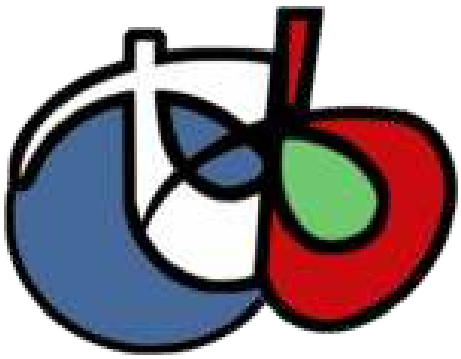
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολη Αγρονομων-Τοπογραφων Μηχανικων
Τομεας Τοπογραφιας-Εργαστηριο Τηλεπισκοπησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διερεύνηση μεθόδων ανίχνευσης μεταβολών σε τηλεπισκοπικές εικόνες, με ανάπτυξη ελεύθερου λογισμικού στην πλατφόρμα **Orfeo Toolbox**

Μαρία Βακαλοπούλου



Υπεύθυνος Καθηγητής
Δημήτριος Αργιαλάς

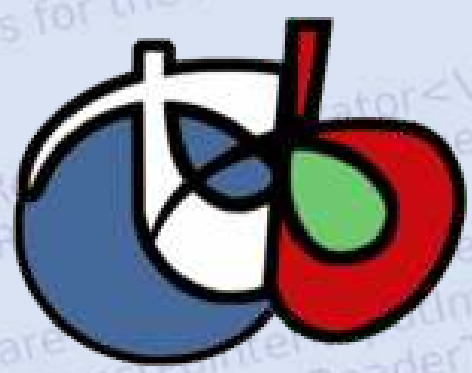
Αθήνα, Οκτώβρης 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολη Αγρονομων-Τοπογραφων Μηχανικων
Τομεας Τοπογραφιας-Εργαστηριο Τηλεπισκοπησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διερεύνηση μεθόδων ανίχνευσης μεταβολών σε τηλεπισκοπικές εικόνες, με ανάπτυξη ελεύθερου λογισμικού στην πλατφόρμα Orfeo Toolbox

Μαρία Βακαλοπούλου



Υπεύθυνος Καθηγητής
Δημήτριος Αργιαλάς

Αθήνα, Οκτώβρης 2011

Αφιερώνεται

Στους γονείς μου
Παντελή και Πέγκυ

Στην αδελφή μου
Έφη

Πρόλογος

Η παρούσα Διπλωματική Εργασία εκπονήθηκε στο πλαίσιο της ολοκλήρωσης των προπτυχιακών σπουδών μου στη Σχολή Αγρονόμων και Τοπογράφων Μηχανικών (ΣΑΤΜ) του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ) της Αθήνας και ακολουθεί τη δομή που προβλέπεται από τον αντίστοιχο Οδηγό Σπουδών της σχολής. Το συγκεκριμένο θέμα δε, ανατέθηκε από το Εργαστήριο Τηλεπισκόπησης του Τομέα Τοπογραφίας της Σχολής.

Ο σύγχρονος Τοπογράφος Μηχανικός υπηρετεί μια πολυσυλλεκτική επιστήμη η οποία αποτελείται από ένα ευρύτατο πεδίο γνώσεων όπως είναι η Γεωδαισία, η Φωτογραμμετρία, η Φωτοερμηνεία-Τηλεπισκόπηση, η Χαρτογραφία, η Επιστήμη της Γεωγραφικής Πληροφορίας και τα Συστήματα των Γεωγραφικών Πληροφοριών, το Κτηματολόγιο, η ανάλυση, ο σχεδιασμός και η οργάνωση του πολυδιάστατου χώρου καθώς και η μελέτη Συγκοινωνιακών, Υδραυλικών και Τεχνικών Έργων.Όλος αυτός ο συνδυασμός των γνώσεων αποσκοπεί στο να εξοπλίσει τον Τοπογράφο Μηχανικό με υψηλού επιπέδου αντίληψη και γνώση των χωρικών ιδιοτήτων του γεωγραφικού χώρου και της ακρίβειας τους αφενός και την ικανότητα άρτιας περιγραφής και ολοκλήρωσης της γεωπληροφορίας αφετέρου.

Ειδικότερα τα τελευταία 20 χρόνια, όλα τα παραδοσιακά αντικείμενα της Σχολής έχουν αποκτήσει νέα διάσταση και δυνατότητες, κυρίως με τη είσοδο σύγχρονων τεχνολογιών όπως της δορυφορικής τεχνολογίας, των ψηφιακών συστημάτων, της πληροφορικής και των τεχνολογιών γνώσης. Ο λόγος αυτός με ώθησε στην ανάληψη της εν λόγω διπλωματικής εργασίας που σκοπό έχει την παρουσίαση και διερεύνηση του Orfeo Toolbox, μιας καινούργιας και ελεύθερης βιβλιοθήκης τηλεπισκόπησης, με μεγάλες δυνατότητες οι οποίες συνεχώς εξελίσσονται. Επίσης μέσα από την ενασχόλησή μου με το συγκεκριμένο θέμα, επεδίωξα την συμμετοχή μου σε κοινότητες ελεύθερου λογισμικού με συνεισφορά μου τόσο σε κώδικα όσο και σε εγχειρίδια στα ελληνικά. Εκτός όμως από την παρουσίαση του Orfeo Toolbox και του γραφικού του περιβάλλοντος Mondeverti, γίνεται προσπάθεια προσθήκης σε αυτά αλγορίθμων ανίχνευσης μεταβολών (change detection), προγραμματισμένα σε γλώσσα προγραμματισμού C++.

Στο σημείο αυτό, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όσους βοήθησαν να υλοποιηθεί η παρούσα εργασία. Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Καθηγητή κύριο Αργιαλά, για την απεριόριστη εμπιστοσύνη, περισσή υπομονή και την αμέριστη συμπαράσταση που επέδειξε κατά την διάρκεια εκπόνησή της. Επιπροσθέτως, ευχαριστώ τον Ιωσηφίδη Χρήστο, Ε.ΔΙ.Π., ο οποίος μαζί με τον Αργυρίδη Αργυρό, Υ.Δ., με μύησαν στον κόσμο του ελεύθερου λογισμικού. Επίσης, ιδιαίτερα ευχαριστώ τον Τζώτσο Άγγελο, Υ.Δ., που με ώθησε στην ανάληψη του συγκεκριμένου θέματος και η βοήθειά του στο ξεκίνημά μου ήταν απεριόριστη. Τέλος, θα ήθελα να ευχαριστήσω τον Δρ. Κολοκούση Πολυχρόνη, Ε.ΔΙ.Π., που ήταν πρόθυμος να απαντήσει σε οποιοδήποτε πρόβλημα αντιμετώπιζα καθ΄ όλη την διάρκεια εκπόνησης της εργασίας.

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας ήταν η παρουσίαση, ανάλυση και αξιοποίηση της βιβλιοθήκης λογισμικού, ελεύθερου κώδικα, Orfeo Toolbox και η υλοποίηση ανίχνευσης μεταβολών μέσω αυτής. Πρόκειται για μια ελεύθερη βιβλιοθήκη γραμμένη σε C++, που περιέχει πληθώρα τηλεπισκοπικών εφαρμογών. Παρόλο που το OTB αποτελεί μια νέα πλατφόρμα κώδικα, οι αλγόριθμοι που περιέχει είναι πολλές και καλύπτουν τις βασικές εργασίες τηλεπισκόπησης. Έτσι και αλλιώς η πλατφόρμα συνεχώς εμπλουτίζεται και βελτιώνεται, καθώς ανά τακτά χρονικά διαστήματα είναι διαθέσιμες καινούργιες εκδόσεις της στους χρήστες. Το γεγονός μάλιστα, ότι αποτελεί λογισμικό ελεύθερου και ανοιχτού κώδικα, καθιστά την βελτίωση του πολύ πιο εύκολη, καθώς οποιοδήποτε πρόβλημα και έλλειψη υπάρχει στον κώδικα, μπορεί να εντοπιστεί και να διορθωθεί από όλους τους χρήστες της.

Το περιεχόμενο της διπλωματικής εργασίας, χωρίζεται σε δύο ξεχωριστά μέρη. Το πρώτο μέρος, αναφέρεται στην εξοικείωση του χρήστη με την πλατφόρμα και το γραφικό της περιβάλλον (monteverdi). Στο μέρος αυτό, πραγματοποιούνται κάποιες βασικοί αλγόριθμοι τηλεπισκόπησης, έτσι ώστε ο χρήστης να εξοικειωθεί με το περιβάλλον και να καταλάβει την φιλοσοφία δόμησης του κώδικα. Οι κώδικες (εκτελέσιμα αρχεία) που εκτελέστηκαν στο συγκεκριμένο κομμάτι, είτε υπήρχαν σαν παραδείγματα μέσα στην πλατφόρμα, είτε τροποποιήθηκαν προκειμένου να εφαρμοστούν στην αντίστοιχη εφαρμογή. Σε κάθε περίπτωση όμως, μέσα από το εγχειρίδιο, ο χρήστης μπορεί να δει πως δομούνται οι βασικές εργασίες τηλεπισκόπησης σε οπτικά δεδομένα, και να κατασκευάσει τα δικά του αρχεία που να περιέχουν δικές του ρουτίνες.

Στο δεύτερο μέρος της διπλωματικής εργασίας, έγινε μια προσπάθεια εμπλουτισμού του κώδικα της πλατφόρμας, που πραγματοποιεί ανίχνευση μεταβολών. Αφού λοιπόν, παρουσιάστηκαν αναλυτικά και εκτελέστηκαν οι αλγόριθμοι ανίχνευσης μεταβολών που χρησιμοποιούν οπτικά δεδομένα και παρέχονταν ήδη από την βιβλιοθήκη, έγινε μια προσπάθεια κατανόησης του τρόπου δόμησης των ανάλογων κλάσεων έτσι ώστε αυτές να χρησιμοποιηθούν και για τους νέους αλγορίθμους που δημιουργήθηκαν. Πέρα απ΄ τους ενσωματωμένους αλγόριθμους ανίχνευσης μεταβολών, οι οποίοι προσαρμόστηκαν στις συνθήκες της εφαρμογής μας, προγραμματίστηκαν στο πλαίσιο συνεισφοράς στην ελεύθερη βιβλιοθήκη OTB και με βάση τις προδιαγραφές της, τέσσερις επιπλέον αλγόριθμοι, διαδεδομένοι σε σύγχρονα εμπορικά λογισμικά. Οι αλγόριθμοι αυτοί είναι ο αλγόριθμος της Διαφοράς Έντασης, ο Tasseled Cap, ο αλγόριθμος Διαχωρισμού Χρωμάτων και ο αλγόριθμος Διαφορών Κλίσεων. Σε πρώτο στάδιο σε κάθε αλγόριθμο έγινε ο κατάλληλος μετασχηματισμός των εικόνων και στην συνέχεια υλοποιήθηκαν και συγκρίθηκαν οι τεχνικές κανονικοποιημένου λόγου και κανονικοποιημένης διαφοράς. Όλοι οι αλγόριθμοι εφαρμόστηκαν σε εικόνες Ikonos του 2000 και 2007, για την περιοχή της Αττικής Οδού και έγινε αξιολόγηση των αποτελεσμάτων τα οποία κρίνονται πολύ ικανοποιητικά.

Abstract

The present Diploma Thesis aims at the presentation, analysis and enrichment of the free/open source library Orfeo ToolBox as well as the application of the algorithms of change detection through it. This open source library is written in C++ programming language and contains a multitude of image processing algorithms. Beside this large number of applications, it also covers the main functionalities of remote sensing image processing, however new it may be. Furthermore it is being continuously enriched and improved, since new releases are quite often available to the users. The fact that is an open source software makes its improvement easier, as every problem and deficiency that may exist in the code, could be located and repaired by its users.

This Thesis, is divided in two parts. The first part, refers to the familiarization of the user with the graphical environment of the platform (Monteverdi). Moreover several remote sensing applications are realized, in order to help the user to get familiarized with and understand the philosophy of the code. The code files executed in the particular part, were either examples in the platform or they were modified in order to apply in every application. In every case, the user can observe through this manual how the basic applications of remote sensing in optical data are constructed, and make his own files with his own routines.

The second part of this Diploma Thesis, is an effort of enriching the code of the platform, that realizes change detection techniques. After the analytic presentation and execution of the existing change detection algorithms which use optical data, an effort was made to understand the way that the existing classes were constructed and used in the existing algorithms. Apart from the use and modification of the existing change detection algorithms, there was an effort to program four new algorithms, anailable in commercial software, according to the OTB development requirements. These algorithms are the following:Magnitude Difference, Tasseled Cap, Color Difference and Band-Slope Difference. Firstly, in every algorithm the suitable transform of the images was performed. Then the techniques of normalized ratio and normalized difference were used in order to derivethe changes. All the algorithms were applied and evaluated at Ikonos images of years 2000 and 2007, in Attica region (Attiki Odos construction sites) and the results were satisfactory.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Με την συνεχόμενη ανάπτυξη της τεχνολογίας, αλλά και με την πληθώρα των δορυφορικών και εναέριων πληροφοριών, η ανάγκη για εύρεση τρόπων αυτοματοποίησης διαδικασιών που παλιότερα γίνονταν με Φωτοερμηνεία είναι μεγάλη. Τα μέσα που διαθέτει ο σημερινός Φωτοερμηνευτής είναι πάρα πολλά και συνεχώς αυξάνονται και εξελίσσονται. Με την είσοδο της τεχνολογίας στο αντικείμενο της Φωτοερμηνείας-Τηλεπισκόπησης, υπάρχει μια μεγάλη βελτίωση στην ακρίβεια των αποτελεσμάτων που παράγονται αλλά και μια μεγάλη μείωση στον απαιτούμενο χρόνο και κόστος προκειμένου αυτά να παραχθούν.

Υπάρχει μεγάλο πλήθος συστημάτων επεξεργασίας ψηφιακών απεικονίσεων, το οποίο ανάλογα με την εφαρμογή και τα δεδομένα ο Φωτοερμηνευτής μπορεί να χρησιμοποιήσει για την επίλυση των διάφορων εφαρμογών του. Τα περισσότερα από τα συστήματα αυτά είναι κλειστά, δεν αφήνουν τον χρήστη να προσαρμόσει τις λειτουργίες τους σύμφωνα με τις ανάγκες του και σε κάθε περίπτωση δεν μπορεί να ασχοληθεί με τους αλγορίθμους που υπάρχουν σε αυτά με αποτέλεσμα να μην γνωρίζει σε κάποιες περιπτώσεις τις παραμέτρους που χρησιμοποιούνται σε μερικά στοιχεία τους για να κρίνει το τελικό αποτέλεσμα. Πολλοί Φωτερμηνευτές λοιπόν, απευθύνονται στο ελεύθερο λογισμικό και όλα τα διαθέσιμα εργαλεία που υπάρχουν σε αυτό.

