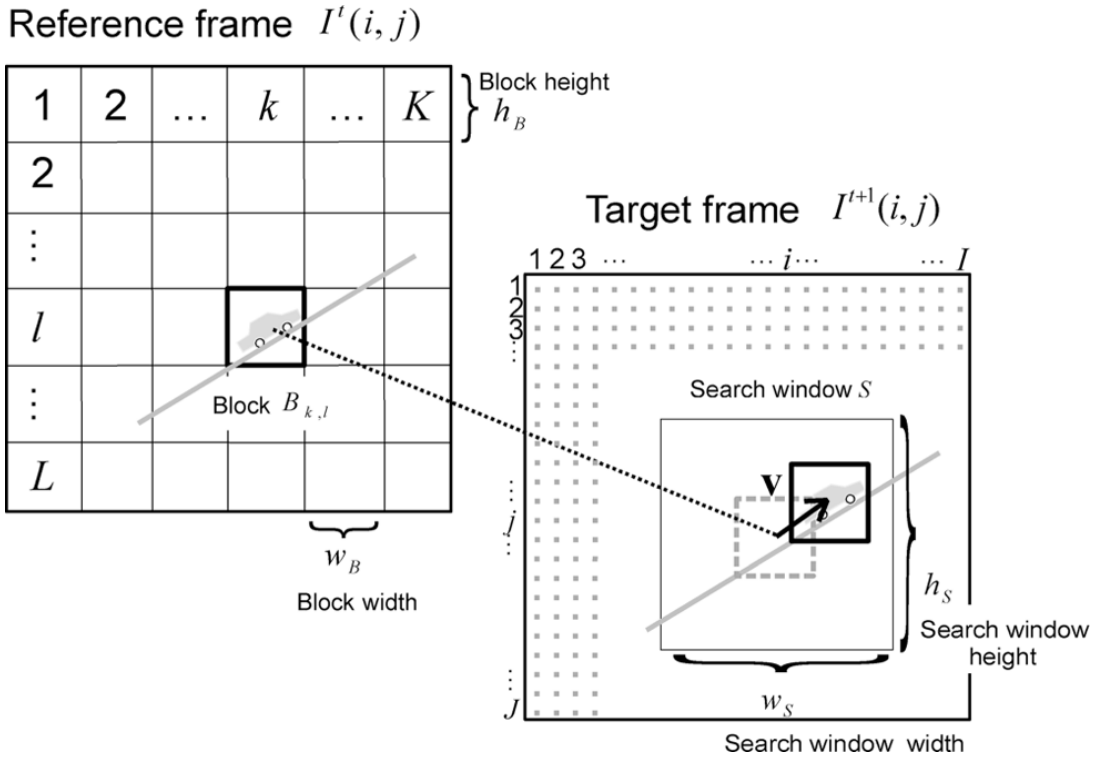
Εικόνα 2.35: Gray-code απεικόνιση για το τέταρτο σετ εικόνων με χρήση του εναλλακτικού stereo – vision αλγορίθμου.

Παρατηρούμε πως ο δεύτερος αλγόριθμος απαιτεί λιγότερη μνήμη για την επεξεργασία της εικόνας, αυτός είναι ο λόγος για τον οποίο απαιτεί περισσότερη ώρα για την ολοκλήρωση του, γιατί πραγματοποιεί την επεξεργασία σε κάθε στήλη ώστε να παράγει τη Grey-coded εικόνα. Χρησιμοποιεί την εικόνα που έχει συνθέσει με βάση τις αρχικές εικόνες και υπολογίζει την ανισότητα ώστε να αποδώσει τιμή στην Grey-coded εικόνα. Άρα, αυτός είναι ο αλγόριθμος που θα υλοποιήσουμε σε γλώσσα περιγραφής υλικού, VHDL. Στη συνέχεια αυτής της εργασίας θα δώσουμε κάποια βασικά στοιχεία για την VHDL και θα δημιουργήσουμε τον κώδικα του αλγορίθμου.

Κεφάλαιο 3ο

3.1.1 Η λογική του matching

Τα τελευταία χρόνια παρατηρούμε ότι έχει γίνει πιο έντονη η ανάγκη για δημιουργία αλγορίθμων υψηλής πιστότητας για την επεξεργασία στερεοσκοπικών εικόνων. Ένα μεγάλο κομμάτι αυτών των τεχνικών έχει δώσει βάση στο matching, δηλαδή στην ένωση μεταξύ των εικόνων οι οποίες διαφέρουν όσον αφορά τη θέση λήψης τους μερικά εκατοστά. Στην παρούσα εργασία η απόσταση λήψης τους θα είναι, όπως έχουμε ήδη αναφέρει έξι εκατοστά. Ένα άλλο κομμάτι της έρευνας πάνω σε αυτούς τους αλγορίθμους πραγματοποιείται χρησιμοποιώντας υπάρχουσες βάσεις δεδομένων, με σετ εικόνων, ώστε να ελέγξουμε την απόδοση τους. Δυστυχώς όμως όσο και αν βελτιώνονται οι αλγόριθμοι που πραγματοποιούν το «ταίριασμα», δεν επαρκούν οι υπάρχουσες βάσεις εικόνων, αλλά και δεν ανανεώνονται κατάλληλα ώστε να μπορούμε να αξιολογήσουμε τους αλγορίθμους. Εκτός αυτού, οι περισσότεροι αλγόριθμοι υλοποιούν σε μεγάλο ποσοστό το «ταίριασμα» πολλών εικονοστοιχείων στα υπάρχοντα σετ εικόνων, για τα οποία έχουμε μια δεδομένη απόδοση. [6]



Εικόνα 4: Σχηματική περιγραφή του matching μεταξύ δύο εικόνων.

3.1.2 Μαθηματικό υπόβαθρο των αλγορίθμων.

Οι αλγόριθμοι που επιτελούν matching, βασίζονται κυρίως στη λογική, ότι ελέγχουν ένα σύνολο εικονοστοιχείων, συνήθως τετράγωνο και έχουν σκοπό την εύρεση του αντίστοιχου block στη μετατοπισμένη εικόνα. Ένα σύνηθες μέγεθος block είναι 16x16 εικονοστοιχεία. Η μετρική στην οποία βασίζεται στο «ταίριασμα» βασίζεται στο ελάχιστο απόλυτο σφάλμα, Mean Absolute Error (MAD), το οποίο δίνεται από τον τύπο:

$$MAD = \frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} | \mathrm{C}_{ij} - \mathrm{R}_{ij} |$$

Επίσης το ελάχιστο τετραγωνικό σφάλμα, Mean Square Error (MSE), το οποίο δίνεται από τον παρακάτω τύπο:

$$MSE = \frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (\mathrm{C}_{ij} - \mathrm{R}_{ij})^2$$

Οι παραπάνω τύποι εφαρμόζονται σε κάθε εικονοστοιχείο μεταξύ των επιλεγμένων block των δύο εικόνων. Το εκφράζει το μέγεθος του block και οι παράγοντες C_{ij} και R_{ij} τα εικονοστοιχεία στα οποία εφαρμόζεται η σύγκριση στο δεδομένο block αντίστοιχα. Η εικόνα που δημιουργείται από το συνδυασμό των δύο εικόνων χαρακτηρίζεται από Peak SNR (Signal to Noise Ratio) το οποίο υπολογίζεται ως εξής:

$$PSNR = 10\log_{10}\frac{(\text{peak to peak value of original data})^2}{MSE}$$

Προφανώς χρησιμοποιώντας κατάλληλο κατώφλι μπορούμε να αναλύσουμε τους αλγορίθμους σε αποδοτικούς ή μη – αποδοτικούς με βάση το PSNR.

3.1.3 Τεχνικές των block - matching αλγορίθμων.