Στο σημείο αυτό κρίνεται σκόπιμο να οριστεί το ελεύθερο λογισμικό και να προσδιοριστούν οι αρχές και τα πλεονεκτήματά του. Οι αρχές λοιπόν του ελεύθερου λογισμικού, οι οποίες το ορίζουν παράλληλα, είναι τέσσερις:

- Μπορεί να χρησιμοποιηθεί ελεύθερα για οποιαδήποτε χρήση.
- Όλος ο κώδικάς του είναι διαθέσιμος και μπορεί να μελετηθεί ελεύθερα.
- Μπορεί να διανεμηθεί ελεύθερα καθώς είναι ελεύθερη η αντιγραφή του.
- Τέλος μπορεί να τροποποιηθεί ελεύθερα από οποιονδήποτε. Με αυτό τον τρόπο επιτυγχάνεται η βελτίωσή του.

Πολλοί είναι αυτοί που νομίζουν πως ελεύθερο λογισμικό είναι αυτό που διανέμεται δωρεάν. Η συγκεκριμένη αντίληψη όμως είναι λάθος καθώς υπάρχουν περιπτώσεις στις οποίες η απόκτηση του ελεύθερου λογισμικού ή η συντήρησή του κοστίζει. Παρόλα αυτά τα πλεονεκτήματα του ελεύθερου λογισμικού είναι πάρα πολλά. Αρχικά ήδη υπάρχει πληθώρα σχετικού λογισμικού διαθέσιμη και μπορεί να χρησιμοποιηθεί από οποιονδήποτε, έπειτα μπορεί να γίνει πολύ εύκολα μεταφορά και εξάπλωση της τεχνογνωσίας καθώς όλος ο κώδικας είναι ανοιχτός σε όποιον έχει την διάθεση να τον διαβάσει. Επίσης ένα άλλο πλεονέκτημα του ελεύθερου λογισμικού

είναι το γεγονός πως με την απόκτηση ενός ορισμένου επιπέδου γνώσης, ο καθένας μπορεί να τροποποιήσει τον κώδικα σύμφωνα με τις ανάγκες και τις προτιμήσεις του, ενώ παράλληλα μπορεί να ελέγξει τα αποτελέσματά του καθώς αυτός είναι υπεύθυνος για αυτά. Τέλος, στο ελεύθερο λογισμικό πολύ σημαντική θέση έχει η κοινότητα η οποία το υποστηρίζει. Με την είσοδο του κάθε χρήστη στο ελεύθερο λογισμικό, αυτομάτως υπάρχει και η αντίστοιχη κοινότητα η οποία τον υποστηρίζει και τον βοηθάει σε τυχόν προβλήματα και απορίες του. Με την σειρά του, όταν αυτός θα γίνει πιο έμπειρος θα συνεχίσει την αλυσίδα, λύνοντας απορίες των νεώτερων χρηστών.

Η παρούσα εργασία όπως ήδη αναφέρθηκε, ασχολείται με την παρουσίαση της βιβλιοθήκης του orfeo toolbox, την παρουσίαση των δυνατοτήτων του και την προσπάθεια εμπλουτισμού της σε αλγορίθμους ανίχνευσης μεταβολών. Το orfeo toolbox αποτελεί μια βιβλιοθήκη ελεύθερου κώδικα, γραμμένη σε γλώσσα C++ και διέπεται από όλες τις αρχές του ελεύθερου λογισμικού. Η βιβλιοθήκη παρέχεται στους χρήστες της χωρίς κόστος και όλος ο κώδικάς της είναι ορατός σε αυτούς. Ο κάθε χρήστης είναι υπεύθυνος για τον τρόπο που θα δομήσει το πρόγραμμα και τα αποτελέσματα που θα παραχθούν από αυτό. Τέλος, υπάρχει και η αντίστοιχη κοινότητα που υποστηρίζει την βιβλιοθήκη. Μέσω μιας λίστας ηλεκτρονικού ταχυδρομείου που διαθέτει, κάθε χρήστης μπορεί να απευθύνεται στην κοινότητα και να λύνει τυχόν προβλήματα και απορίες πάνω στην βιβλιοθήκη. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

1.2 Στόχος της εργασίας

Μέσα από την συγκεκριμένη εργασία, έγινε μια προσπάθεια να παρουσιαστεί το εργαλείο τηλεπισκόπησης orfeo toolbox καθώς επίσης και η χρήση αλγορίθμων υλοποιημένων σε C++ που αφορούν την ανίχνευση μεταβολών σε δορυφορικές πολυφασματικές εικόνες. Μετά το πέρας της εργασίας, στόχος είναι η σωστή χρήση του συγκεκριμένου εργαλείου τηλεπισκόπησης αλλά και η συνεισφορά στην κοινότητα του orfeo αλγορίθμων που να χρησιμοποιούνται από όλους τους χρήστες.

Η ενασχόληση με το συγκεκριμένο εργαλείο τηλεπισκόπησης προέκυψε καθώς είναι ένα τελείως καινούργιο εργαλείο με πολλές δυνατότητες. Ο κώδικάς του είναι ελεύθερος και ανοιχτός, οπότε δίνει την δυνατότητα να μελετηθεί αλλά και να χρησιμοποιηθεί με όποιο στόχο κρίνεται σκόπιμο στα πλαίσια της συγκεκριμένης εργασίας. Αποτελεί με άλλα λόγια ένα εργαλείο άξιο ενδιαφέροντος καθώς δίνει την δυνατότητα σε κάθε χρήστη να συμμετέχει στην ανάπτυξή του. Έτσι και αλλιώς έχει προκαλέσει το ενδιαφέρον πολλών ερευνητών αλλά και γενικότερα χρηστών από πανεπιστήμια, ερευνητικούς οργανισμούς και εταιρίες σε πολλά μέρη του κόσμου.

Πέρα από μια γενική προσέγγιση στο περιβάλλον του OTB, το θέμα της συγκεκριμένης εργασίας επικεντρώνεται στην ανίχνευση μεταβολών σε δορυφορικές εικόνες. Ανίχνευση μεταβολών είναι η διαδικασία αναγνώρισης διαφορών σε αντικείμενα και φαινόμενα, που παρατηρούνται μέσα από διαχρονικές εικόνες της ίδιας περιοχής. Η βιβλιογραφία για το συγκεκριμένο θέμα είναι πολύ μεγάλη καθώς διάφορες μέθοδοι έχουν αναπτυχθεί για το σκοπό αυτό. Οι αλγόριθμοι που αναπτύσσονται στα πλαίσια της εργασίας απευθύνονται σε εικόνες υψηλής χωρικής ακρίβειας, καθώς οι εικόνες που χρησιμοποιούνται ανήκουν στον δορυφόρο Ikonos. Οι αλγόριθμοι έχουν ως βάση τα pixel της εικόνας και μέσα από διάφορες διαδικασίες παράγουν μια εικόνα που περιέχει την διαφορά που υπάρχει στην περιοχή μεταξύ των δύο διαφορετικών εποχών.

1.3 Οργάνωση της εργασίας

Η οργάνωση της παρούσας διπλωματικής γίνεται σε δύο βασικά στάδια. Στο πρώτο στάδιο γίνεται η ανάλυση της βιβλιοθήκης (OTB) αλλά και του γραφικού περιβάλλοντος αυτής (Monteverdi), με ανάλυση όλων των διατιθέμενων εργαλείων και υλοποίηση βασικών εφαρμογών τηλεπισκόπησης έτσι ώστε κάθε νέος χρήστης να μπορεί να απευθυνθεί σε αυτή για εύρεση κάθε πληροφορίας και επίλυση κάθε προβλήματος. Στο δεύτερο στάδιο, έχοντας αποκτήσει μια βάση γνώσης για τις δυνατότητες και τον τρόπο δόμησης των προγραμμάτων στην συγκεκριμένη βιβλιοθήκη, έγινε προσπάθεια για δημιουργία μια σειράς από αλγορίθμους ανίχνευσης μεταβολών για εικόνες υψηλής ευκρίνειας.

Αναλυτικότερα, το πρώτο μέρος της εργασίας έχει ως σκοπό την ανάλυση ενός τελείως καινούργιου εργαλείου με πολλές σημαντικές εφαρμογές τηλεπισκόπησης. Έτσι λοιπόν δημιουργήθηκε ένα εγχειρίδιο που να εισάγει κάθε χρήστη στην βιβλιοθήκη αλλά και το γραφικό της περιβάλλον (κεφάλαια 3 και 4). Στο συγκεκριμένο πλαίσιο, εκτός από την παρουσίαση όλων των εργαλείων του Monteverdi αναλυτικά, ακολούθησε η εφαρμογή κάποιον αλγορίθμων τηλεπισκόπησης σε δορυφορικές εικόνες Quickbird που απεικονίζουν την περιοχή της Αχλάδας του Ηρακλείου. Τα αποτελέσματα ήταν άκρως ενθαρρυντικά, τόσο από άποψη αποτελεσμάτων όσο και από άποψη ταχύτητας επεξεργασίας εικόνων μεγάλων διαστάσεων.Στην συνέχεια, οι ίδιες εφαρμογές πραγματοποιήθηκαν και μέσω της χρήσης της βιβλιοθήκης του OTB. Στο αντίστοιχο κεφάλαιο, αναλύθηκε ο κώδικας των αλγορίθμων έτσι ώστε να γίνει κατανοητός ο τρόπος που δουλεύει η βιβλιοθήκη.

Στην συνέχεια και σε δεύτερο στάδιο αναλύθηκαν οι αλγόριθμοι ανίχνευσης μεταβολών που υπάρχουν ήδη στην βιβλιοθήκη (κεφάλαιο 6). Στο σημείο αυτό παρατηρήθηκε ένα πρόβλημα καθώς οι αλγόριθμοι που υπήρχαν μέσα στο πρόγραμμα απευθύνονταν μόνο σε εικόνες png και δούλευαν μόνο για ένα κανάλι. Παρουσιάστηκε λοιπόν η ανάγκη δημιουργίας αλγορίθμων που να επεξεργάζονται δορυφορικές πολυφασματικές εικόνες (κεφάλαιο 7). Έτσι οι αλγόριθμοι που επιλέχθηκαν να υλοποιηθούν ήταν οι εξής:

- Αλγόριθμος διαφοράς έντασης
- Αλγόριθμος Tasseled Cap
- Αλγόριθμος διαχωρισμού Κυρίων Χρωμάτων
- Αλγόριθμος διαφορών κλίσεων

Στο πλαίσιο της συγκεκριμένης εργασίας οι παραπάνω αλγόριθμοι υλοποιήθηκαν σε γλώσσα C++ προκειμένου να ενσωματωθούν στην βιβλιοθήκη.Τέλος, τα δεδομένα που χρησιμοποιήθηκαν ήταν δύο εικόνες Ikonos ενός τμήματος της Αττικής οδού, πριν και μετά την κατασκευή της.

Κεφάλαιο 2

Ανασκόπηση βιβλιογραφίας

2.1 Γενικά για τηλεπισκόπηση

Τηλεπισκόπηση είναι η επιστήμη και τεχνική, που ασχολείται με τις αρχές, τις αναλογίες και ψηφιακές μεθόδους και τα όργανα, με τα οποία επιτυγχάνεται από μακρυά, η συλλογή, επεξεργασία, και ανάλυση, πλήθους ποιοτικών και μετρητικών πληροφοριών, για τη γη, τους ωκεανούς, την ατμόσφαιρα και το φυσικό και το κοινωνικοοικονομικό περιβάλλον γενικότερα, (αλλά και για τις σχέσεις, τις αλληλεξαρτήσεις και τις αλληλεπιδράσεις τους και τις τάσεις μεταβολής τους δια μέσου του χρόνου), καθώς επίσης και για οποιοδήποτε αντικείμενο, φαινόμενο, γεγονός και συμβάν, ή και για οποιαδήποτε διαδικασία μεταβολής τους. Η Τηλεπισκόπηση αξιοποιεί ¨απεικονίσεις' της πραγματικότητας στις περιοχές του ορατού φωτός, του υπερύθρου και των μικροκυμάτων, του φάσματος της Ηλεκτρομαγνητικής Ακτινοβολίας. [Ρόκος, 2005]

Η τηλεπισκόπηση χρησιμοποιεί οπτικές αλλά και ψηφιακές τεχνικές για την αναγνώριση και ερμηνεία αντικειμένων. Η πρώτη κατηγορία, δηλαδή οι οπτικές τεχνικές αναγνώρισης αντικειμένων στηρίζεται κυρίως στην ποιοτική φωτοερμηνεία πολυφασματικών εικόνων. Η δεύτερη κατηγορία οι ψηφιακές δηλαδή, τεχνικές αναγνώρισης αντικειμένων στηρίζεται κυρίως στην ποσοτική κατάταξη των εικονοστοιχείων μιας πολυφασματικής εικόνας σε συγκεκριμένες φασματικές κατηγορίες.Η κατηγορία αυτή, αποτελεί και το αντικείμενο της Ψηφιακής Τηλεπισκόπησης. [Αργιαλάς, 1998]

2.2 Γενικά για ψηφιακή τηλεπισκόπηση

Την σημερινή εποχή η επιστήμη της Ψηφιακής Τηλεπισκόπησης έχει αναπτυχθεί σε μεγάλο βαθμό, συνεργάζεται με άλλες επιστήμες όπως είναι η όραση υπολογιστών, τα συστήματα GIS και πολλές άλλες, ενώ οι μέθοδοι που υπάρχουν για την επεξεργασία εικόνων συνεχώς αυξάνονται και βελτιώνονται. Όλα αυτά οφείλονται φυσικά και στο γεγονός ότι και τα μέσα που συλλέγουν εικόνες βελτιώθηκαν σε μεγάλο βαθμό, πλήθυναν και βοήθησαν στην καλύτερη παρατήρηση της γήινης επιφάνειας. Οι δορυφόροι που παρέχουν δεδομένα πλέον είναι πάρα πολλοί, με πολύ καλή χωρική ακρίβεια και χαμηλό κόστος.

Τα πλεονεκτήματα της Ψηφιακής Τηλεπισκόπησης σε σχέση με την παραδοσιακή Φωτοερμηνεία -Τηλεπισκόπηση είναι πολλά και σημαντικά. Αρχικά, έχει πολύ μεγάλη ποσοτική ακρίβεια που προέρχεται από τη δυνατότητα του υπολογιστή να επεξεργάζεται τις εικόνες με βάση το δομικό στοιχείο του εικονοστοιχείου και με εύρος όλα τα υπάρχοντα πολυφασματικά δεδομένα που παρέχονται σε αυτές. Έπειτα, απαιτούν πολύ μικρή ανθρώπινη παρέμβαση και συμβολή, κάτι που εμποδίζει στην δημιουργία λαθών λόγω του ανθρώπινου παράγοντα. Τέλος, λόγω των αλγορίθμων που έχουν κατασκευαστεί, επιτυγχάνεται μεγάλη οικονομία χρόνου και κόστους. Με το ίδιο κόστος, οι περιοχές μελέτης που μπορούν να επεξεργαστούν στο ίδιο χρόνο ε-

ίναι πολύ μεγαλύτερες. Αυτά είναι κάποια από τα πλεονεκτήματα που διαθέτει η Ψηφιακή Τηλεπισκόπηση και παραδοσιακά εστιάζει στην επεξεργασία πολυφασματικών δεδομένων.

2.3 Περιγραφή δεδομένων για την Ψηφιακή Τηλεπισκόπηση

Όπως έχει αναφερθεί ήδη, η Ψηφιακή Τηλεπισκόπηση είναι η επιστήμη της παρατήρησης της επιφάνειας της γης. Η παρατήρηση αυτή επιτυγχάνεται μέσω της χρήσης ψηφιακών σαρωτών (τηλεπισκοπικών ανιχνευτών) που ανιχνεύουν την ανάκλαση της ηλεκτρομαγνητικής ακτινοβολίας της γήινης επιφάνειας και την αποδίδουν ως ψηφιακή εικόνα. Οι σαρωτές μπορεί να είναι εγκατεστημένοι σε τεχνητούς δορυφόρους που βρίσκονται σε τροχιά γύρω από τη γη ή να βρίσκονται σε αερομεταφερόμενα μέσα (αεροσκάφοι, ελικόπτερα). Ένα διαστημικό όχημα μπορεί να μεταφέρει περισσότερους από ένα ανιχνευτές, έτσι πολλές φορές προκαλείται σύγχυση μεταξύ οχήματος και σαρωτή. [http://el.wikipedia.org/wiki/, 2011]

Ανάλογα με την προέλευση της ηλεκτρομαγνητικής ακτινοβολίας, η οποία αντανακλάται και στη συνέχεια ανιχνεύεται, οι δέκτες - σαρωτές (ή αισθητήρες) μπορούν να διακριθούν σε παθητικούς και σε ενεργητικούς. Παθητικοί είναι εκείνοι που ανιχνεύουν ηλεκτρομαγνητική ακτινοβολία προερχόμενη από μία φυσική πηγή (συνήθως ο ήλιος), ενώ ενεργητικοί σαρωτές είναι εκείνοι που ΄φωτίζουν΄ (προσβάλλουν) οι ίδιοι το στόχο χρησιμοποιώντας την δική τους πηγή ακτινοβολίας, π.χ. εικονοληπτικά ραντάρ. Τα δεδομένα λοιπόν που χρησιμοποιούνται μπορούν να προέρχονται είτε από ενεργητικούς ανιχνευτές είτε από ραντάρ. Και οι δύο τύποι δεδομένων χρησιμοποιούνται ευρέως ανάλογα με την εφαρμογή κάθε φορά, απλά σε κάθε περίπτωση η μεθοδολογία προσέγγισης και οι αλγόριθμοι που χρησιμοποιούνται είναι διαφορετικοί. [http://el.wikipedia.org/wiki/, 2011]

Τα δεδομένα που μεταδίδονται από τους ανιχνευτές μετατρέπονται σε 2-διαστους πίνακες ν γραμμών και μ στηλών, όπου η τιμή του κάθε εικονοστοιχείου αντιπροσωπεύει την ανταναάκλαση της ηλεκτρομαγνητικής ακτινοβολίας. Γενικά τα τηλεπισκοπικά δεδομένα δηλαδή οι ψηφιακές εικόνες χαρακτηρίζονται από τις παρακάτω ιδιότητες:

- Χωρική ανάλυση: Είναι η χωρική διακριτική ικανότητα της εικόνας, δηλαδή το μέγεθος του pixel σε πραγματικές διαστάσεις. Στην πράξη η χωρική ανάλυση καθορίζει τις ελάχιστες διαστάσεις των αντικειμένων ικανών να αποτυπωθούν στη ψηφιακή εικόνα. Υψηλής χωρικής ανάλυσης εικόνες, διαθέσιμες για πολιτική χρήση, έχουν μέγεθος pixel από 15μ εώς και 0.6μ.

- Φασματική Ανάλυση: Αποτελεί το φασματικό εύρος κάθε φασματικού καναλιού. Εικόνες με μικρό φασματικό εύρος (π.χ. 10 νανόμετρα) είναι εικόνες υψηλής φασματικής ανάλυσης και προσφέρονται για λεπτομερή εξέταση και αναγνώριση των υλικών που αποτυπώνονται, χρησιμοποιώντας τεχνικές υπερφασματικής ανάλυσης.

- Ραδιομετρική Ανάλυση: Είναι η φασματική διακριτική ικανότητα, δηλαδή το πόσο εύκολα ή δύσκολα μπορούν να διακριθούν υλικά γειτονικής φασματικής συμπεριφοράς σε μια εικόνα.

2.4 Λογισμικά τηλεπισκόπησης

Με την όλο και μεγαλύτερη πληθώρα και βελτίωση των δεδομένων της τηλεπισκόπησης, δημιουργείται η όλο και μεγαλύτερη ανάγκη για λογισμικά που να επεξεργάζονται τα δεδομένα που παρέχονται. Τα περισσότερα από τα λογισμικά αυτά είναι κλειστά και σε τακτά χρονικά

διαστήματα αλλάζουν εκδόσεις, δηλαδή αυξάνουν και τις δυνατότητές τους. Παρακάτω παρουσιάζονται κάποια από τα γνωστότερα λογισμικά τηλεπισκόπησης και οι δυνατότητές τους. Η διάκριση των λογισμικών αυτών, έγινε σε δύο επίπεδα. Το πρώτο επίπεδο αφορούσε το εύρος των τηλεπισκοπικών τους εφαρμογών και έτσι διακρίθηκαν τα κλασικά και τα εξειδικευμένα λογισμικά, ενώ το δεύτερο επίπεδο της διάκρισής τους έγινε με βάση το αν είναι εμπορικά ή ελεύθερα και αν έχουν ως βάση το εικονοστοιχείο ή το αντικείμενο.

2.4.1 Κλασικά λογισμικά τηλεπισκόπησης

- **ERDAS Imagine**
  Το λογισμικό ERDAS Imagine αποτελεί ένα κλειστό λογισμικό τηλεπισκόπησης που σχεδιάστηκε για γεωχωρικές εφαρμογές. Η πρώτη επίσημη έκδοσή του έγινε το 1978, ενώ η τελευταία είναι αυτή του 2010 και αποτελεί την 10.1. Το λογισμικό αυτό αποτελείται από επιμέρους ανεξάρτητα τμήματα, ενώ αυτό που χρησιμοποιείται κυρίως στην τηλεπισκόπηση είναι το ERDAS Imagine. Το ERDAS αποτελεί ένα εργαλείο που επιτρέπει στον χρήστη να πραγματοποιήσει αρκετές εφαρμογές σε μια εικόνα και να απαντήσει σε συγκεκριμένες γεωχωρικές ερωτήσεις. [http://www.erdas.com/Homepage.aspx, 2011]

  Χρησιμοποιώντας τις τιμές των δεδομένων και την θέση τους, έχει την ικανότητα να εξάγει αντικείμενα που δεν είναι ορατά και να να προσδιορίζει την γεωγραφική τους θέση. Οι εφαρμογές του συγκεκριμένου λογισμικού είναι πολλές και τα αποτελέσματά του πολύ καλά. Επίσης μπορεί να χρησιμοποιήσει μια ποικιλία από format εικόνων τόσο οπτικών όσο και radar. Τέλος, διαθέτει και αυτό την δική του κοινότητα για όλους τους χρήστες. Οι εφαρμογές του, αγγίζουν όλο το εύρος της τηλεπισκόπησης, για το λόγο αυτό κατατάσσεται και στα κλασικά λογισμικά.

- **ENVI** Το κλειστό λογισμικό ENVI (ENvironment for Visualizing Images) χρησιμοποιείται για εφαρμογές και ανάλυση γεωχωρικών δεδομένων. Το λογισμικό διαθέτει δυνατότητες επεξεργασίας προχωρημένων εφαρμογών που αφορούν τα γεωχωρικά δεδομένα και μπορεί να επεξεργαστεί εικόνες σε όλα τα επίπεδα γεωπληροφορίας. Με άλλα λόγια, παρέχει εργαλεία τα οποία βοηθούν τον χρήστη να εξάγει την πληροφορία που τον ενδιαφέρει ανεξάρτητα από την εφαρμογή. Ο χρήστης μπορεί να εφαρμόσει όλες τις ενέργειές του μέσω ενός φιλικού περιβάλλοντος που είναι συμβατό με όλους τους τύπους δεδομένων. [http://www.ittvis.com/language/en-us/productsservices/envi.aspx, 2011]

  Το ENVI κατασκευάστηκε από την εταιρία U.S. Geological Surney και ήταν κατασκευασμένο σε περιβάλλον DOS. Σήμερα η εξελιγμένη του μορφή, βρίσκεται στην έκδοση 4.8. Όπως σε όλα τα λογισμικά διαθέτει και την δική του κοινότητα για παροχή απαντήσεων στους χρήστες του.

- **ERMapper** Το κλειστό λογισμικό ERMapper παρέχει ένα σύστημα συμπίεσης πολυφασματικών εικόνων, η συμπίεση μπορεί να φτάσει από το 2% μέχρι το 5% της αρχικής διάστασης της εικόνας. Με τον τρόπο αυτό εξοικονομείται πολύ χώρος αποθήκευσης δεδομένων. Εκτός όμως από τις μετατροπές στις πολυφασματικές εικόνες, το λογισμικό μπορεί να πραγματοποιήσει πληθώρα τηλεπισκοπικών εφαρμογών χάρης στην ποικιλία των αλγορίθμων που περιέχει.

  Σήμερα το λογισμικό αυτό ανήκει στην εταιρία ERDAS, ενώ η τελευταία έκδοσή του είναι η 7.2.

- **Grass** Το λογισμικό GRASS (Geographic Resources Analysis Support System) α-

ποτελεί ένα ελεύθερο GIS λογισμικό που χρησιμοποιείται για επεξεργασία γεωχωρικών δεδομένων. Το συγκεκριμένο λογισμικό είναι ένας συνδιασμός επεξεργασίας raster/vector δεδομένων, ανάλυσης εικόνας και απεικόνισης δεδομένων. Περιέχει ένα μεγάλο πλήθος μοντέλων για διαχείριση, επεξεργασία και απεικόνιση των γεωχωρικών δεδομένων. [http://grass.fbk.eu/, 2011]

Ανεξάρτητα με όλα τα υπάρχοντα λογισμικά GIS το GRASS παρέχει εξολοκλήρου πρόσβαση στους αλγορίθμους του, δίνοντας την δυνατότητα στους χρήστες να γράψουν τα δικά του GIS μοντέλα είτε μελετώντας τα ήδη υπάρχοντα είτε διαβάζοντας το αντίστοιχο εγχειρίδιο του λογισμικού που υπάρχει. Η χρήση του λογισμικού είναι μεγάλη και για το λόγο αυτό υπάρχει και η αντίστοιχη κοινότητα που το υποστηρίζει, αλλά παράλληλα υπάρχουν και εταιρίες που παρέχουν επαγγελματική υποστήριξη του λογισμικού. Το λογισμικό είναι γραμμένο σε C, C++, Python, Tcl, η σημερινή έκδοσή του είναι η 6.4.1, ενώ η άδειά του είναι η GNU General Public License.

- **Orfeo ToolBox**
  Το OTB αποτελεί όπως έχει αναφερθεί, μια βιβλιοθήκη, η οποία παρέχει όλους τους βασικούς αλγορίθμους για εργασίες τηλεπισκόπησης. Η ανάπτυξη της βιβλιοθήκης συνεχώς εξελίσσεται, παρέχοντας στον χρήστη όλο και μεγαλύτερη γκάμα εφαρμογών, πολλές φορές και πιο εξειδικευμένων. Δεν κρίνεται σκόπιμο να αναφερθεί κάτι παραπάνω καθώς η συγκεκριμένη βιβλιοθήκη αναπτύσσεται σε μεγάλο βαθμό σε όλο το εύρος της διπλωματικής.

2.4.2 Εξειδικευμένα λογισμικά τηλεπισκόπησης

- **eCognition**
  Το λογισμικό eCognition αποτελεί ένα από τα γνωστότερα και με πολλές εφαρμογές κλειστά λογισμικά τηλεπισκόπησης. Αποτελεί ένα λογισμικό που στηρίζεται στην αντικειμενοστραφή ανάλυση. Η πρώτη επίσημη έκδοσή του έγινε το Μάιο του 2000 και σήμερα βρίσκεται στην έκδοση 8.64. Παρουσιάζει μια υπολογιστική μέθοδο για εξαγωγή πληροφορίας από εικόνες, χρησιμοποιώντας ιεραρχίες από αντικείμενα (γκρούπ από εικονοστοιχεία) κάτι που ήταν πολύ διαφορετικό από τον παραδοσιακό τρόπο επεξεργασίας εικόνων.Ήταν πρωτοπόρο στην επεξεργασία 64bit εικόνων. [http://www.ecognition.com, 2011]

  Προσπαθώντας να μιμηθεί το ανθρώπινο μυαλό, το συγκεκριμένο λογισμικό χρησιμοποιεί διαδικασίες κατάτμησης και ταξινόμησης εξετάζοντας τα εικονοστοιχεία όχι μεμονωμένα αλλά σαν ομάδα. Με τον τρόπο αυτό κατασκευάζει αντικείμενα χρησιμοποιώντας κάποια χαρακτηριστικά όπως χρώμα, μέγεθος, σχήμα, υφή αλλά και τις σχέσεις μεταξύ των αντικειμένων γενικότερα. Οι ιδιότητες αυτές είναι πάρα πολλές και δίνουν την δυνατότητα στον χρήστη να εξάγει οποιαδήποτε πληροφορία με μεμονωμένη αλλά και συνδυασμένη χρήση τους. Τέλος, για το συγκεκριμένο λογισμικό, υπάρχει και η αντίστοιχη κοινότητα στην οποία μπορεί να απευθυνθεί ο χρήστης για οποιαδήποτε πληροφορία.