Στη συνέχεια θα κάνουμε μια σύντομη αναφορά σε μερικές τεχνικές με βάση τις οποίες πραγματοποιούν το «ταίριασμα» οι αλγόριθμοι αυτοί. Δεν θα μπούμε στη διαδικασία σύγκρισης των αλγορίθμων αυτών καθώς αφήνεται στον αναγνώστη να αποφασίσει με βάση τις δικές του απαιτήσεις. Υπάρχουν περιπτώσεις στις οποίες αυτό που μας ενδιαφέρει είναι η ταχύτητα του αλγορίθμου ενώ άλλες στις οποίες μας ενδιαφέρει η υψηλή ποιότητα της τελικής

εικόνας. Σε αυτή την εργασία θα περιοριστούμε στην αναφορά των βασικών τους χαρακτηριστικών, αγνοώντας την απόδοσή τους. Τους αλγορίθμους αυτούς θα τους χρησιμοποιήσουμε ως βάση για να αναπτύξουμε τον αλγόριθμο της εργασίας μας.

3.1.3.1 Exhaustive Αναζήτηση

Αυτή η τεχνική χρησιμοποιεί μια συνάρτηση κόστους σε κάθε πιθανή τοποθεσία του block. Έτσι οδηγούμαστε στο βέλτιστο ταίριασμα στο παράθυρο αναφοράς που χρησιμοποιούμε. Το αποτέλεσμα είναι η εικόνα η οποία προκύπτει από το matching να έχει υψηλό PSNR συγκρινόμενη με οποιοδήποτε άλλο αλγόριθμο.

3.1.3.2 Αναζήτηση Tree Step

Αναφερόμαστε σε έναν σχετικά παλιό block – matching, οποίος τρέχει ως εξής:

- Η αναζήτηση ξενικά από το κέντρο.
- Το βήμα της αναζήτησης είναι S = 4 και η παράμετρος της αναζήτησης είναι P=7.
- Αναζητά μεταξύ 8 τοποθεσιών +/- S εικονοστοιχεία γύρω από το κέντρο της εικόνας το οποίο εκφράζεται ως σημείο (0,0).
- Διάλεξε μεταξύ 9 περιοχών αναζήτησης, βασιζόμενος στην ελάχιστη συνάρτηση κόστους.
- Επέλεξε νέα περιοχή αναζήτησης.
- Θέσε βήμα S = S/2.
- Επανέλαβε τη διαδικασία αναζήτησης έως ότου το βήμα γίνει S=1.

3.1.3.3 Λογαριθμική αναζήτηση Δύο Διαστάσεων

Σχετίζεται σε μεγάλο βαθμό με αναζήτηση Tree Step , αλλά είναι πιο ακριβής για μεγαλύτερα παράθυρα αναζήτησης. Τα βήματα που ακολουθεί είναι τα εξής:

- Ξεκίνα με περιοχή αναζήτησης το κέντρο της εικόνας.
- Επέλεξε βήμα αναζήτησης S=8.
- Κάνε αναζήτηση σε 4 περιοχές σε απόσταση S από το κέντρο της εικόνας.
- Βρες την τοποθεσία που ικανοποιεί την ελάχιστη συνάρτηση κόστους.

- Αν υπάρχει σημείο στο οποίο η αναζήτηση έχει καλύτερα αποτελέσματα από το κέντρο:
    - Επίλεξε το ως νέο κέντρο.
    - Επανέλαβε τα βήματα 2 και 3.
- Αν το βέλτιστο σημείο είναι στο κέντρο θέσε βήμα S = S/2.
- Αν το S = 1 όλες οι περιοχές σε απόσταση 8 από το κέντρο έχουν ελεγχθεί.
- Θέσε τα διανύσματα κίνησης ως τα σημεία με τη μικρότερη συνάρτηση κόστους.

3.1.3.2 Αναζήτηση Νέο Tree Step

Η λογική του αλγορίθμου περιγράφεται παρακάτω:

- Ξεκίνα με περιοχή αναζήτησης το κέντρο της εικόνας.
- Πραγματοποίησε αναζήτηση σε 8 περιοχές σε απόσταση +/- S με S = 4 και σε 8 περιοχές σε απόσταση +/- S με S = 1 γύρω από το κέντρο (0,0).
- Διάλεξε από τις περιοχές που έχει πραγματοποιηθεί η αναζήτηση αυτή με το ελάχιστο κόστος.
- Αν υπάρχουν περιοχές που συμπίπτουν σε ελάχιστο κόστος, σταμάτα και θέσε ως διάνυσμα κίνησης το (0,0).
- Αν η ελάχιστη συνάρτηση κόστους είναι μοναδική για μια περιοχή θέσε S = 1, και νέα περιοχή αναζήτησης την περιοχή αυτή.
    - Έλεγξε τα βάρη αυτών των περιοχών, ανάλογα την περιοχή μπορεί να ελέγξεις 3 ή 5 πόντους.
- Θέσε ως νέο διάνυσμα κίνησης αυτό με μικρότερο βάρος.

3.1.3.3 Απλή και Αποδοτική αναζήτηση

Αποτελεί μια βελτίωση του κλασικού Tree Step και ακολουθεί την εξής λογική:

Πρώτη φάση:

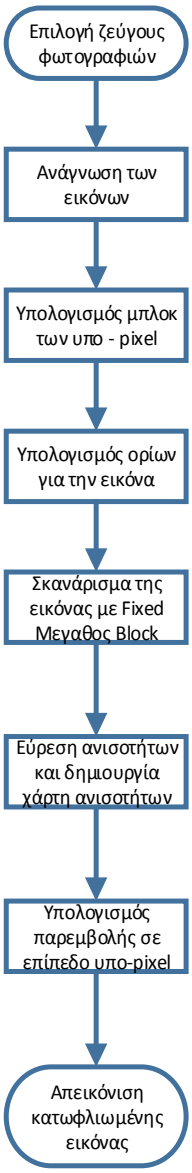
- Διαίρεσε την περιοχή αναζήτησης σε 4 τεταρτημόρια.
- Κάνε αναζήτηση στα τρία τεταρτημόρια, στο κεντρικό, έστω ότι το ονομάζω Α, και σε άλλα δύο, έστω Β και Γ, τα οποία απέχουν S = 4 περιοχές από το Α σε ορθογώνια μετρική.
- Θα πρέπει να ισχύει κάτι από τα παρακάτω:

- Αν (MAD(A)>=MAD(B) and MAD(A)>=MAD(C)), επέλεξε σημεία στο δεύτερο τεταρτημόριο IV.
- Αν (MAD(A)>=MAD(B) and MAD(A)<=MAD(C)), επέλεξε σημεία στο δεύτερο τεταρτημόριο I.
- Αν MAD(A)<MAD(B) and MAD(A)<MAD(C)), επέλεξε σημεία στο δεύτερο τεταρτημόριο II.
- Αν (MAD(A)<MAD(B) and MAD(A)>=MAD(C)), επέλεξε σημεία στο δεύτερο τεταρτημόριο III.

Δεύτερη φάση:

- Βρες την τοποθεσία με το μικρότερο βάρος.
- Θέσε νέα περιοχή αναζήτησης όπως βρέθηκε από το παραπάνω σημείο.

- Θέσε το νέο βήμα αναζήτησης ίσο με S = S/2.
- Επανέλαβε τα βήματα έως ότου S = 1.
- Επέλεξε την περιοχή με το μικρότερο βάρος.