- **Erdas DeltaCue**
  Το συγκεκριμένο εργαλείο, αποτελούσε κομμάτι του Erdas Imageine, απλά παρεχόταν ξεχωριστά στον χρήστη. Το εργαλείο αυτό λοιπόν, χρησιμοποιήθηκε για εφαρμογές ανίχνευσης μεταβολών. Παρείχε δηλαδή μια σειρά αλγορίθμους και τα αντίστοιχα εργαλεία που χρειαζόταν ο χρήστης για εφαρμογές ανίχνευσης μεταβολών.

- **Erdas Feature Analyst**
  Το συγκεκριμένο πακέτο, αποτελεί ένα κομμάτι του Erdas Imagine, με την διαφορά ότι

παρέχεται ξεχωριστά στον χρήστη. Χρησιμοποιείται για εφαρμογές που χρησιμοποιούν το αντικείμενο ως δομικό στοιχείο. Οι εφαρμογές του δηλαδή είναι αρκετά περιορισμένες σε σχέση με το εύρος των τηλεπισκοπικών εφαρμογών.

- **Erdas Objective**
  Το συγκεκριμένο εργαλείο, αποτέλεσε ξεχωριστό πακέτο του Erdas Imagine, το οποίο διαθέτει μια σειρά από εργαλεία, που δημιουργούν μια σειρά από γεωχωρικά layers σε τηλεπισκοπικές απεικονίσεις. Συνδυάζει την επαγωγική γνώση με την γνώση των expert, σε ένα περιβάλλον καθαρά αντικειμενοστραφούς ανάλυσης εικόνας.

- **ENVI Feature Extraction and Zoom**
  Όπως και παραπάνω, το πακέτο αυτό, αποτελεί ένα κομμάτι του ENVI, το οποίο παρέχεται ξεχωριστά από το βασικό πακέτο. Η λειτουργία του είναι η εξαγωγή και επεξεργασία αντικειμένων από την αρχική εικόνα. Με λίγα λόγια, εξάγει διάφορα επίπεδα πληροφορίας από τις εικόνες, και με βάση αυτά γίνεται η όποια περαιτέρω επεξεργασία πάνω σε αυτές.

- **Matlab**
  Το λογισμικό Matlab, αποτελεί μια υψηλού επιπέδου γλώσσα και διαδραστικό περιβάλλον, το οποίο βοηθάει στην επίλυση μεγάλου εύρους εφαρμογών. Το συγκεκριμένο περιβάλλον, χρησιμοποιείται από πολλούς κλάδους, ένας από τους οποίους είναι η τηλεπισκόπηση. Παρέχει πολλά εργαλεία για επεξεργασία εικόνων, συνήθως μικρών διαστάσεων, τα οποία όμως είναι αρκετά εξειδικευμένα, καθώς δεν παρέχουν όλο το εύρος των τηλεπισκοπικών εφαρμογών. Για το λόγο αυτό και κατατάσσεται σε αυτή την κατηγορία.

- **QGis**
  Αποτελεί ένα ανοιχτό λογισμικό επεξεργασίας εικόνων, που υποστηρίζεται από τις περισσότερες πλατφόρμες συστημάτων. Οι εφαρμογές του δεν αφορούν μόνο τηλεπισκόπηση, αλλά και εφαρμογές GIS. Για το λόγο αυτό κατατάσσεται στα εξειδικευμένα λογισμικά τηλεπισκόπησης.

2.4.3 Διαχωρισμός εμπορικών και ελεύθερων λογισμικών

Στο σημείο αυτό, κρίνεται σκόπιμο, να δημιουργηθεί ένας πίνακας που να ξεχωρίζει τα εμπορικά, ιδιωτικά λογισμικά τηλεπισκόπησης, από τα ελεύθερα, ανοιχτά. Η διάκριση αυτή έχει προκύψει λίγο πολύ από το παραπάνω κεφάλαιο.

Λειτουργικό σύστημα	**Εμπορικό λογισμικό**	**Ελεύθερο λογισμικό**
eCognition	✓	
ERMapper	✓	
ERDAS Imagine	✓	
ERDAS DeltaCue	✓	
ERDAS Feature Analyst	✓	
ERDAS Objective	✓	
ENVI	✓	
ENVI Feature Extraction and Zoom	✓	
Grass		✓
Συνεχίζεται στην επόμενη σελίδα		

Πίνακας 2.1 – συνέχεια από προηγούμενη σελίδα

Λειτουργικό σύστημα	**Εμπορικό λογισμικό**	**Ελεύθερο λογισμικό**
QGIS		✓
OTB		✓

Πίνακας 2.1: Διάκριση λογισμικών σε ελεύθερα και εμπορικά

2.4.4 Διαχωρισμός ανάλογα με το δομικό στοιχείο

Παλαιότερα, όλες οι εφαρμογές τηλεπισκόπησης, γίνονταν με δομικό στοιχείο το εικονοστοιχείο της δορυφορικής εικόνας. Σήμερα, οι μέθοδοι αυτοί έχουν εξελιχθεί, χρησιμοποιώντας πολλές φορές ως δομικό στοιχείο το αντικείμενο, δηλαδή μια συστάδα εικονοστοιχείων. Έτσι λοιπόν, δημιουργήθηκε η Αντικειμενοστραφής Ανάλυση εικόνας, λαμβάνει υπόψη της την τηλεπισκοπική απεικόνιση σαν ένα σύνολο ομοιογενών περιοχών-αντικειμένων.

Η διαδικασία λοιπόν με την οποία πραγματοποιείται η αντικειμενοστραφής προσέγγιση είναι η ακόλουθη. Στην αρχική εικόνα, εφαρμόζονται διαδικασίες κατάτμησης, με τις οποίες η εικόνα μετατρέπεται σε ένα σύνολο πρωτογενών αντικειμένων. Οι διαδικασίες κατάτμησης είναι διαδικασίες χαμηλού ή μέσου επιπέδου που δεν χρησιμοποιούν γνώση. Οι αλγόριθμοι κατατμησης εικόνων, γενικά δεν αποσκοπούν να εντοπίσουν και να οριοθετήσουν σημασιολογικά αντικείμενα, αλλά να παράγουν πρωτογενή αντικείμενα τα οποία στη συνέχεια θα υποστούν διαδικασίες επεξεργασίας και ταξινόμησης από "έξυπνους" αλγορίθμους ή από συστήματα στηριγμένα στη γνώση. [Αργιαλάς Δ. και Τζώτσος Α., 2011]

Έτσι λοιπόν, δημιουργήθηκε ένας πίνακας που διαχωρίζει τα λογισμικά ανάλογα με το πιο δομικό στοιχείο χρησιμοποιούν για την εφαρμογή των αλγορίθμων τους.

Λειτουργικό σύστημα	**Pixel Based**	**Object based**	**Mixed**
eCognition		✓	
ERMapper	✓		
ERDAS DeltaCue	✓		
ERDAS Feature Analyst		✓	
ERDAS Objective		✓	
ENVI			✓
ENVI Feature Extraction and Zoom		✓	
Grass	✓		
QGIS		✓	
OTB			✓

Πίνακας 2.2: Διάκριση λογισμικών ανάλογα με το δομικό τους στοιχείο

2.5 Τηλεπισκοπικοί Δέκτες

Στο συγκεκριμένο κεφάλαιο αναλύονται οι τηλεπισκοπικοί δέκτες που χρησιμοποιούνται για την παραγωγή τηλεπισκοπικών δεδομένων. Το σύνολο των τηλεπισκοπικών δεκτών μπορεί να αναλυθεί σε δύο είδη: τους ενεργητικούς δέκτες και τους οπτικούς ή παθητικούς δέκτες. Η διάκριση αυτή προέρχεται από το είδος της ηλεκτρομαγνητικής ακτινοβολίας, η οποία ανακλάται και στη συνέχεια ανιχνεύεται από τους δέκτες. Παθητικοί είναι οι δέκτες που ανιχνεύουν την ηλεκτρομαγνητική ακτινοβολία προερχόμενη από μια φυσική πηγή (συνήθως ο ήλιος), ενώ ενεργητικοί δέκτες είναι εκείνοι που φωτίζουν (προσβάλλουν) οι ίδιοι το στόχο χρησιμοποιώντας την δική του πηγή ακτινοβολίας, π.χ.εικονοληπτικά ραντάρ.

2.5.1 Οπτικοί δέκτες

Όπως αναφέρθηκε και παραπάνω, οπτικοί ή παθητικοί δέκτες είναι αυτοί που λαμβάνουν και καταγράφουν την ανακλώμενη ηλεκτρομαγνητική ακτινοβολία ή την εκπεμπόμενη θερμότητα. Ευαισθητοποιούνται στο τμήμα του φάσματος, το οποίο εκτείνεται από την περιοχή των πολύ μικρών μηκών κύματος (μικρότερα των 4μm) της υπεριώδους ακτινοβολίας, εώς την περιοχή του μήκους κύματος των 1.000μm. Ανάλογα δε με την εφαρμογή στην οποία θα αξιοποιηθούν απεικονίσεις από οπτικό δέκτη, επιλέγεται και η αντίστοιχη φασματική περιοχή. Οι δέκτες αυτοί τοποθετούνται πάνω σε πλατφόρμες και έτσι δημιουργήθηκαν δύο είδη απεικονίσεων για την τηλεπισκόπηση: οι αεροφωτογραφίες και οι δορυφορικές εικόνες. Μάλιστα, οι δορυφόροι αποτελούν τις ιδανικότερες πλατφόρμες για την τοποθέτηση των τηλεπισκοπικών δεκτών καθώς διαθέτουν προκαθορισμένη τροχιά, η διάρκεια πτήσης τους είναι πολύ μεγαλύτερη και επιτρέπει μια συνεχή παρακολούθηση της γης. Παρακάτω αναλύονται οι δορυφορικοί τηλεπισκοπικοί δέκτες καθώς αυτοί είναι που χρησιμοποιούνται κυρίως στις εφαρμογές τηλεπισκόπησης.

Οι οπτικοί δορυφορικοί τηλεπισκοπικοί δέκτες (οι δυνατότητες των οποίων περιορίζονται από δυσμενείς καιρικές συνθήκες), έχουν ως κύρια αποστολή την παρατήρηση και συστηματική παρακολούθηση της φυσικής γήινης επιφάνειας όπως αυτή διαμορφώνεται από φυσικές και ανθρωπογενείς παρεμβάσεις. Είναι δέκτες παγχρωματικοί και πολυφασματικοί, με δυνατότητες ταυτόχρονης παρατήρησης στο ορατό αλλά και σε τμήματα του υπερύθρου του φάσματος της ηλεκτρομαγνητικής ακτινοβολίας. Λειτουργούν σε φασματικά παράθυρα ώστε να επιτυγχάνεται η βέλτιστη δυνατή ευκρίνεια. Η τυπική διακριτική τους ικανότητα κυμαίνεται από 0.60 εώς 100 μέτρα ενώ το δυνατό πλάτος αποτύπωσης κυμαίνεται από 11 εώς 180 χιλιόμετρα. [http://www.gisfun.50megs.com/Tilepiskopisi.html, 2011]

Σήμερα υπάρχει μια πληθώρα δορυφορικών προγραμμάτων τηλεπισκόπησης και κατ΄ επέκταση και δεκτών. Κάποια από τα ποιο γνωστά προγράμματα που υπάρχουν σήμερα είναι οι LandsatTM, LandsatMSS, Spot, Modis, Aster, Ikonos, ο QuickBird και πολλοί άλλοι που χρησιμοποιούνται και παρέχουν πληθώρα τηλεπισκοπικών δεδομένων. Με τα προγράμματα αυτά δεν επιτυγχάνεται μόνο η συνεχής παροχή τηλεπισκοπικών δεδομένων μέσα σε μικρό χρονικό διάστημα και με χαμηλό κόστος, αλλά παράλληλα παρέχεται και μεγάλη διακριτική ικανότητα για μεγάλες περιοχές απεικόνισης. Η πολύ καλή χωρική διακριτική ικανότητα των δεδομένων αυτών δίνει την δυνατότητα επεξεργασίας τους με πληθώρα αλγορίθμων των οποίων τα αποτελέσματα έχουν μεγάλη ακρίβεια.

2.5.2 Ενεργητικοί δέκτες

Το μήκος της ακτινοβολίας που λειτουργούν οι συγκεκριμένοι δέκτες είναι μεταξύ 1mm-1m.Παρά την εξαιρετική αναλυτικότητα των εναέριων φωτογραφιών, τα μικροκυματικά αισθητήρια παρουσιάζουν μια σειρά από πλεονεκτήματα που τα καθιστούν χρήσιμα για μια σειρά από

εφαρμογές. Ο σημαντικότερος λόγος για την χρήση μικροκυμάτων στην τηλεπισκόπηση είναι ότι διαπερνούν τα σύννεφα και ως ένα βαθμό και την βροχή, ενώ επιπλέον είναι ανεξάρτητα της ύπαρξης ηλιακού φωτός πράγμα που τα καθιστά ικανά να χρησιμοποιηθούν και νύχτα. Επίσης, ένα σημαντικό πλεονέκτημα των ενεργητικών δεκτών, αποτελεί το γεγονός ότι οι ιδιότητες της εκπεμπόμενης και λαμβανόμενης ακτινοβολίας (όπως η ισχύς, η συχνότητα και η πόλωση) μπορούν να προκαθοριστούν ανάλογα με το σκοπό της χρήσης των απεικονίσεων.

Υπάρχουν πολλά είδη ενεργητικών τηλεπισκοπικών δεκτών και συστημάτων που παρέχουν πληθώρα δεδομένων στους χρήστες. Το σύστημα SAR (Synthetic Aperture Radar) ή αλλιώς ραντάρ συνθετικού ανοίγματος είναι ένα από αυτά. Αποτελεί την εξελιγμένη μορφή του SLAR, διαθέτει μια σταθερή κεραία που σαρώνει μια λωρίδα γήινης επιφάνειας εκπέμποντας πλευρικά της πλατφόρμας μικροκύματα, ενώ στην συνέχεια δέχεται το τμήμα της ακτινοβολίας που ανακλάται από το αντικείμενο και καταγράφει την ένταση του σήματος και το χρόνο επιστροφής του. Επίσης μια βελτίωση σε σχέση με το SLAR αποτελεί το γεγονός ότι πετυχαίνει κατά τεχνητό τρόπο συνθετικό μήκος-διάφραγμα κεραίας πολύ μεγαλύτερο του φυσικού και έτσι βελτιώνει σημαντικά την διαμήκη αναλυτικότητα του SLAR. Αυτό πετυχαίνεται με σύμφωνη αποθήκευση της επιστροφής από τον ίδιο στόχο, σε διαφορετικές χρονικές στιγμές. [http://www.gisfun.50megs.com/Tilepiskopisi.html, 2011]

Ένα άλλο επίσης πολύ γνωστό σύστημα ενεργών τηλεπισκοπικών δεκτών είναι το LIDAR (LIght Detection And Ranging. Ο συγκεκριμένος δέκτης εκπέμπει χιλιάδες παλμούς lazer το δευτερόλεπτο, οι οποίοι ανακλώνται από το αντικείμενο, επιστρέφουν στον δέκτη και ο χρόνος της επιστροφής μετράται με χρονόμετρο ακριβείας και μετατρέπεται σε απόσταση. Στην Ελλάδα η συγκεκριμένη τεχνική χρησιμοποιείται για τη μέτρηση της κατακόρυφης κατανομής των οπτικών ιδιοτήτων των αερολυμάτων, του όζοντος, αλλά και των υδρατμών, των νεφών και της θερμοκρασίας της τροπόσφαιρας. Επιπλέον, χρησιμοποιείται για τη μελέτη της δομής και της χωρο-χρονικής μεταβολής του Ατμοσφαιρικού Οριακού Στρώματος και την ανίχνευση της εισροής αερολυμάτων στον Ελληνικό εναέριο χώρο από το έδαφος εώς ύψος 15000-20000 μέτρων πάνω από τη μέση στάθμη της θάλασσας.

2.5.3 Λίγα λόγια για τον Ikonos

Οι εικόνες που θα χρησιμοποιηθούν στην παρούσα εργασία αποτελούν κατά κύριο λόγο εικόνες υψηλής διακριτικής ικανότητας που προέρχονται από τον δέκτη Ikonos. Για το λόγο αυτό κρίνεται σκόπιμο να αναφερθούν κάποιες γενικές πληροφορίες για τον συγκεκριμένο δέκτη και τον τρόπο που λειτουργεί.

Ο Ikonos είναι ένας εμπορικός δορυφόρος παρατήρησης της γης και ήταν ο πρώτος που παρείχε στο κοινό εικόνες υψηλής διακριτικής ευκρίνειας στα 1 και 4 μέτρα ανάλυση. Παρέχει πολυφασματικές και παγχρωματικές εικόνες οι οποίες προορίζονται για επιστημονικές έρευνες, μελέτες και ειρηνικές εφαρμογές. Οι τηλεπισκοπικές απεικονίσεις Ikonos είναι δυνατό να χρησιμοποιηθούν για εφαρμογές για τις οποίες μέχρι εκείνη την εποχή γινόταν χρήση αεροφωτογραφιών. Τα προτερήματα των απεικονίσεων Ikonos έναντι των αεροφωτογραφιών είναι η σταθερότητα της πλατφόρμας και του δέκτη (με άμεση συνέπεια την καλύτερη ποιότητα των απεικονίσεων), η συστηματικότητα των επαναληπτικών λήψεων κάτω από τις ίδιες συνθήκες και το χαμηλό κόστος.Στον παρακάτω πίνακα (Πίνακας 2.3) παρουσιάζονται κάποια χαρακτηριστικά του δορυφόρου:

Κανάλια	**Παγχρωματικος δέκτης**	**Πολυφασματικός δέκτης**			
		1	2	3	4
Φασματική διακριτική ικανότητα	Ορατό + Εγγύς υπέρυθρο	Μπλέ	Πράσινο	Κόκκινο	Εγγύς υπέρυθρο
Μήκος κύματος	0.45-0.9 μm	0.45-0.53 μm	0.52-0.61 μm	0.64-0.72 μm	0.76-0.88 μm
Χωρική διακριτική ικανότητα	1 μέτρο	4 μέτρα			
Εύρος κάλυψης	11 χιλιόμετρα				
Επαναληπτικότητα λήψης	2.9 ημέρες με 1m Δ.Ι. 1.5 ημέρες με 1.5m Δ.Ι.				
Γεωμετρική ακρίβεια	23m χωρίς φωτοσταθερά εώς 2m με παραγωγή ορθοφωτογραφίας				

Πίνακας 2.3: Κύρια χαρακτηριστικά του δορυφόρου Ikonos

Ο τηλεπισκοπικός δορυφόρος Ikonos-2 ακολουθεί κυκλική, ηλιοσύγχρονη, σχεδόν πολική τροχιά σε ύψος 681km με κλίση 98.1°, ως προς τον Ισημερινό, η οποία εγγυάται πλήρη κάλυψη της γης. Η τροχιά του διασταυρώνεται με τον Ισημερινό στις 10:30 το πρωί. Ο δέκτης του μπορεί να παίρνει κεκλιμένες απεικονίσεις έως και σε 700km απόσταση από οποιαδήποτε πλευρά της τροχιάς του δορυφόρου.

Κεφάλαιο 3

Εισαγωγή στο Orfeo Toolbox και Monteverdi

3.1 Εισαγωγή

Το Orfeo Toolbox αποτελεί μια βιβλιοθήκη γραμμένη σε C++, ενώ περιέχει αλγορίθμους τηλεπισκόπησης και επεξεργασίας εικόνας. Από την άλλη μεριά, το Monteverdi αποτελεί το γραφικό περιβάλλον που προσφέρει η συγκεκριμένη βιβλιοθήκη για όλους τους χρήστες της.

Το λογισμικό, πραγματοποιήθηκε από τον Γαλλικό Οργανισμό Διαστήματος CNES, το 2006, και ακόμα και σήμερα αναπτύσσεται, με την όλο και μεγαλύτερη συμμετοχή της Κοινότητας του Ελεύθερου Λογισμικού. Τα μέλη που ασχολούνται με την ανάπτυξη των αλγορίθμων που περιέχει η βιβλιοθήκη όλο και πληθαίνουν με αποτέλεσμα να πληθαίνουν και να βελτιώνονται και οι εφαρμογές που παρέχει. Σκοπός το συγκεκριμένου πακέτου είναι να παρέχει στον χρήστη όλα τα απαραίτητα εργαλεία για να χρησιμοποιήσει και να επεξεργαστεί τις δορυφορικές του εικόνες. Η βιβλιοθήκη μπορεί να χρησιμοποιηθεί σε υψηλής ευκρίνειας δορυφορικές εικόνες όπως αυτές που προκύπτουν από τους δορυφόρους Pleiades και Cosmo-Skymed καθώς επίσης και σε ένα ευρύ φάσμα άλλων αισθητήρων. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

Το OTB όπως επίσης και το γραφικό του περιβάλλον Monteverdi δημιουργήθηκαν από την συνεργασία και την προσπάθεια μιας ολόκληρης κοινότητας. Το συγκεκριμένο εργαλείο μπορεί και αναμένεται να χρησιμοποιηθεί τόσο για σκοπούς έρευνας, διδασκαλίας όσο και για εμπορικούς σκοπούς. Έτσι και αλλιώς το πακέτο μπορεί να διανεμηθεί κάτω από την ελεύθερη άδεια CeCILL. Η συγκεκριμένη άδεια είναι συμβατή με την GPL (GNU General Public License) η οποία αποτελεί ελεύθερη άδεια για λειτουργικά και οποιαδήποτε άλλη χρήση. Η συγκεκριμένη άδεια διασφαλίζει στην ουσία ότι όλες οι εκδόσεις οποιουδήποτε προγράμματος που την διαθέτει είναι ελεύθερες και θα παραμείνουν ελεύθερες για όλους τους χρήστες.

Η δημιουργία του συγκεκριμένου προγράμματος χρειαζόταν καθώς:

- Δημιούργησε νέες δυνατότητες και επιδόσεις (υψηλή ευκρίνεια σε οπτικά δορυφορικά αποτελέσματα και αποτελέσματα από ραντάρ, ποιότητα αποτελεσμάτων, δυνατότητα να συνδυαστούν δορυφορικά δεδομένα και δεδομένα από radar). [Emmanuel Christophe, Jordi Inglada, 2009]
- Ήταν απαραίτητη η δημιουργία νέων μεθόδων επεξεργασίας ή ακόμα και προσαρμογή των ήδη υπαρχόντων.
- Η ανάγκη κατανόησης όλων των νέων μεθόδων επεξεργασίας έτσι ώστε να αποκτήσει

όσο γίνεται καλύτερα αποτελέσματα στο σύστημά του.

3.2 Εισαγωγικά για το OTB

Προτού αρχίσει ο χρήστης να εξοικειώνεται με το πλήθος των αλγορίθμων που παρέχει το OTB, καλό είναι να εξοικειωθεί με τον τρόπο που είναι οργανωμένο το λογισμικό και την δομή των αρχείων του. Είναι πολύ σημαντικό να μπορεί ο χρήστης να πλοηγηθεί μέσα στην βάση του κώδικα του προγράμματος προκειμένου να βρει λυμένα παραδείγματα, κώδικα ή ακόμα και έγγραφα έτσι ώστε να λύσει κάθε είδους πρόβλημά του. Έτσι λοιπόν το OTB είναι οργανωμένο σε επιμέρους τρεις ενότητες: το OTB, το OTB-Documents και το OTB-Applications. Ο πηγαίος κώδικας του προγράμματος μαζί με τα παραδείγματα και τις εφαρμογές μπορεί να βρεθεί στην ενότητα του OTB, τα έγγραφα και οι οδηγοί εκπαίδευσης μπορούν να βρεθούν στην ενότητα του OTB-Documents, ενώ τέλος, σύνθετες εφαρμογές με χρήση του OTB (και άλλων συστημάτων όπως του ITK και της FLTK είναι διαθέσιμα στην ενότητα του OTB-Applications). Συνήθως ο χρήστης θα ασχοληθεί με την ενότητα του OTB εκτός και άμα ανήκει στην ομάδα των προγραμματιστών, διδάσκει ένα μάθημα, ή ψάχνει για λεπτομέρειες σε κάποιο είδος προβλήματος. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

Όπως έχει προαναφερθεί το OTB είναι μια βιβλιοθήκη γραμμένη σε C++ και βασισμένη στην βιβλιοθήκη ITK, βιβλιοθήκη για ιατρικές εφαρμογές. Παρόλο που η γλώσσα που είναι γραμμένη το OTB είναι η C++ εφαρμογές έχουν γραφτεί και σε υπόλοιπες ελεύθερες γλώσσες προγραμματισμού (FLOSS) όπως η Python και η Java, όπου και οι εφαρμογές αυτές είναι διαθέσιμες σε ξεχωριστό πρόγραμμα OTB-Wrapping. Ένα ακόμα πολύ σημαντικό σημείο αποτελεί το γεγονός ότι το OTB μπορεί να εγκατασταθεί πολύ εύκολα σε πληθώρα λειτουργικών συστημάτων όπως τα Windows,το MacOS, και διάφορες διανομές GNU/Linux όπως το Ubuntu και το Open Suse. Το γεγονός αυτό το κάνει προσιτό σε μεγάλο μέρος χρηστών ανεξάρτητα με ποια γλώσσα προγραμματισμού είναι εξοικειωμένη ή ποιο λειτουργικό σύστημα χρησιμοποιούν. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

Ο κώδικας του OTB είναι πολύ πλούσιος σε πολλές εφαρμογές, πράγμα που το καθιστά ένα επιπλέον πολύ σημαντικό εργαλείο τηλεπισκόπησης. Όλος ο κώδικας του OTB προκειμένου να γίνει compile με τον σωστό τρόπο, χρησιμοποιείται η CMake, ένα ανοιχτού λογισμικού σύστημα δόμησης (build) προγραμμάτων. Ο ρόλος της CMake είναι να ελέγχει την διαδικασία μεταγλώττισης (compilation) του κώδικα χρησιμοποιώντας μια πλατφόρμα και ανεξάρτητα αρχεία διαμόρφωσης. Ένα ακόμα σημαντικό στοιχείο για την CMake είναι το γεγονός ότι είναι αρκετά πολύπλοκη και υποστηρίζει περίπλοκα συστήματα, σαν το OTB. Οι πληροφορίες που χρειάζεται η CMake για να μεταγλωττίσει τον κώδικα βρίσκονται σε ένα αρχείο CMakeList.txt το οποίο βρίσκεται σε κάθε φάκελο που υπάρχει ο αντίστοιχος κώδικας. Συνήθως η πληροφορία που περιέχει το αρχείο, οδηγεί στις βιβλιοθήκες και τα αρχεία επικεφαλίδες (header files) που έχουν προσδιοριστεί από τον χρήστη και είναι απαραίτητα για την εκτέλεση του προγράμματος.

Προκειμένου ο χρήστης να εξοικειωθεί πολύ πιο εύκολα και ομαλά με το περιβάλλον του OTB, μπορεί να διαβάσει εκτός από τον αναλυτικό οδηγό του λογισμικού, και έναν οδηγό που περιγράφει τις εφαρμογές του OTB Applications αλλά και του Monteverdi. Ο συγκεκριμένος οδηγός (CookBook) είναι πολύ χρήσιμος για τους μη προγραμματιστές που θέλουν απλά να καταλάβουν τον τρόπο με τον οποίο λειτουργούν οι εφαρμογές του OTB. Στην συνέχεια, ο χρήστης μπορεί να απευθυνθεί στο doxygen όπου αποτελεί τον λεπτομερή οδηγό για όλον τον κώδικα του προγράμματος. Στον συγκεκριμένο οδηγό, εμφανίζονται όλες οι κλάσεις και

οι υποκλάσεις που χρησιμοποιεί το συγκεκριμένο πρότυπο (template) έτσι ώστε ο χρήστης να μπορεί να ελέγξει το διάγραμμα ροής του. Τέλος, οι προγραμματιστές του περιβάλλοντος, έχουν δημιουργήσει ένα ολόκληρο wiki που περιέχει όλες τις απαραίτητες για τον χρήστη πληροφορίες.