Επιλογή ζεύγους φωτογραφιών
Ανάγνωση των εικόνων
Υπολογισμός μπλοκ των υπο - pixel
Υπολογισμός ορίων για την εικόνα
Σκανάρισμα της εικόνας με Fixed Μεγαθος Block
Εύρεση ανισοτήτων και δημιουργία χάρτη ανισοτήτων
Υπολογισμός παρεμβολής σε επίπεδο υπο-pixel
Απεικόνιση κατωφλιωμένης εικόνας

Ο VHDL κώδικας για κάθε διαδικασία που περιγράφηκε στο παραπάνω flowchart

Χρησιμοποιούμε το εργαλείο Matlab και κάνουμε resize τις εικόνες ώστε να βελτιώσουμε την απόδοση του αλγορίθου. Επειδή η φωτογραφική μηχανή που χρησιμοποιήσαμε είναι υψηλής ευκρίνειας οι αρχικές εικόνες ξεπερνούν τα 3 MB οπότε πραγματοποιούμε resize της τάξεως του 10% της αρχικής εικόνας και παίρνουμε εικόνες μεγέθους 300x400x3 τις οποίες θα επεξεργαστούμε στο vhdl κώδικα. [7][8]

Το τμήμα του κώδικα για την ανάγνωση των εικόνων [10][11]

```
-- xrhsh vivliothikwn

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use std.textio.all;


-- anagnwsh twn eikonwn

entity IMAGE_READ is

Port ( CLK : in  STD_LOGIC;

IMAGE_LOAD : IN  STD_LOGIC:='1';

IMAGE_DONE : OUT  STD_LOGIC);

end entity;


architecture Behavioral of IMAGE_READ is

type image is array (1 to 300,1 to 400, 1 to 3) of std_logic_vector(7 downto 0);

file IMAGE_FILE:png open read_mode is "C:\Users\BIB\Desktop\nikos\codes";


begin

PROCESS(CLK,ROW,COLUMN,IMAGE_LOAD)

variable LINE_NUMBER:line;

variable TEMP_PIXEL_VALUE: bit_vector(7 downto 0);

variable image_matrix:IMAGE;


BEGIN

if (clk'event and clk = '1') then


IF(IMAGE_LOAD='1') THEN

for i in 1 to 300 loop

for j in 1 to 400 loop

for k in 1 to 3 loop

   readline (IMAGE_FILE, LINE_NUMBER);

   read (LINE_NUMBER, TEMP_PIXEL_VALUE);

   image_matrix(i,j)                    :=
to_stdlogicvector(TEMP_PIXEL_VALUE);

      if(i=300 and j=400 and k=3) then

         TEMP_image_done:='1';

         image_done<='1';

      else

         TEMP_image_done:='0';

         image_done<='0';

      end if;

END LOOP;

END LOOP;

END LOOP;

END IF;
```

```
END IF;

END PROCESS;


end Behavioral;
```

Στη συνέχεια παρουσιάζεται ο κώδικας για τη δημιουργία των μπλοκ της εικόνας. Το μέγεθος κάθε μπλοκ είναι 16x16.

```
-- block registers


library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_arith.all;

use ieee.std_logic_unsigned.all;

use work.all;

use work.motion_estimation_8x8_package.all;


entity block_registers is

port

(

clk: in std_logic;

-- Input ports

words: in REG_COLS;

we_blk: in WE_REG_ROWS;


blk: out REG_ROWS


);

end block_registers;


architecture behaviour of  block_registers is


component regNbits is

generic

(

N: integer  :=32

);


port

(

-- Input ports

clk: in std_logic;

data_in: in  std_logic_vector(DATA_WIDTH-1 DOWNTO 0);

we: in  std_logic;


-- Output ports

data_out:  out  std_logic_vector(DATA_WIDTH-1  DOWNTO
0)

);

end component;



SIGNAL i: INTEGER RANGE 0 TO NUM_ROWS-1;

SIGNAL j: INTEGER RANGE 0 TO NUM_COLS-1;


signal  blk_registers_data_in,  blk_registers_data_out:
REG_ROWS;


begin


Row_NUM_ROWS_less_1:
```

```
FOR j IN 0 TO NUM_COLS-1 GENERATE
register_blk: regNbits
generic map ( N => 8 )
port map(
clk=> clk,
data_in=> words(j),
we=> we_blk(NUM_ROWS-1)(j),
data_out=> blk_registers_data_out(NUM_ROWS-1)(j)
);
END GENERATE;

Row:
FOR i IN 0 TO NUM_ROWS-1 GENERATE
Cols:
FOR j IN 0 TO NUM_COLS-1 GENERATE

ifgenerate:
IF (i /= NUM_ROWS-1) GENERATE

register_blk: regNbits
generic map ( N => DATA_WIDTH )
port map(
clk=> clk,
data_in=> blk_registers_data_out(i+1)(j),
we=> we_blk(i)(j),
data_out=> blk_registers_data_out(i)(j)
);

END GENERATE;

END GENERATE;
```

```
END GENERATE;


blk <= blk_registers_data_out;

end behaviour;

-- block registers manager

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use work.all;
use work.motion_estimation_8x8_package.all;

entity block_registers_manager is
port
(
rst: in std_logic;
clk: in std_logic;
-- Input ports
we_start_blk: in std_logic;
we_blk: out WE_REG_ROWS
);
end block_registers_manager;

architecture behaviour of  block_registers_manager is

TYPE STATE_TYPE is (start, s01);
```

```
SIGNAL sig_count, count_in, count_out: INTEGER RANGE
0 TO (BLOCK_DIMENSION+1) := 0;

SIGNAL sel_count, we_count: std_logic;

SIGNAL state, next_state: STATE_TYPE;

SIGNAL sig_we_blk: WE_REG_ROWS;


begin


we_blk <= sig_we_blk;


state_machine: process(state, we_start_blk, count_out)

begin

case (state) is


when start =>


if (we_start_blk = '1') then

next_state <= s01;

we_count <= '1';

else

we_count <= '0';

next_state <= start;

end if;


sel_count <= '0';

sig_we_blk <= (others => (others => '0'));


when s01 =>

if (count_out = (BLOCK_DIMENSION-1)) then

next_state <= start;

we_count <= '0';

else

next_state <= s01;

we_count <= '1';

end if;


sel_count <= '1';


sig_we_blk <= (others => (others => '1'));

end case;

end process;



registers: process(clk)

begin

if (clk'event and clk = '1') then

if (we_count = '1') then

count_out <= count_in;

end if;

end if;

end process;


update_state: process (clk, rst)

begin

if (rst = '1') then

state <= start;

elsif (clk'event and clk = '1') then

state <= next_state;

end if;

end process;


sig_count <= count_out + 1;


mux_reg_count:
```

```
with sel_count select

count_in <=

0 when '0',

sig_count when '1',

0when others;


end behaviour;
```

Και τέλος παρουσιάζουμε των μπλοκ matching αλγόριθμο ο οποίος θα υλοποιηθεί μεταξύ των δύο εικόνων(Left.png και Right.png)

```
-- block matching tmhma


library ieee;

use ieee.std_logic_1164.all;

use ieee.numeric_std.all;

use work.hpel_package.all;


entity hp_macroblock_buffer is

   port (

      clock : in std_logic;

      din   : in hp_macroblock_buffer_i;

      dout  : out hp_macroblock_buffer_o

   );

end hp_macroblock_buffer;


architecture hp_macroblock_buffer of hp_macroblock_buffer is

   type pel_ram_t is array (0 to 7) of std_logic_vector(63 downto 0);

   type ref_ram_t is array (0 to 15) of std_logic_vector(79 downto 0);

   type col_ram_t is array (0 to 15) of std_logic_vector(79 downto 0);

   type row_ram_t is array (0 to 15) of std_logic_vector(71 downto 0);

   type diag_ram_t is array (0 to 15) of std_logic_vector(71 downto 0);


   signal pel_mem  : pel_ram_t;

   signal ref_mem  : ref_ram_t;

   signal col_mem  : col_ram_t;

   signal row_mem  : row_ram_t;

   signal diag_mem : diag_ram_t;


   signal pel_dout  : std_logic_vector(63 downto 0);

   signal ref_dout  : std_logic_vector(79 downto 0);

   signal col_dout  : std_logic_vector(79 downto 0);

   signal row_dout  : std_logic_vector(71 downto 0);

   signal diag_dout : std_logic_vector(71 downto 0);

begin


   dout.pel.dout  <= pel_dout;

   dout.ref.dout  <= ref_dout;

   dout.col.dout  <= col_dout;

   dout.row.dout  <= row_dout;

   dout.diag.dout <= diag_dout;


   process (clock, din.pel)

   begin

      if clock'event and clock = '1' then

         if din.pel.wren = '1' then

            pel_mem(to_integer(unsigned(din.pel.addr)))     <= din.pel.din;

         end if;
```

```
        pel_dout <= pel_mem(to_integer(unsigned(din.pel.addr)));
    end if;
  end process;

  process (clock, din.ref)
  begin
    if clock'event and clock = '1' then
      if din.ref.wren = '1' then
        ref_mem(to_integer(unsigned(din.ref.addr))) <= din.ref.din;
      end if;

      ref_dout <= ref_mem(to_integer(unsigned(din.ref.addr)));
    end if;
  end process;

  process (clock, din.col)
  begin
    if clock'event and clock = '1' then
      if din.col.wren = '1' then
        col_mem(to_integer(unsigned(din.col.addr))) <= din.col.din;
      end if;

      col_dout <= col_mem(to_integer(unsigned(din.col.addr)));
    end if;
  end process;

  process (clock, din.row)
  begin
    if clock'event and clock = '1' then
      if din.row.wren = '1' then
        row_mem(to_integer(unsigned(din.row.addr))) <= din.row.din;
      end if;

      row_dout <= row_mem(to_integer(unsigned(din.row.addr)));
    end if;
  end process;

  process (clock, din.diag)
  begin
    if clock'event and clock = '1' then
      if din.diag.wren = '1' then
        diag_mem(to_integer(unsigned(din.diag.addr))) <= din.diag.din;
      end if;

      diag_dout <= diag_mem(to_integer(unsigned(din.diag.addr)));
    end if;
  end process;

end hp_macroblock_buffer;
```