Σήμερα σύμφωνα με τα στατιστικά στοιχεία της επίσημης σελίδας του OTB υπάρχουν περίπου 21 άτομα που ασχολούνται με την βελτίωση του OTB, ενώ ο κώδικάς του έχει περάσει τα 6 εκατομμύρια γραμμές. Παράλληλα, τα άτομα που ασχολούνται με το συγκεκριμένο περιβάλλον πραγματοποιούν διάφορες παρουσιάσεις σε όλο το κόσμο προκειμένου να το κάνουν περισσότερο γνωστό. Τα παραπάνω στοιχεία αποδεικνύουν πως το περιβάλλον αυτό αποτελεί ένα αναπτυσσόμενο λογισμικό, με πολλές δυνατότητες για εφαρμογές τηλεπισκόπησης.

Τέλος στο σημείο αυτό κρίνεται σκόπιμο να παρουσιαστεί ένας πίνακας που να περιέχει τις βασικές εφαρμογές που μπορεί σήμερα να πραγματοποιήσει το OTB

Εφαρμογή	**Λεπτομέρειες**
Επεξεργασία εικόνας	Διάβασμα και επεξεργασία των περισσότερων τύπων τηλεπισκοπικών εικόνων (με χρήση της GDAL), πρόσβαση στα μετα δεδομένα της εικόνας, οπτικοποίησή της
Πρόσβαση δεδομένων	πρόσβαση σε διανυσματικά δεδομένα (shapefile, kml), μοντέλα DEM, δεδομένα Lidar
Φίλτρα εικόνων	φίλτρα για θόρυβο, ενίσχυση οπτικών και radar δεδομένων
Εξαγωγή αντικειμένων	εξαγωγή υφής με χρήση των Haralick, SFS, Pantex, ένταση των ακμών, εξαγωγή σημείων και γραμμών ενδιαφέροντος
Κατάτμηση εικόνας	αλγόριθμοι region growing, watershed, level sets
Ταξινομήσεις	αλγόριθμοι K-Means, SVM, Markov random fields
Ανίχνευση μεταβολών	—
Ορθο-διόρθωση και προβολή σε χάρτη	χρήση ossim
Ραδιομετρικοί δείκτες	βλάστηση, νερό, έδαφος
Αντικειμενοστραφής κατάτμηση και φίλτρα	—
Υπολογισμός PCA	—
Απεικονίσεις	παρέχει ένα ευέλικτο σύστημα απεικόνισης
Πολλές άλλες εφαρμογές	

Πίνακας 3.1: Βασικές εφαρμογές OTB

3.3 Εισαγωγή στο Monteverdi

Η εφαρμογή Monteverdi αποτελεί το γραφικό περιβάλλον του OTB, παρέχοντας στον χρήστη τα κύρια εργαλεία που χρειάζεται για οποιαδήποτε τηλεπισκοπική εφαρμογή. Η εφαρμογή του

Monteverdi είναι τελείως ανεξάρτητη από αυτή του OTB καθώς δεν περιέχει όλους τους αλγορίθμους του OTB και απευθύνεται απλά σε χρήστες που δεν έχουν εξοικειωθεί ακόμα με την χρήση του συγκεκριμένου κώδικα.

Η συγκεκριμένη εφαρμογή αποτελεί μια έξυπνη αρχιτεκτονική λογισμικού, το οποίο επιτρέπει αλυσιδωτές ενέργειες που πραγματοποιούνται με την επιλογή της κατάλληλης φόρμουλας από ένα σετ από μενού που θα αναφερθούν σε επόμενο κεφάλαιο. Η εφαρμογή υποστηρίζει τόσο raster (πινακοποιημένους τύπους) τύπους δεδομένων όσο και vector (διανυσματικά δεδομένα). Ένα ακόμα πολύ σημαντικό στοιχείο της αρχιτεκτονικής είναι το γεγονός ότι χρησιμοποιεί τα πλεονεκτήματα της διαχείρισης ροών στα δεδομένα (streaming) και πολλαπλών νημάτων (multi threading) στο προγραμματισμό, τα οποία χρησιμοποιούν ως τρόπο λειτουργίας τις διαχειρίσεις ροών (pipelines) όπως άλλωστε και ο ίδιος ο κώδικας του OTB στηρίζεται στον ίδιο τρόπο. [Emmanuel Christophe, Julien Michel and Jordi Inglada,2011] Τέλος, ο χρήστης μπορεί να επεξεργαστεί τόσο αρχεία δορυφορικών εικόνων όσο και εικόνων ραντάρ.

Οι κύριες λειτουργίες που έχει σήμερα το Monteverdi αποτελούν τα βασικά εργαλεία της τηλεπισκόπησης, δηλαδή, φίλτρα, γεωμετρικές διορθώσεις, κατατμήσεις, ταξινομήσεις, πράξεις καναλιών, επεξεργασία SAR εικόνων και απεικόνιση των αποτελεσμάτων. Το περιβάλλον του μπορεί να προσαρμοστεί τόσο από εφαρμογές που στηρίζονται σε εικονοστοιχεία όσο και από εφαρμογές που στηρίζονται σε αντικείμενα. Παρόλο που οι εφαρμογές που υποστηρίζει το περιβάλλον είναι περιορισμένες σε σχέση με τις δυνατότητες του κώδικα, συνεχώς οι developers αναπτύσσουν το γραφικό περιβάλλον, προσθέτοντας και άλλες.

3.4 Εγκατάσταση OTB και Monteverdi

Προτού ο χρήστης αρχίσει να εξοικειώνεται με τον κώδικα του OTB, πρέπει να εγκαταστήσει τον κώδικα αλλά και το γραφικό περιβάλλον του OTB στον υπολογιστή του. Περιοδικές εκδόσεις του προγράμματος καθώς επίσης και του γραφικού περιβάλλοντός του μπορούν να βρεθούν στην ιστοσελίδα του OTB. Οι εκδόσεις αυτές είναι διαθέσιμες σε τακτά χρονικά διαστήματα και ανακοινώνονται στην σελίδα του προγράμματος αλλά και στις λίστες ηλεκτρονικού ταχυδρομείου, στις οποίες ο χρήστης καλό είναι να γραφεί προκειμένου να λαμβάνει τα νέα του προγράμματος αλλά και να ρωτάει οποιαδήποτε απορία του. Σήμερα, οι εκδόσεις που είναι διαθέσιμες και θα ασχοληθούμε με αυτές σε όλη την διπλωματική είναι για τον κώδικα η Orfeo-Toolbox-3.10 και για το γραφικό του περιβάλλον η Monteverdi-1.8. Οι εκδόσεις αυτές, δόθηκαν στους χρήστες τον Ιούνιο του 2011.

Όπως έχει ήδη αναφερθεί παραπάνω το orfeo είναι συμβατό με σχεδόν όλα τα λογισμικά. Οι τρόποι για να εγκατασταθεί το OTB και το Monteverdi είναι δύο. Ο πρώτος είναι χρησιμοποιώντας τα έτοιμα αποθετήρια (repositories) που υπάρχουν στην επίσημη σελίδα του προγράμματος και ο δεύτερος είναι εγκαθιστώντας χειροκίνητα τον κώδικα στο σύστημα του κάθε χρήστη. Σε κάθε περίπτωση προτιμάται ο πρώτος τρόπος ο οποίος είναι πιο αυτόματος και παράλληλα ο χρήστης είναι σίγουρος πως ο κώδικας που εγκαθιστά υπάρχει στο σωστό μέρος. Παρόλα αυτά για λόγους πληρότητας θα αναλυθούν παρακάτω και οι δύο τρόποι εγκατάστασης σε περιβάλλον Linux, Ubuntu.

Προκειμένου να λειτουργήσει το OTB χρειάζεται μια σειρά από στοιχεία. Εκτός λοιπόν από την cmake που αναφέραμε παραπάνω, εξαρτάται από τρεις βιβλιοθήκες. Οι βιβλιοθήκες αυτές είναι οι ITK, GDAL και Fltk

- **Βιβλιοθήκη ITK**
  Η βιβλιοθήκη δημιουργήθηκε το 1999 από την Εθνική Βιβλιοθήκη της Ιατρικής της Αμερικής, όπου και ανέλαβε τριετές συμβόλαιο για να αναπτύξει ένα ελεύθερου κώδικα,

εργαλείο καταχώρισης και κατάτμησης. Ολοκληρώθηκε το 2002 με το όνομα ITK ή Insight Toolkit.
Η ITK είναι γραμμένη σε C++, αποτελεί μια πλατφόρμα για τα επιμέρους προγράμματα και χρησιμοποιεί την CMake προκειμένου να δομήσει το περιβάλλον στο οποίο θα μεταγλωττίσει τα διάφορα προγράμματα. Παράλληλα, εκτός από την C++ διαθέτει και Wrapping σε άλλες γλώσσες προγραμματισμού όπως είναι η Tcl, Java, Python. Με τον τρόπο αυτό όλοι οι χρήστες μπορούν να χρησιμοποιήσουν το λογισμικό σε πληθώρα γλωσσών προγραμματισμού. Επίσης, η βιβλιοθήκη χρησιμοποιεί τις τεχνικές του γενετικού προγραμματισμού μέσω των προτύπων της C++.
Όπως κάθε λογισμικό, διαθέτει την κοινότητα που το υποστηρίζει και συμμετέχει ενεργά σε διάφορες εκδηλώσεις. Η άδεια που διαθέτει είναι η BSD άδεια ελεύθερου κώδικα, η οποία επιτρέπει την χρήση για πολλούς σκοπούς.
Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται κυρίως για ιατρικές εφαρμογές, αλλά το γεγονός ότι μπορεί να αναπτύσσει μέσω ακραίων προγραμματιστικά μεθόδων αλγορίθμους εύρεσης ακμών και κατάτμησης πολυδιάστατων δεδομένων, γίνεται πολύ καλό εργαλείο για την ανάλυση εικόνας γενικότερα. Στην περίπτωσή μας και εάν ο χρήστης εγκαθιστά τον κώδικα χειροκίνητα, έχει την επιλογή να εγκαταστήσει την δική του έκδοση της βιβλιοθήκης ή να χρησιμοποιήσει αυτή που έχει ήδη το OTB. Τέλος, πολλές από τις κλάσεις βάσεις που χρησιμοποιεί ο κώδικας, ανήκουν στην βιβλιοθήκη της ITK. [http://www.itk.org/, 2011]
Στο σημείο αυτό κρίνεται σκόπιμο να παρουσιαστεί ένας πίνακας με τα βασικά εργαλεία που διαθέτει η βιβλιοθήκη:

Εφαρμογή	**Λεπτομέρειες**
Χρήση δεδομένων και αλγορίθμων κατάτμησης και εγγραφής	Κυρίως για ιατρικές εφαρμογές, χωρίς να απαγορεύει την χρήση και σε άλλου τύπου δεδομένων
Χρήση δεδομένων από διάφορους τύπους εικόνων	Επεξεργασία εικόνων ανεξάρτητα από τύπο και διαστάσεις
Δεν περιέχει γραφικό περιβάλλον για απεικόνιση	Το πεδίο αυτό καλύπτεται από άλλα εργαλεία όπως τα VTK, VisPack, Metadata
Δεν περιέχει πολλά εργαλεία για επεξεργασία αρχείων	Το πεδίο καλύπτεται από άλλα εργαλεία
Υποστηρίζεται η παράλληλη επεξεργασία δεδομένων	————
Μπορεί να χρησιμοποιηθεί από πληθώρα πλατφόρμων και μεταγλωττιστών χωρίς προβλήματα	————

Πίνακας 3.2: Βασικές εφαρμογές ITK

- **Βιβλιοθήκη GDAL**
  Η βιβλιοθήκη της GDAL είναι μια βιβλιοθήκη που υποστηρίζει όλα τα raster γεωχωρικά δεδομένα. Όπως και η παραπάνω, πρόκειται για μια βιβλιοθήκη ελεύθερου κώδικα. Εκτός από το γεγονός ότι υποστηρίζει όλους τους τύπους γεωχωρικών δεδομένων, η

συγκεκριμένη βιβλιοθήκη παρέχει ένα χρήσιμο εργαλείο για επεξεργασία και μετατροπή των δεδομένων. Τα περισσότερα λογισμικά ελεύθερου κώδικα που ασχολούνται με εικόνες την χρησιμοποιούν προκειμένου να αναγνωρίζουν τα δεδομένα τους.
Σήμερα, βρίσκεται στην έκδοση 1.8.1 η οποία και δόθηκε επίσημα στους χρήστες τον Ιούλιο του 2011. Είναι μια πλατφόρμα γραμμένη σε C, C++, ενώ οι αλγόριθμοι της υπάρχουν και σε άλλες γλώσσες όπως Perl, Python, V56 Bindings, Java, Ruby. Τέλος, ο χρήστης εκτός από τους πολύ καλούς οδηγούς εκμάθησης της πλατφόρμας μπορεί να απευθυνθεί και στην αντίστοιχη κοινότητα για οποιαδήποτε απορία του.
Ο χρήστης θα πρέπει να εγκαταστήσει την GDAL στο σύστημά του προκειμένου να μπορέσει να χρησιμοποιήσει τον κώδικα του orfeo. Έτσι και αλλιώς η συγκεκριμένη βιβλιοθήκη είναι απαραίτητη σε όλες σχεδόν τις τηλεπισκοπικές εφαρμογές. [http://www.gdal.org, 2011]

- **Βιβλιοθήκη FLTK**
  Η βιβλιοθήκη της FLTK (Fast, Light Toolkit) είναι μια πλατφόρμα GUI η οποία κατασκευάστηκε ως ένα γραφικό περιβάλλον για την OpenGL, αλλά παράλληλα είναι κατάλληλη για όλα συστήματα που χρησιμοποιούν τον γενετικό προγραμματισμό. Σήμερα η έκδοση που υπάρχει για την βιβλιοθήκη είναι η 1.3.0 και βγήκε τον Ιούνιο του 2011. Είναι γραμμένη σε C++ και μπορεί να λειτουργήσει στα περισσότερα λειτουργικά συστήματα.
  Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για να οπτικοποιήσει όλες τις συναρτήσεις και τους αλγορίθμους που υπάρχουν στον κώδικα. Είναι απολύτως απαραίτητη για την χρήση του κώδικα. Ο κώδικας του OTB έχει δοκιμαστεί με την έκδοση της Fltk 1.1.7 και 1.1.9. Ο χρήστης μπορεί και σε αυτή την περίπτωση να διαλέξει μεταξύ του να κατεβάσει την βιβλιοθήκη από την σελίδα της Fltk ή να χρησιμοποιήσει αυτή που έχει από μόνος του ο κώδικας και αντιστοιχεί στην έκδοση 1.1.9. [http://www.fltk.org, 2011]

3.4.1 Εγκατάσταση από τα αποθετήρια (repositories)

Προκειμένου η χρήση του OTB να γίνει πιο εύκολη στους χρήστες τους, οι προγραμματιστές του έφτιαξαν έτοιμα πακέτα για μια σειρά από λειτουργικά συστήματα, έτσι ώστε η εγκατάστασή του να γίνει αυτόματη. Στον παρακάτω πίνακα παρουσιάζονται τα πακέτα για κάθε τμήμα του OTB και για κάθε λειτουργικό τμήμα. Τα κελιά που είναι κενά, δεν έχουν κάποιο πακέτο έτοιμο για εγκατάσταση και ο χρήστης πρέπει να το εγκαταστήσει από το πηγαίο κώδικα όπως θα παρουσιαστεί παρακάτω. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

Λειτουργικά Συστήματα	**OTB library**	**Monteverdi**	**OTB-Applications**	**Wrapping (Java and Python**
Windows 2000/XP/Vista/Seven		Windows installer	Windows installer	Windows installer
MacOs X 10.5 and higher		app's DMG file		
Ubuntu Linux 10.10, 10.04 and 9.10 (32/64 bits)	APT repository	APT repository	APT repository	

Πίνακας 3.3 – συνέχεια από προηγούμενη σελίδα

Λειτουργικά Συστήματα	**OTB library**	**Monteverdi**	**OTB-Applications**	**Wrapping (Java and Python**
OpenSuse 11.2 and higher (32/64 bits)	RPM package	RPM package	RPM package	

Πίνακας 3.3: Διαθέσιμα πακέτα OTB

Στο σημείο αυτό, θα παρουσιαστούν όλες οι απαραίτητες εντολές που πρέπει να δώσει ένας χρήστης με λειτουργικό Linux, Ubuntu για να εγκαταστήσει το πακέτο στον υπολογιστή του. Αρχικά σε περίπτωση που ο χρήστης χρησιμοποιεί για εγκατάσταση προγραμμάτων το Synaptic, μπορεί από το γραφικό αυτό περιβάλλον να προσθέσει τα πακέτα που θέλει. Σε περίπτωση όμως που ο χρήστης θέλει να να εγκαταστήσει το λογισμικό από το τερματικό, μπορεί απλά να πληκτρολογήσει τις παρακάτω εντολες:

```
sudo aptitude install add-apt-repository
sudo add-apt-repository ppa:otb/orfeotoolbox-stable
sudo aptitude update
sudo aptitude install otb otbapp monteverdi otb-wrapping-python
otb-wrapping-java
```

Με τον τρόπο αυτό ο κάθε χρήστης έχει εγκαταστήσει την σταθερή έκδοση τουOTB στον υπολογιστή του, μαζί με τις εφαρμογές του Monteverdi αλλά και τα Wrapping Java, Python. Υπάρχει περίπτωση ο χρήστης να μην θέλει την σταθερή έκδοση του προγράμματος αλλά τις επιμέρους διορθώσεις του κώδικα που γίνονται κάθε βδομάδα. Σε αυτή την περίπτωση τα πακέτα είναι διαφορετικά και οι εντολές διαφορετικές. Σε καμία περίπτωση όμως δεν πρέπει ο χρήστης να εγκαταστήσει και τα δύο πακέτα γιατί τότε θα δημιουργηθούν προβλήματα συμβατότητας. Για περισσότερες λεπτομέρειες ο χρήστης καλό είναι να ανατρέξει στο αντίστοιχο κεφάλαιο του επίσημου οδηγού για εγκατάσταση του λογισμικού.

3.4.2 Εγκατάσταση του OTB από τον πηγαίο κώδικα

Το λογισμικό του OTB έχει δοκιμαστεί σε διάφορους συνδυασμούς συστημάτων, μεταγλωττιστών και λογισμικών όπως είναι τα MS-Windows, Linux αλλά και τα Solaris, Mac OSX. Από τους διάφορους συνδυασμούς προέκυψε πως λειτουργεί με τους ακόλουθους μεταγλωττιστές:

- Με τους μεταγλωττιστές Cygwin, MinGW, Visual Studio 7, 8, 9 καθώς επίσης και τους Visual Studio Express 2005, 2008, 2010 σε λειτουργικό περιβάλλον MS-Windows.
- Με τον GCC για το Unix/Linux.
- Με τον GCC πάλι για περιβάλλον Mac.

Στο σημείο αυτό πρέπει να τονιστεί πως κάποιοι μεταγλωττιστές μπορεί να παρουσιάσουν κάποια προβλήματα σε περίπτωση που δεν είναι στην τελευταία τους έκδοση. Στην περίπτωση αυτή, ο χρήστης μπορεί να αναβαθμίσει τον μεταγλωττιστή του και ο κώδικας θα μεταγλωττιστεί με το σωστό τρόπο.

Πριν πραγματοποιήσει οποιαδήποτε ενέργεια ο χρήστης πρέπει να κατεβάσει τον κώδικα του OTB. Αυτό μπορεί να γίνει με τρεις διαφορετικούς τρόπους.

- Από την αντίστοιχη σελίδα που υπάρχει η επίσημη έκδοση του κώδικα: http://www.orfeo-toolbox.org/otb/download.html. Έτσι ο χρήστης μπορεί να αποκτήσει την τελευταία έκδοση του προγράμματος σε συμπιεσμένη μορφή και να την εγκαταστήσει στον υπολογιστή του.
- Από τον OTB mercurial server αντιγράφοντας την τελευταία επίσημη έκδοση μέσω του Mercurial. Το Mercurial είναι μια ελεύθερη, πηγή διανομής και διαχείρισης εργαλείων. Έχει την δυνατότητα να χειρίζεται προγράμματα οποιουδήποτε μεγέθους και να προσφέρει ένα εύκολο και φιλικό γραφικό περιβάλλον για χρήση. Ο χρήστης πρέπει να εγκαταστήσει το Mercurial πριν κατεβάσει τον κώδικα.
- Αντιγράφοντας από τον OTB mercurial server την τελευταία αναβάθμιση του προγράμματος, χωρίς αυτή να είναι απαραίτητα σταθερή.

Στην συνέχεια, ο χρήστης πρέπει να εγκαταστήσει τις αντίστοιχες βιβλιοθήκες ITK, GDAL, FLTK όπως και παραπάνω αναφέρθηκε. Η όλη μεταγλώττιση του κώδικα γίνεται όπως επισημάνθηκε προηγουμένως μέσω της CMake. Το περιβάλλον του OTB απαιτεί τουλάχιστον την έκδοση 2.6 της CMake. Προκειμένου να γίνει η μεταγλώττιση του κώδικα η CMake απαιτεί δύο είδη πληροφορίας, το πρώτο είναι ο χώρος στον οποίο υπάρχει ο κώδικας και έχει κατέβει στο σύστημα του χρήστη και το δεύτερο είναι ο χώρος στον οποίο θα παραχθεί το προϊόν της μεταγλώττισης. Προκειμένου να τρέξει η CMake, γίνεται η χρήση της εντολής ccmake για το περιβάλλον Unix και CMakeSetup σε περιβάλλον Windows. Η ccmake πρέπει να πραγματοποιηθεί μέσα στον αρχικό φάκελο που βρίσκεται ο πηγαίος κώδικας. Οι περισσότερες μεταβλητές που χρειάζονται για την σωστή εγκατάσταση του OTB είναι ήδη προκαθορισμένες. Για όποια μεταβολή κάνει ο χρήστης πρέπει να πραγματοποιήσει configuration. Την στιγμή λοιπόν που δεν θα υπάρχουν άλλα configure τότε ο χρήστης μπορεί να συνεχίσει κάνοντας generate τον κώδικα. Από την στιγμή που θα γίνει το generate, ο κώδικας του OTB είναι έτοιμος για χρήση. Η ίδια διαδικασία πραγματοποιείται για την εγκατάσταση του Monteverdi αλλά και των OTB-Applications, Wrapping.

Από την στιγμή που το πρόγραμμα έχει εγκατασταθεί σωστά στο σύστημα του χρήστη, δημιουργούνται κάποιοι φάκελοι που περιέχουν όλο τον κώδικά του. Οι φάκελοι αυτοί βρίσκονται στα παρακάτω directories:

- *usr/bin*
- */usr/include/otb*
- */usr/lib/otb*
- */var/lib/dpkg/info*
- */usr/bin/monteverdi*
- */usr/share/menu/monteverdi*
- */usr/share/doc/monteverdi*

3.4.3 Εγκατάσταση από πηγαίο κώδικα σε Ubuntu

Στο κεφάλαιο αυτό, θα παρουσιαστεί αναλυτικότερα ο τρόπος με τον οποίο το OTB μπορεί να εγκατασταθεί από τον κώδικα σε περιβάλλον Ubuntu 11.04.

Αρχικά ο χρήστης πρέπει να κατεβάσει τον κώδικα με κάποιον από τους τρεις διαθέσιμους

τρόπους και να τον τοποθετήσει σε έναν ξεχωριστό φάκελο. Στο φάκελο αυτό θα βρίσκονται απλά τα συμπιεσμένα αρχεία που έχει κατεβάσει στο σύστημά του ο χρήστης. Στην συνέχεια, ο χρήστης πρέπει να εγκαταστήσει μια σειρά από βιβλιοθήκες που είναι απαραίτητες για την μεταγλώττιση του κώδικα. Οι βιβλιοθήκες αυτές είναι αυτές που αναφέρθηκαν παραπάνω και πάνω σε αυτές στηρίζεται και το περιβάλλον του OTB. Σε περίπτωση έτσι και αλλιώς που δεν υπάρχει στο σύστημα του χρήστη κάποια βιβλιοθήκη, κατά την διάρκεια χτισίματος του προγράμματος θα εμφανιστεί λάθος και ο χρήστης θα έχει την δυνατότητα να εγκαταστήσει το αρχείο που λείπει.

Μέσα στο φάκελο που έχει κατεβάσει και αποσυμπιέσει τον κώδικα ο χρήστης, δημιουργεί έναν δεύτερο φάκελο (OTB-bin) καθώς προκειμένου να τρέξει η cmake χρειάζεται ένα binary χώρο. Στο περιβάλλον του Linux η cmake τρέχει μέσα από το τερματικό του συστήματος. Έτσι ο χρήστης πρέπει να ανοίξει το τερματικό του μέσα στον φάκελο που δημιούργησε (OTB-bin) και να τρέξει την cmake μέσω της εντολής ccmake ../OrfeoToolbox-3.8.0/. Μέσω της εντολής αυτής, το σύστημα βρίσκει το αρχείο του OTB και αρχίζει το χτίσιμό του. Με την εντολή αυτή στον άδειο φάκελο που δημιούργησε ο χρήστης αρχίζουν να δημιουργούνται επιμέρους αρχεία και φάκελοι. Στην συνέχεια στο παράθυρο που εμφανίζεται στο χρήστη επιλέγεται η επιλογή c (configure) και στην συνέχεια g (generate). Η όλη διαδικασία παίρνει λίγο χρόνο. Αφού ολοκληρωθεί, ο χρήστης πρέπει να τρέξει την εντολή make και τέλος sudo make install. Με τον τρόπο αυτό αρχίζει όλη η εγκατάσταση του κώδικα στο περιβάλλον του χρήστη.

Η διαδικασία όπως αναφέρθηκε παραπάνω είναι ακριβώς η ίδια για την εγκατάσταση του monteverdi αλλά και όποιου άλλου τμήματος του προγράμματος θέλει ο χρήστης. Έτσι και αλλιώς όλα τα προγράμματα του OTB χτίζονται μέσω της cmake, οπότε ο χρήστης πρέπει να εξοικειωθεί με την χρήση της.

Κεφάλαιο 4

Monteverdi

4.1 Εισαγωγή

Πριν αρχίσει οποιαδήποτε εφαρμογή στον κώδικα του OTB καλό είναι να εξοικειωθεί ο χρήστης και με το γραφικό περιβάλλον που παρέχει το OTB. Για το λόγο αυτό στο κεφάλαιο αυτό θα πραγματοποιηθούν κάποιες εφαρμογές με το Monteverdi. Το Monteverdi χρησιμοποιείται κυρίως από τους χρήστες που δεν έχουν εξοικειωθεί με το περιβάλλον του OTB και χρειάζονται ένα εργαλείο για να πραγματοποιήσουν κάποιες βασικές εφαρμογές τηλεπισκόπησης. Αποτελεί δηλαδή, έναν προθάλαμο μέχρι ο χρήστης να εξοικειωθεί με τον τρόπο που είναι δομημένο το OTB και να αρχίσει να πραγματοποιεί τις εφαρμογές του σε αυτό.

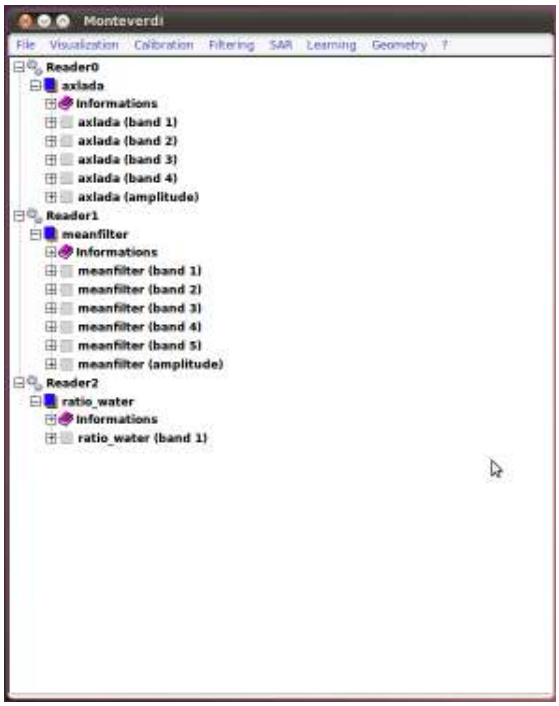
Αρχικά, στο σημείο αυτό πρέπει να αναφερθεί πως το Monteverdi δεν περιέχει όλους τους αλγορίθμους και τις εφαρμογές που βρίσκονται στον κώδικα του OTB, αλλά περιέχει κάποιες από αυτές που είναι απαραίτητες για τις κύριες εφαρμογές τηλεπισκόπησης. Κάθε νέα εισαγωγή δεδομένων γίνεται με χρήση του δείκτη reader. Τα δεδομένα που εισάγει ο χρήστης μπορεί να είναι είτε raster είτε vector, και για το λόγο αυτό έχουν το χαρακτηρισμό δεδομένα, data set και όχι εικόνες. Με αυτό τον τρόπο, ο χρήστης δεν χρειάζεται να κάνει καμία ειδική ενέργεια προκειμένου να αναγνωριστεί το συγκεκριμένο ανά περίπτωση format των δεδομένων. Έτσι λοιπόν όπως φαίνεται και στην παρακάτω εικόνα (Σχήμα 4.1) υπάρχουν τα εξής μενού επιλογών:

- Το μενού επιλογών file περιέχει όλα τα εργαλεία για άνοιγμα της εικόνας (Open dataset), αποθήκευσης κάποιας υποεικόνας που δημιουργείται (Save dataset, Save datacet(advanced)), εργαλείο για να μπορεί ο χρήστης να εφαρμόζει σταδιακά ένα πλήθος ενεργειών χωρίς να θέλει να αποθηκεύσει τα αποτελέσματα (Cache dataset), εργαλείο για κοπή εικόνας (Extract Roi from dataset), ένωση των καναλιών μιας εικόνας σε ένα κανάλι (Concatenate images), υπολογισμό των στατιστικών της εικόνας και την μετατροπή της vector εικόνας σε raster (Rasterize vector data).