Το τελικό βήμα είναι να μετατρέψουμε την εικόνα που προκύπτει από το block matching σε binary ώστε να έχουμε την ίδια μορφή αρχείου με αυτή των αποτελεσμάτων του εργαλείου matlab. Χρησιμοποιώντας ένα κατώφλι θα θέσουμε τις τιμές που είναι μικρότερες από το

κατώφλι ως 0 και στη συνέχεια θα θέσουμε τις τιμές που είναι μεγαλύτερες ή ισες από το κατώφλι ως 1 και θα αναπαραστήσουμε το αποτέλεσμα.

Κεφάλαιο 4ο

4.1 Περιγραφή της υλοποίησης

Κατά τη διάρκεια της έρευνας για το κομμάτι της υλοποίησης σε γλώσσα περιγραφής υλικού VHDL (VHSIC Hardware Description Language). Η γλώσσα VDHL χρησιμοποιείται στο σχεδιασμό ηλεκτρονικών αυτομάτων για την περιγραφή ψηφιακών και μεικτών ψηφιακών συστημάτων, δηλαδή συστημάτων με ψηφιακά και αναλογικά στοιχεία, όπως τα FPGAs.

Τα FPGAs, field-programmable gate arrays, είναι τύπος προγραμματιζόμενου ολοκληρωμένου κυκλώματος γενικής χρήσης το οποίο διαθέτει πολύ μεγάλο αριθμό τυποποιημένων πυλών και άλλων ψηφιακών λειτουργιών όπως απαριθμητές, καταχωρητές μνήμης, γεννήτριες PLL κα. Σε ένα τέτοιο κύκλωμα θα εφαρμόσουμε τον κώδικα και θα υλοποιήσουμε τον αλγόριθμο stereo vision ώστε να παράγουμε τις Gray coded εικόνες από το συνδυασμό των αρχικών εικόνων που έχουμε τραβήξει. Σήμερα συναντήσαμε πολλές εφαρμογές σε αυτό το πεδίο, στο οποίο διεξάγεται ακόμα μεγάλο κομμάτι έρευνας.

4.2 Γενική περιγραφή των εργαλείων

Για να υλοποιήσουμε αυτό το τμήμα της εργασίας θα χρησιμοποιήσουμε τα εξής δύο εργαλεία:

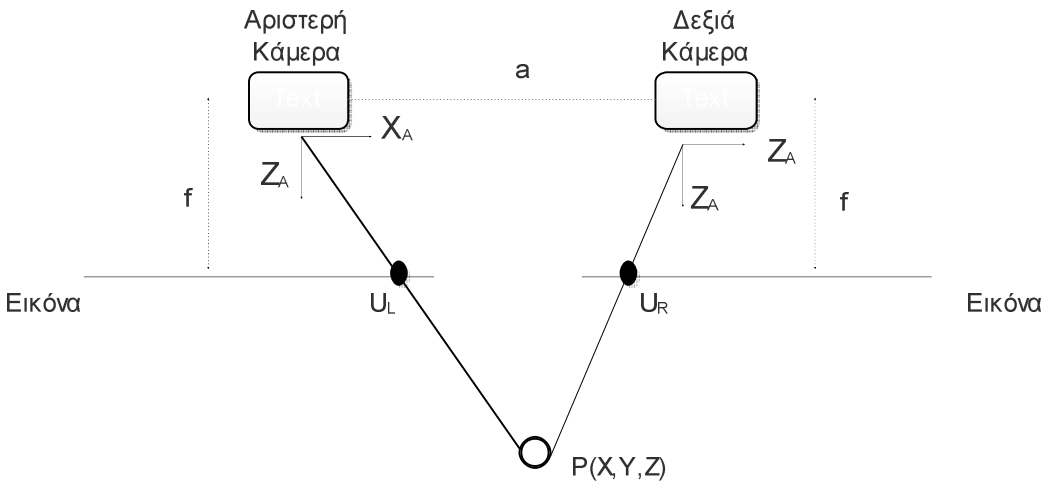
- Xilinx Ise webPack [9][10]
- Modelsim Student Pack – Altera [11]

Χρησιμοποιούμε τα δύο παραπάνω εργαλεία για να γράψουμε τον κώδικα και να μπορούμε να κάνουμε debugging και στη συνέχεια με τα built in FRGAs του ISE θα ελέγξουμε τη αποδοτικότητα του αλγορίθμου.

Οι έκδοση του εργαλείου ISE είναι το webPack το οποίο παρέχεται δωρεάν από τη Xilinx για εκπαιδευτικούς σκοπούς και φυσικά δεν παρέχει το πλήρες πακέτο. Επίσης χρησιμοποιούμε το Modelsim της Altera, την έκδοση Student Pack την οποία παρέχει δωρεάν η εταιρεία για εκπαιδευτικούς σκοπούς επίσης.

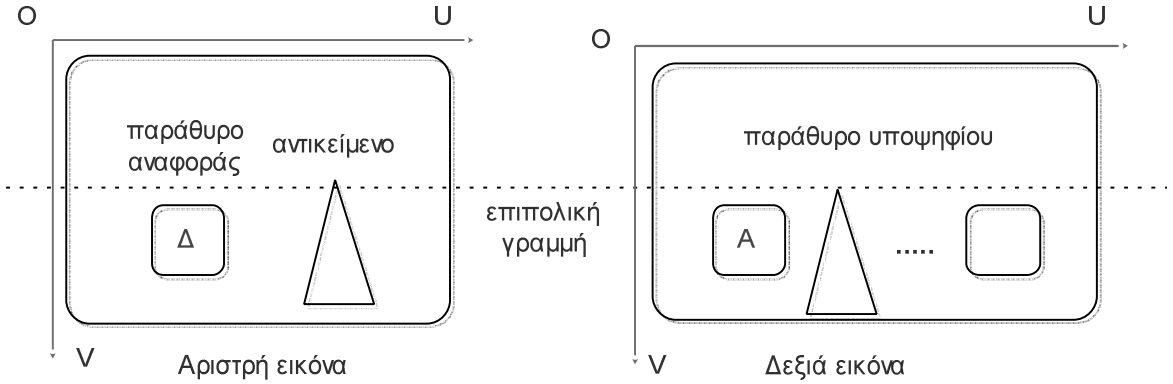
4.3 Εικονική Περιγραφή της διαδικασίας

Στη συνέχεια θα χρησιμοποιήσουμε κώδικα ανοιχτού λογισμικού για να υλοποιήσουμε τη στερεοσκοπική απεικόνιση η οποία περιγράφεται από την παρακάτω εικόνα:

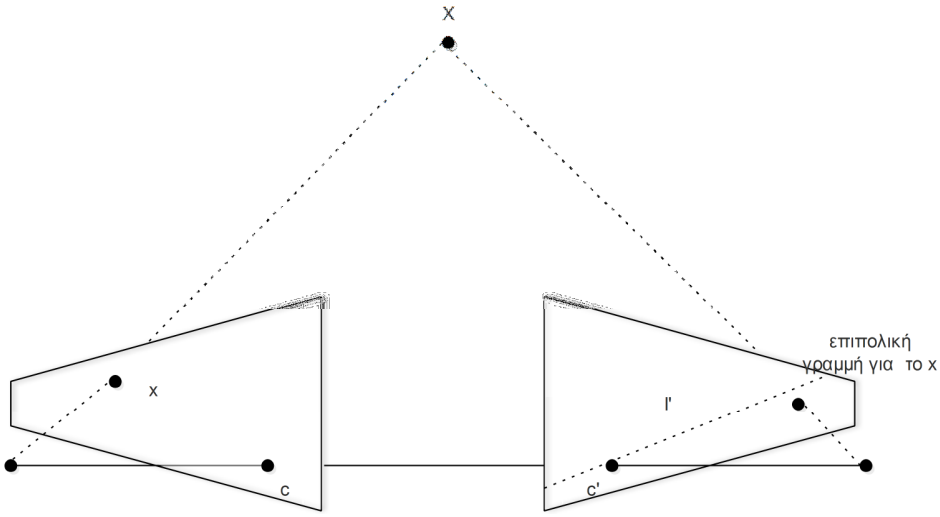


Εικόνα 4.1: Απλοποιημένο σύστημα στερεοσκοπικής όρασης

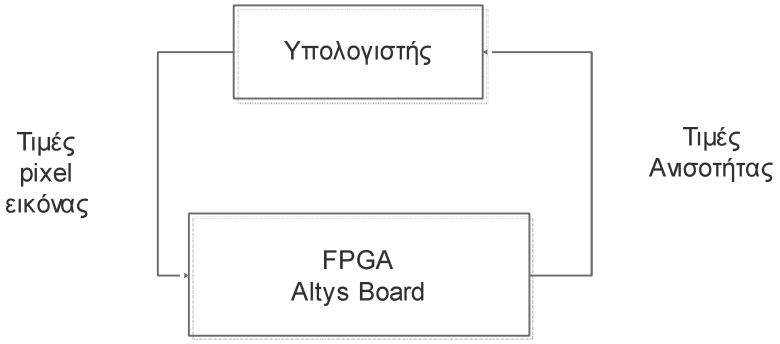
Η λογική με την οποία λειτουργεί και ο κώδικας περιγραφής υλικού παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 4.2: Εύρεση σημείων αλληλεπίδρασης μεταξύ των δύο εικόνων.



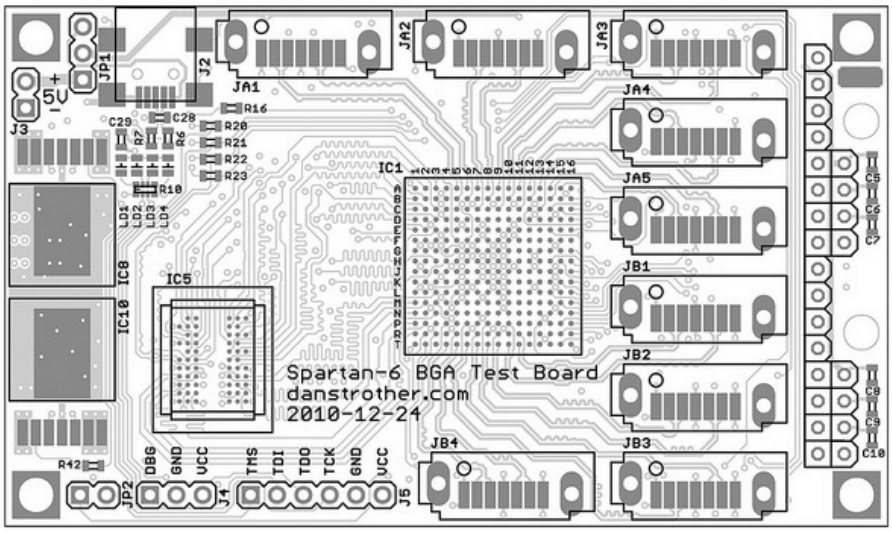
Εικόνα 4.3: Η επιπολική γραμμή που δείχνει το σημείο X για τις δύο εικόνες.



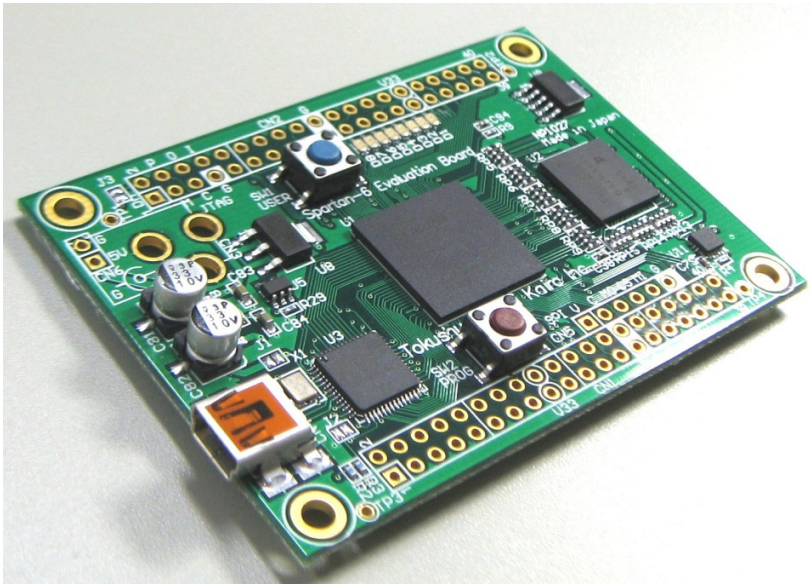
Εικόνα 4.4: Το module για τον έλεγχο του FRGA. Ο υπολογιστής στέλνει την εικόνα στο FPGA και τις τιμές της ανισότητας.

4.4 Το FPGA που θα χρησιμοποιήσουμε

Το ολοκληρωμένο που θα χρησιμοποιήσουμε για την υλοποίηση είναι της οικογένειας Spartan 6.



Εικόνα 4.5: Ολοκληρωμένο της οικογένειας Spartan.



Εικόνα 4.6: Εικόνα ολοκληρωμένου Spartan 6.

Εικόνα 4.7: Εξωτερική μορφή του ολοκληρωμένου Spartan 6.

Η οικογένεια Spartan 6 προσφέρει μια ισοζυγισμένη λύση χαμηλού κόστους, χαμηλής κατανάλωσης, και Low risk. Η σημερινή έκδοση προσφέρει 42% μικρότερη κατανάλωση και 12% μεγαλύτερες επιδόσεις σε σχέση με τις συσκευές προηγούμενης γενιάς. Τα συγκεκριμένα

ολοκληρωμένα προσφέρουν προηγμένη τεχνολογία διαχείρισης ισχύς, έως 150K λογικά κύτταρα, ολοκλήρωση σε PCI Express blocks, βελτιωμένη υποστήριξη μνήμης, 250MHz DSP κομμάτια και 3,2 Gbps low-power πομπό-δέκτες. [15] [16]

4.5 Παρουσίαση Αποτελεσμάτων

Σε αυτό το σημείο θα παρουσιάσουμε τις εικόνες με το disparity του συνδυασμού τους.

Για το πρώτο σετ εικόνων έχουμε:

Εικόνα 4.8: Δεξιά λήψη.

Εικόνα 4.9: Αριστερή λήψη.

Εικόνα 4.10: Disparity των εικόνων.

Για το δεύτερο σετ εικόνων έχουμε:

Εικόνα 4.11: Δεξιά λήψη.

Εικόνα 4.12: Αριστερή λήψη.

Εικόνα 4.13: Disparity των εικόνων.

Για το δεύτερο σετ εικόνων έχουμε:

Εικόνα 4.11: Δεξιά λήψη.

Εικόνα 4.12: Αριστερή λήψη.

Εικόνα 4.13: Disparity των εικόνων.

Για το τρίτο σετ εικόνων έχουμε:

Εικόνα 4.14: Δεξιά λήψη.

Εικόνα 4.15: Αριστερή λήψη.

Εικόνα 4.16: Disparity των εικόνων.

Για το τέταρτο σετ εικόνων έχουμε:

Εικόνα 4.17: Δεξιά λήψη.

Εικόνα 4.18: Αριστερή λήψη.

Εικόνα 4.19: Disparity των εικόνων.

Για το πέμπτο σετ εικόνων έχουμε:

Εικόνα 4.20: Δεξιά λήψη.

Εικόνα 4.21: Αριστερή λήψη.

Εικόνα 4.22: Disparity των εικόνων.

Κεφάλαιο 5

5.1 FPGAs σε Εφαρμογές Τεχνικής Στερεοσκοπικής όρασης

Σε αυτό το κεφάλαιο θα αναλύσουμε τη λογική με την οποία μπορούμε να πραγματοποιήσουμε σε πραγματικό χρόνο τη στερεοσκοπική απεικόνιση, χρησιμοποιώντας τα παραπάνω ζεύγη εικόνων, σε επίπεδο hardware. Στην εποχή μας έχουν πραγματοποιηθεί ουσιαστικές βελτιώσεις όσον αφορά τους block – matching αλγορίθμους και τη ψηφιακή υλοποίηση τους.

Παρ' όλο που block – matching είναι η πιο ακριβή διαδικασία, της στερεοσκοπικής απεικόνισης, δεν υπάρχει κάποια γενική λύση σε αυτό το πρόβλημα στο πρόβλημα του matching αν και πολλές εκτιμήσεις δείχνουν να βελτιστοποιούν αυτό το πρόβλημα, παρά τα προβλήματα που προκύπτουν από το θόρυβο που προσθέτει η κάμερα και τις φωτομετρικές διαφορές.

Για να πετύχουμε λοιπόν τη 3D απεικόνιση από τις εικόνες που έχουμε λάβει και δεδομένου ότι δεν μπορούμε να χρησιμοποιήσουμε active light συστήματα, η λύση είναι η ψηφιακή στερεοσκοπική απεικόνιση. Επειδή η στερεοσκοπική απεικόνιση χρησιμοποιείται πλέον σε ένα μεγάλο πλήθος εφαρμογών, ο σκοπός είναι η δημιουργία αυτόνομων συστημάτων με μικρό κόστος και χαμηλή κατανάλωση ισχύος.

Για πραγματικού χρόνου στερεοσκοπικές εφαρμογές, απαιτείται hardware ειδικού σκοπού, όπως επεξεργαστές Ψηφιακής Επεξεργασίας ή ειδικού σκοπού FPGAs. Τα βασικά πλεονεκτήματα αυτών των FPGAs είναι το χαμηλό κόστος και ο μικρός κύκλος ρολογιού. Αυτά τα ειδικού σκοπού συστήματα επιτρέπουν τον παραλληλισμό και τα επίπεδα παροχέτευσης που απαιτούνται για του αλγορίθμους στερεοσκοπικής όρασης.

Η ανάπτυξη της λογικής μας επιτρέπει να παράγουμε προγραμματίσιμες συσκευές, που υλοποιούν στερεοσκοπική ανάλυση σε υψηλού ρυθμού βίντεο. Το θετικό στοιχείο είναι ότι τα συστήματα αυτά μπορούν να χρησιμοποιηθούν και σε άλλες εφαρμογές.

5.2 Θεωρητική Περιγραφή της Διαδικασίας

Σε αυτό το μέρος θα περιγράψουμε τον τρόπο με τον οποίο, χρησιμοποιώντας τον παραπάνω κώδικα θα υλοποιούσαμε σε πρακτικό επίπεδο τη στερεοσκοπική απεικόνιση, με χρήση ενός FPGA.

Για την διαδικασία αυτή θα χρησιμοποιούσαμε ένα εργαλείο με το οποίο μπορούμε να πραγματοποιήσουμε συγγραφή του κώδικα και debugging. Στη συνέχεια αφού ολοκληρωθεί η παραπάνω διαδικασία μπορούμε να μετατρέψουμε τον κώδικα σε δυαδική μορφή τον κώδικα και να προγραμματίσουμε το ολοκληρωμένο έτσι ώστε να μπορούμε να πραγματοποιήσουμε τη στερεοσκοπική ανάλυση στο ολοκληρωμένο. Για να αναλύσουμε αυτή τη διαδικασία θα χρησιμοποιήσουμε το εργαλείο ISE webPack της Xilinx ώστε να παρουσιάσουμε τη διαδικασία.

5.3 Σχετικές Υλοποιήσεις

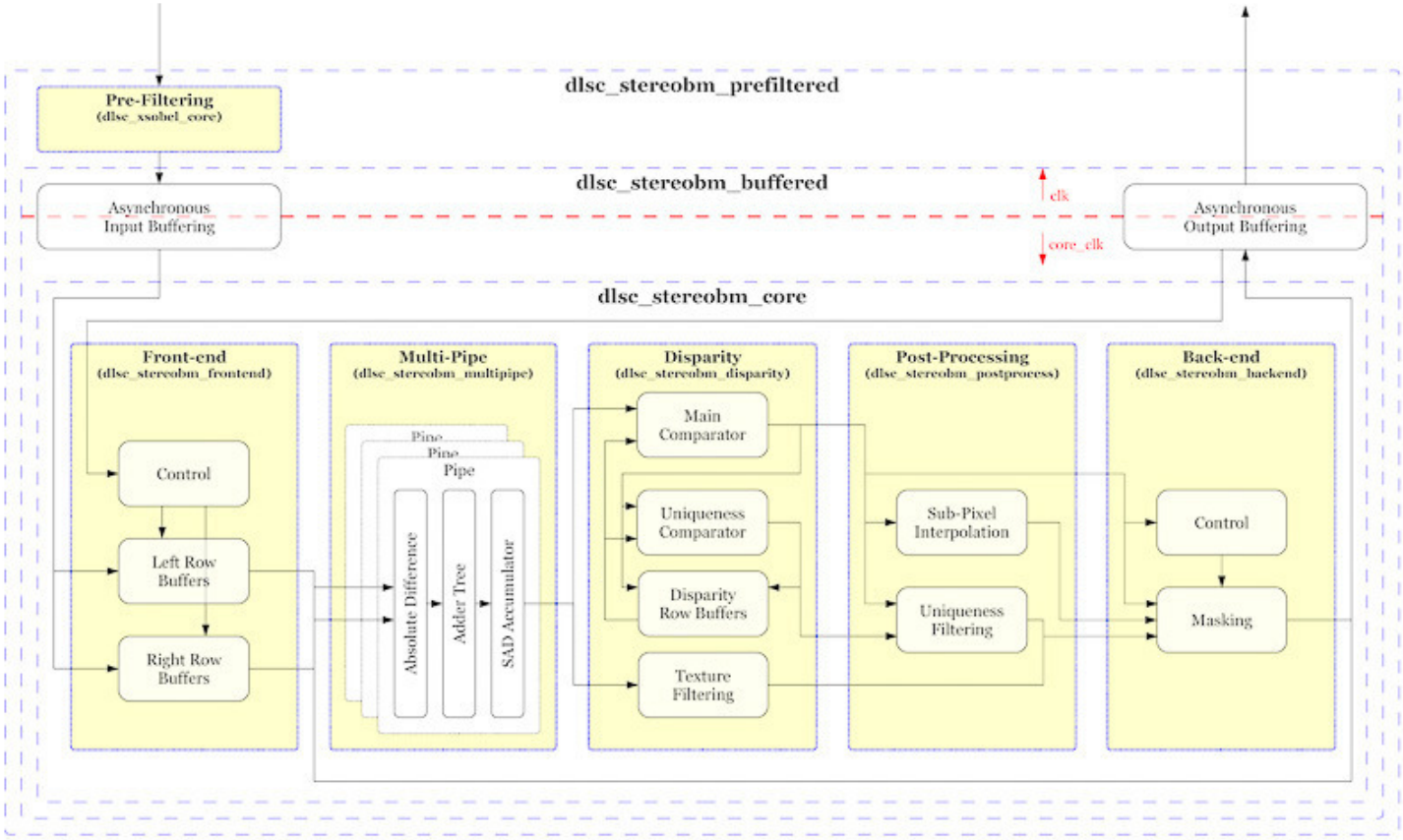
Παράλληλα με την ανάπτυξη αλγορίθμων για την για την στερεοσκοποική όραση, έχουμε και πολλά παραδείγματα για τις υλοποιήσεις για σε πραγματικό χρόνο.

Το 1993 ο Feugeras και οι συνεργάτες του παρουσίασαν μια υλοποίηση σε πραγματικό χρόνο, η οποία βασίστηκε σε ολοκληρωμένα και πλακέτες για ψηφιακή επεξεργασία. Έφτιαξαν μια κανονικοποιημένη εκδοχή αλγορίθμου συσχέτισης δυο εικόνων, με μεταξύ τους μετατόπιση, μεγέθους 256x256 pixel.

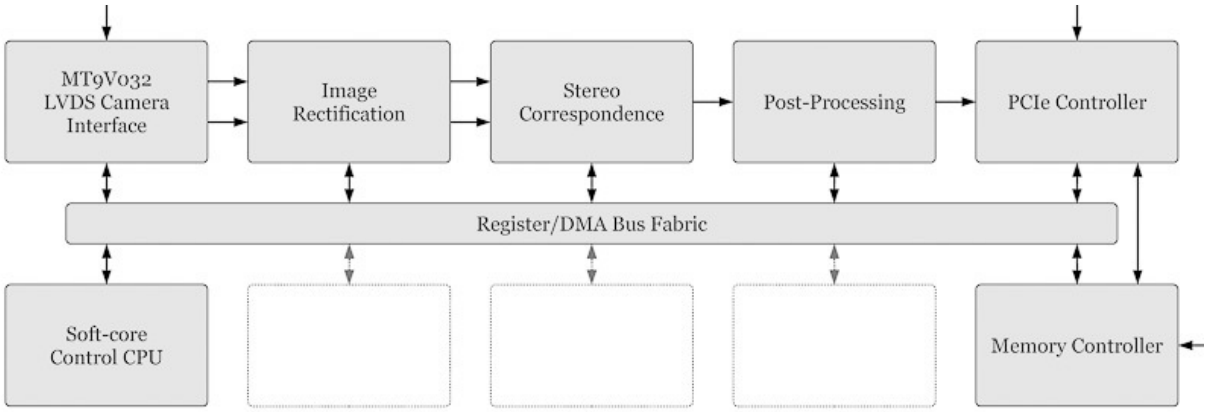
Την ίδια χρονιά ο Webb, υλοποίησε έναν ταχύ αλγόριθμο με πολλαπλά κατώφλια στη CMU Warp μηχανής. Σε αυτό το πρωτότυπο χρησιμοποιούνταν 64 iWarp επεξεργαστές οι οποίοι έφτασαν 15 fps σε εικόνες 265x240 pixel. Επίσης, ο Kanade και οι συνεργάτες του έφτιαξαν επεξεργαστή που πέτυχε 30 fps, χρησιμοποιώντας ειδικά κατασκευασμένο υλικό, αποτελούμενο από οκτώ C40 ολοκληρωμένα για ψηφιακή επεξεργασία σήματος.

Real-Time System	**Image Size**	**Frame Rate**	**Algorithm**
(1993) INRIA	256x256	3.6 fps	Normailized Correlation
(1993) CMU iWarp	256x240	15 fps	SSAD
(1996) CMU Stereo Machine	256x240	30 fps	SSAD
(1997) SRI SVS	320x240	30 fps	SAD
(1997) PARTS	320x240	42 fps	Census
(1999) SAZAN	320x240	20 fps	SSAD
(2001) SRI SVS	320x240	30 fps	SAD

Η διαδικασία η οποία έχει ακολουθηθεί για τον αλγόριθμο που θα χρησιμοποιήσουμε παρουσιάζεται παρακάτω: Ουσιαστικά έχουμε έναν πυρήνα για το FPGA που θα χρησιμοποιήσουμε ο οποίος επιτελεί την ανάλυση της εικόνας και μας δίνει ως αποτέλεσμα την στερεοσκοπική απεικόνισή της.



Σε υψηλότερο επίπεδο η ανάλυση η οποία επιτελείται είναι η εξής:



Η εικόνα που έχει «τραβήξει» η κάμερα αναλύεται και επεξεργάζεται σε πρώτο στάδιο στο επίπεδο stereo correspondence και στη συνέχεια επέρχεται επιπλέον επεξεργασία, μέχρι να καταλήξει στον ελεγκτή μνήμη του ολοκληρωμένου και να περάσουν τα αποτελέσματα στη συσκευή εξόδου.

Βιβλιογραφία

[1] W. Abbeloos, “Matlab Central,” *2012*. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/28522-stereo-matching/content/stereomatch.m.

[2] “Computer Vision in Matlab,” 2013. [Online]. Available: • http://cvmatlab.blogspot.gr/2012/10/stereo-vision.html.

[3] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Int. J. Comput. Vis.*, vol. 47, no. 1–3, pp. 7–42, 2002.

[4] A. Aziz, “Image-based motion estimation in a stream programming language,” 2007.

[5] C. Cuadrado, A. Zuloaga, J. E. S. Us, and J. J. I. M. Enez, “VHDL Description of a Synthetizable and Reconfigurable Real-Time Stereo Vision Processor,” no, 2005.

[6] M. Class, “Computer Vision with MATLAB,” pp. 1–29, 2011.

[7] Wikipedia, “Block-matching algorithm.” [Online]. Available: https://en.wikipedia.org/wiki/Block-matching_algorithm.

[8] C. Citron, “Stereo Vision System Module for Low-Cost Fpgas for Autonomous Mobile Robots,” no. August, 2014.

[9] “Hardware Description of Multi-Directional Fast Sobel Edge Detection Processor by VHDL for Implementing on FPGA .” [Online]. Available: http://rms.scu.ac.ir/Files/Articles/Journals/Abstract/pxc3879872.pdf201351712950207.pdf.

[10] “VHDL based Sobel Edge Detection.” [Online]. Available: http://pnrsolution.org/Datacenter/Vol3/Issue1/161.pdf.

[11] “FPGA IMPLEMENTATION of SOBEL EDGE DETECTOR.” [Online]. Available: http://sciencepublication.org/ijast/documents/proceeding/46.pdf.

[12] Xilinx, “Xilinx ISE 10.1 Quick Start Tutorial,” pp. 1–28, 2008.

[13] U. G. April, “ISE In-Depth Tutorial,” vol. 695, pp. 1–150, 2012.

[14] A. Corporation, “Altera Software Installation and Licensing,” 2013.

[15] “Spartan-6 BGA test board.” [Online]. Available: http://danstrother.com/2011/01/16/spartan-6-bga-test-board/.

[16] “Aggrandize Bit Plane Coding using Gray Code Method.” [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.206.4518.

Παράρτημα Β: Κώδικες

Matlab

Αλγόριθμος: Εναλλακτικός Stereo-vision

```
%Now here is the Matlab code for the reading of the two images and overlaying them.

a = vision.ImageDataTypeConverter;%creates the object
b = vision.ColorSpaceConverter('Conversion','RGB to intensity');
leftI3 = step(a,Il);
leftI = step(b,leftI3);
rightI3 = step(a,Ir);
rightI = step(b,rightI3);
figure(1), clf;
imshow(rightI3), title('Right');
figure(2), clf;
imshow(cat(3,rightI,leftI,leftI)), axis image;
title('Color composite (right=red, left=cyan)');

Dbasic = zeros(size(leftI), 'single');
disparityRange = 15;
% Selects (2*halfBlockSize+1)-by-(2*halfBlockSize+1) block.
halfBlockSize = 3;
blockSize = 2*halfBlockSize+1;

tmats = cell(blockSize);

hWaitBar = waitbar(0, 'Performing basic block matching...');
nRowsLeft = size(leftI, 1);

for m=1:nRowsLeft
   % Set min/max row bounds for image block.
   minr = max(1,m-halfBlockSize);
   maxr = min(nRowsLeft,m+halfBlockSize);
   % Scan over all columns.
   for n=1:size(leftI,2)
      minc = max(1,n-halfBlockSize);
      maxc = min(size(leftI,2),n+halfBlockSize);
```

```
      % Compute disparity bounds.
      mind = max( -disparityRange, 1-minc );
      maxd = min( disparityRange, size(leftI,2)-maxc );
      template = rightI(minr:maxr,minc:maxc);
      templateCenter = floor((size(template)+1)/2);
      roi = [minc+templateCenter(2)+mind-1 ...
           minr+templateCenter(1)-1 ...
           maxd-mind+1 1];
      % Lookup proper TemplateMatcher object; create if empty.
      if isempty(tmats{size(template,1),size(template,2)})
         tmats{size(template,1),size(template,2)} = ...
            vision.TemplateMatcher('ROIInputPort',true);
      end
      thisTemplateMatcher = tmats{size(template,1),size(template,2)};

      loc = step(thisTemplateMatcher, leftI, template, roi);
      Dbasic(m,n) = loc(1) - roi(1) + mind;
   end
   waitbar(m/nRowsLeft,hWaitBar);
end

close(hWaitBar);

figure, imshow(Dbasic)
```

Αλγόριθμος: Stereomatch

```
% Master Thesis: Real-Time Stereo Vision     Wim Abbeloos    May 2010
% Karel de Grote-Hogeschool University College, Belgium
%
% FAST MATLAB STEREO MATCHING ALGORITHM (SAD)
% Description: This function performs the computationally expensive step of
% matching two rectified and undistorted stereo images.  The output is a
% dense disparity map.  If camera parameters are known, this allows for
% three dimensional reconstruction.
%
% Please note this function requires the Image Processing Toolbox!
%
% [spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc)
%
```

```
% The standard images included are from
% [1]       D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.
% International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002.
% [2]       D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light.
% In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), volume 1, pages 195-202, Madison, WI, June 2003.

function [spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc)

% Set Parameters
WS  = uint16(windowsize);              % Set window size, must be uneven
WS2 = uint16( ( WS - 1 ) / 2 );        % Half window
D   = uint16(disparity)+1;             % number of disparities

% Read image sizes
heightL = uint16( size( imgleft, 1 ) );    heightR = uint16( size( imgright, 1 ) );
widthL  = uint16( size( imgleft, 2 ) );    widthR  = uint16( size( imgright, 2 ) );
if ( heightL ~= heightR  ||  widthL ~= widthR )
   error('Height and width of left and right image must be equal');
end

% Initialization
pcost = zeros( heightL, widthL, D, 'uint8' );
wcost = zeros( heightL, widthL, D, 'single' );
dmap  = zeros( heightL, widthL, 'uint8' );
dcost = zeros( heightL, widthL, 'single' );
h = zeros(WS,WS,'double'); h(1,1) = 1; h(1,WS) = -1; h(WS,1) = -1; h(WS,WS) = 1;

% Calculate pixel cost
for Dc = 1 : D
  maxL = widthL + 1 - Dc;
  pcost(:, Dc : widthL, Dc ) = imabsdiff( imgright( :, 1 : maxL), imgleft( :, Dc : widthL) );
end

% Calculate integral cost
icost = single(pcost);
icost = cumsum( cumsum( icost ), 2 );

% Calculate window cost
wcost = imfilter(icost,h,'same','symmetric');
```

```
% Search disparity value
[ dcost(:,D+WS2:widthL), dmap(:,D+WS2:widthL)] = min( wcost(:,D+WS2:widthL,:),[], 3 );
for j=WS2+1:D+WS2
   [ dcost(:,j), dmap(:,j)] = min( wcost(:, j, 1 : (j - WS2) ),[], 3 );
end

% Adjust disparity map
warning off;
spdmap = single(dmap-1);

% Subpixel interpolation

if spacc==1
for j=D+1:widthL
for i=1:heightL
if dmap(i,j)>1 && dmap(i,j)<D
p = polyfit2((single(dmap(i,j)-2:dmap(i,j))),shiftdim(single(wcost(i,j,dmap(i,j)-1:dmap(i,j)+1)),1),2);
temp=roots(p);
spdmap(i,j)=real(temp(1));
end
end
end
end
warning on;
```

run:

```
clear all
clc
%enallaktikos algorithmos stereo-vision
%first set
cd('images/1')
Ir = imresize(imread('Right.png'), 0.5);
Il = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
alg1

clear all
clc
%second set
cd('images/2')
Ir = imresize(imread('Right.png'), 0.5);
Il = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
```

```
alg1

clear all
clc
%third set
cd('images/3')
Ir = imresize(imread('Right.png'), 0.5);
Il = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
alg1

%fourth set
cd('images/4')
Ir = imresize(imread('Right.png'), 0.5);
Il = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
alg1

%fifth set
cd('images/5')
Ir = imresize(imread('Right.png'), 0.5);
Il = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
alg1



%steromatch algorithmos
%first set
cd('images/1')
imgright = imresize(imread('Right.png'), 0.5);
imgleft = imresize(imread('Left.png'), 0.5);

cd ..
cd ..

%set parameteres
windowsize = 9;
disparity = 0.5;
spacc = 0.2;

[spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc);

figure, imshow(spdmap)

%second set
cd('images/2')
imgright = imresize(imread('Right.png'), 0.5);
imgleft = imresize(imread('Left.png'), 0.5);
cd ..
cd ..

%set parameteres
windowsize = 9;
disparity = 0.5;
```

```
spacc = 0.2;

[spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc);

figure, imshow(spdmap)

clear all
clc

%third set
cd('images/3')
imgright = imresize(imread('Right.png'), 0.5);
imgleft = imresize(imread('Left.png'), 0.5);
cd ..
cd ..

%set parameteres
windowsize = 9;
disparity = 0.5;
spacc = 0.2;

[spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc);

figure, imshow(spdmap)

clear all
clc
%fourth set
cd('images/4')
imgright = imresize(imread('Right.png'), 0.5);
imgleft = imresize(imread('Left.png'), 0.5);
cd ..
cd ..

%set parameteres
windowsize = 9;
disparity = 0.5;
spacc = 0.2;

[spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc);

figure, imshow(spdmap)
 clear all
clc
%fifth set
cd('images/5')
imgright = imresize(imread('Right.png'), 0.5);
imgleft = imresize(imread('Left.png'), 0.5);
cd ..
cd ..
 %set parameteres
windowsize = 9;
disparity = 0.5;
spacc = 0.2;

[spdmap, dcost, pcost, wcost] = stereomatch(imgleft, imgright, windowsize, disparity, spacc);

figure, imshow(spdmap)
```