- Το μενού επιλογών Visualization περιέχει όλα τα εργαλεία για εμφάνιση της εικόνας στον χρήστη μέσω διαφόρων εφαρμογών.

- Το μενού επιλογών Calibration περιέχει όλες τις ενέργειες για διόρθωση της δορυφορικής εικόνας (Optical Calibration) ή μιας εικόνας από ραντάρ (SAR Calibration) από την επιρροή της ατμόσφαιρας.

- Το μενού επιλογών Filtering περιέχει, το εργαλείο BandMath, για να μπορεί ο χρήστης να πραγματοποιήσει τις δικές του αριθμητικές πράξεις μεταξύ των καναλιών της εικόνας (π.χ. λόγοι), το εργαλείο για κατωφλίωση της εικόνας Threshold, μια σειρά από φίλτρα Feature Extraction, το εργαλείο για να πραγματοποιείται σε πανχρωματικές εικόνες



Σχήμα 4.1: Κεντρικό μενού Monteverdi

Pasharping, φίλτρα για change detection καθώς επίσης και το εργαλείο για διαχωρισμό αντικειμένων mean shift clustering τα οποία θα χρησιμοποιηθούν σε ταξινομήσεις.

- Το μενού επιλογών SAR περιέχει όλα τα εργαλεία για επεξεργασία εικόνων ραντάρ.
- Το μενού επιλογών Learning περιέχει τις ταξινομήσεις που έχει το Monteverdi δηλαδή επιβλεπόμενη ταξινόμηση SVM classification και μη επιβλεπόμενη ταξινόμηση KMeans clustering.
- Τέλος, το μενού επιλογών Geometry περιέχει όλα τα εργαλεία για την γεωμετρική διόρθωση της εικόνας.

4.2 Περιεχόμενα μενού του Monteverdi

Αναλυτικότερα, κάθε μενού επιλογών από τα παραπάνω περιέχει τα εξής υπο-μενού και εργαλεία: [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011] [Grizonnet Manuel and Inglada Jordi, 2010]

4.2.1 File

Σχήμα 4.2: Κεντρικό μενού Monteverdi για File

4.2.1.1 Open dataset

Από το συγκεκριμένο υπο-μενού μπορεί ο χρήστης να φορτώσει την εικόνα του στο σύστημα προκειμένου να ξεκινήσει την επεξεργασία της. Το εργαλείο αναγνωρίζει αμέσως τον τύπο του δεδομένου που φορτώνει ο χρήστης, δηλαδή αν πρόκειται για δορυφορική εικόνα ή εικόνα ραντάρ, οπότε η εικόνα φορτώνεται στο σύστημα. Αυτομάτως, δημιουργείται ένας δείκτης reader, που περιέχει την εικόνα. Η εικόνα αυτή αναλύεται από το σύστημα στα συστατικά της. Αρχικά εμφανίζεται όλη η εικόνα έτσι όπως έχει εισαχθεί από τον χρήστη, στην συνέχεια εμφανίζεται η εικόνα που περιέχει τις πληροφορίες για τις γεωμετρικές συντεταγμένες των pixel της, ενώ από κάτω ακολουθούν τα κανάλια της εικόνας ένα ένα ξεχωριστά. Παράλληλα, ο χρήστης, μπορεί να δει σε κάθε περίπτωση τον τύπο δεδομένων με το οποίο το σύστημα διαβάζει την εικόνα, καθώς επίσης και από ποια πηγή αντλεί τις πληροφορίες για τα επιμέρους μέρη που δημιουργεί.

4.2.1.2 Save dataset

Στο υπο-μενού αυτό, ο χρήστης μπορεί να αποθηκεύσει την εικόνα που θέλει να εξάγει και έχει δημιουργηθεί μέσα από μια σειρά από διαδικασίες. Το μόνο που έχει να κάνει ο χρήστης, είναι να επιλέξει την εικόνα που θέλει και στην συνέχεια του εμφανίζεται ένα παράθυρο στο οποίο επιλέγει τον φάκελο στον οποίο θα αποθηκεύσει την εικόνα, αλλά και τον τύπο των δεδομένων για τα pixel της εικόνας. Προτείνεται ο χρήστης να αφήνει τον τύπο που είναι προ επιλεγμένος από το σύστημα, εκτός και άμα έχει συγκεκριμένο λόγο που να επιθυμεί αλλαγή του τύπου των δεδομένων της εικόνας. Οι τύποι των δεδομένων που υποστηρίζονται καθώς επίσης και το μέγεθός τους είναι τα εξής: unsigned char (8bits), short (16 bits), int (32 bits), float (32 bits), double (64 bits), unsigned short (16 bits), unsigned int (32 bits). Η ύπαρξη τόσων πολλών τύπων δεδομένων είναι απαραίτητη καθώς όλες οι διαδικασίες που πραγματοποιούνται στο monteverdi οδηγούν σε float τύπο δεδομένου για τα εικονοστοιχεία μιας εικόνας. Σε μεγάλες τηλεπισκοπικές όμως εφαρμογές αυτό οδηγεί σε μεγάλο όγκο δεδομένων. Έτσι ο χρήστης μπορεί να επιλέξει ένα άλλο τύπο δεδομένου για τα εικονοστοιχεία της εικόνας του, έτσι ώστε το μέγεθός της να είναι μικρότερο και παράλληλα το αποτέλεσμά του ικανοποιητικό. Τέλος, ο χρήστης έχει την δυνατότητα, να επιλέξει να αποθηκευτούν τα metadata της εικόνας ή όχι.

4.2.1.3 Save dataset(advanced)

Στο συγκεκριμένο υπο-μενού ο χρήστης μπορεί να αποθηκεύσει τα δεδομένα του με περισσότερες όμως δυνατότητες. Αρχικά, αφού επιλέξει το μενού, εμφανίζεται ένα παράθυρο (writer application) στο οποίο εμφανίζονται η εικόνα, το αποτέλεσμα καθώς και ο τύπος των εικονοστοιχείων της εικόνας. Στο παράθυρο αυτό, ο χρήστης, μπορεί να πραγματοποιήσει μια σειρά από ενέργειες. Αρχικά μπορεί να εξάγει ένα κανάλι από την εικόνα που θέλει να αποθηκεύσει και να αποθηκεύσει μόνο αυτό ή ένα συνδυασμό καναλιών από τα παρεχόμενα από αυτή. Επίσης μια πολύ σημαντική εφαρμογή που μπορεί να γίνει από το συγκεκριμένο μενού είναι ότι μπορεί να γίνει μετατροπή της εικόνας από 64bit σε 32bit και οποιοδήποτε συνδυασμό τους. Έτσι από οποιοδήποτε τύπο δεδομένων μπορεί να παραχθεί μια σειρά από άλλους.

4.2.1.4 Cache dataset

Το υπο-μενού αυτό είναι πολύ ενδιαφέρον να αναλυθεί.Ο κώδικας του OTB προκειμένου να υλοποιηθεί, δημιουργεί μια σειρά από διοχέτευση ροών(pipeline). Καμία ενέργεια από αυτές που πραγματοποιεί ο χρήστης δεν πραγματοποιείται εκτός και άμα το ζητήσει ο ίδιος συγκεκριμένα. Με αυτό τον τρόπο λοιπόν ο χρήστης μπορεί ταυτόχρονα να ανοίγει μια εικόνα, να πραγματοποιεί γεωμετρική διόρθωση και να εφαρμόζει ένα φίλτρο για παράδειγμα, χωρίς να πραγματοποιείται τίποτα, μέχρι να ενεργοποιήσει την εκτέλεση της ροής. Η λειτουργία

αυτή είναι πολύ βολική καθώς ο χρήστης μπορεί να εφαρμόσει όλες τις ενέργειες που πρέπει να γίνουν στην εικόνα και μετά αυτές να πραγματοποιηθούν αυτόματα χωρίς την επιτήρησή του σε κάθε επιμέρους στάδιο. Στο γραφικό περιβάλλον όμως του OTB, κάθε ενέργεια που κάνει ο χρήστης αποθηκεύεται στο τελευταίο μοντέλο της ροής. Παρόλα αυτά, μερικές φορές, ο χρήστης μπορεί να θέλει να εφαρμόσει ένα μέρος της ροής που έχει δημιουργήσει, χωρίς να θέλει να δώσει σε αυτό ένα όνομα ή να πάρει κάποιο αποτέλεσμα. Η ενέργεια αυτή πραγματοποιείται από το συγκεκριμένο μενού. Με λίγα λόγια, το αποτέλεσμα της διαδικασίας μπορεί να αποθηκευτεί σε ένα προσωρινό αρχείο το οποίο μπορεί να βρεθεί στο αρχείο Caching το οποίο δημιουργείται από την εφαρμογή. Μια άλλη περίπτωση που ο χρήστης μπορεί να χρησιμοποιήσει την συγκεκριμένη εφαρμογή είναι όταν χρειάζεται μια εισαγωγή ενός μοντέλου που ήδη υπάρχει. Σε περίπτωση που ο χρήστης χρησιμοποιήσει το συγκεκριμένο εργαλείο πριν την οριστικοποίηση του αποτελέσματος, δημιουργεί ένα ολόκληρο σετ από δεδομένα για μια πιο ομαλή λειτουργία. [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

4.2.1.5 Extract ROI from dataset

Το υπο-μενού αυτό χρησιμοποιείται έτσι ώστε ο χρήστης σε περίπτωση που θέλει να απομονώσει ένα κομμάτι της εικόνας για να δουλέψει σε αυτό. Αφού λοιπόν πρώτα, από το αντίστοιχο παράθυρο ο χρήστης επιλέξει την εικόνα από την οποία θα απομονώσει κομμάτι της, στην συνέχεια εμφανίζεται ένα παράθυρο στο οποίο φαίνεται η εικόνα στο αριστερό μέρος, ενώ στο δεξί μέρος έχουμε τα εξής στοιχεία: το μέγεθος σε pixel της εικόνας, τις εικονοσυντεταγμένες των τεσσάρων άκρων του τετραγώνου που προσδιορίζει την προς εξαγωγή εικόνα καθώς επίσης και τις γεωγραφικές τους συντεταγμένες. Έτσι, ο χρήστης έχει απλά να επιλέξει την εικόνα που θέλει κεντράροντας το τετράγωνο στην περιοχή ενδιαφέροντος και απλά να πατήσει το save. Αυτή η ενέργεια μπορεί να γίνει είτε γραφικά πειράζοντας τα όρια του αρχικού τετραγώνου, είτε προσδιορίζοντας ακριβώς τα όρια του καινούργιου τετραγώνου πληκτρολογώντας τα. Αμέσως με την εκτέλεση της ενέργειας, θα δημιουργηθεί στο Monteverdi ένας δείκτης Extract ROI, ο οποίος θα περιέχει την απομονωμένη εικόνα. Καλό είναι το υπο-μενού αυτό να χρησιμοποιείται σε πολύ μεγάλες εικόνες, έτσι ώστε όλη η μετέπειτα επεξεργασία να γίνεται πολύ πιο γρήγορα.

4.2.1.6 Concatenate images

Με αυτή την λειτουργία ο χρήστης μπορεί να ενώσει εικόνες οι οποίες έχουν τις ίδιες διαστάσεις και να φτιάξει μια εικόνα με πολλά κανάλια. Έτσι ο χρήστης μπορεί για παράδειγμα να φτιάξει μια εικόνα που περιέχει την αρχική εικόνα σε συνδυασμό με την ίδια εικόνα έχοντας όμως εφαρμόσει ένα φίλτρο. Στην συνέχεια μπορεί να διαλέξει τον κατάλληλο κατά την γνώμη του συνδυασμό καναλιών στην νέα εικόνα και να τα προβάλει στην οθόνη ώστε να ολοκληρώσει την εφαρμογή του. Προκειμένου να γίνει αυτό, αρκεί να χρησιμοποιήσει το συγκεκριμένο μενού και απλά να φορτώσει όσα κανάλια από όποια εικόνα θέλει. Αυτόματα πάλι δημιουργείται ένας δείκτης concaterate ο οποίος περιέχει τόσα κανάλια όσα έχει περάσει ο χρήστης από την αντίστοιχη εικόνα.

4.2.1.7 Rasterize Vector data

Το συγκεκριμένο υπο-μενού χρησιμοποιείται για μετατροπή των vector δεδομένων σε raster. Ο χρήστης εισάγει την vector εικόνα και αυτή μετατρέπεται σε raster δεδομένα.

4.2.1.8 Export to Kmz

Με το υπο-μενού αυτό, ο χρήστης μπορεί να εξάγει τα δεδομένα του (είτε είναι raster είτε vector) στο Google Earth. Έτσι και αλλιώς το σύστημα έχει τις γεωγραφικές συντεταγμένες

των δεδομένων που εισάγει ο χρήστης καθώς αυτές βρίσκονται στα metadata της εικόνας. Έτσι ο χρήστης απλά φορτώνει την εικόνα από το μενού και συμπληρώνει όλα τα στοιχεία της.

4.2.1.9 Tile Map Import

Στο συγκεκριμένο υπο-μενού ο χρήστης μπορεί να δει τον χάρτη της περιοχής που θέλει. Με το μενού αυτό,ανοίγει ένα παράθυρο στο οποίο ο χρήστης εισάγει το όνομα της περιοχής και πατώντας το search το σύστημα προσδιορίζει τις γεωδαιτικές συντεταγμένες του ονόματος που εισάγει ο χρήστης και του εμφανίζει τον χάρτη που επιλέγει. Βέβαια στην περίπτωσή μας, το μενού αυτό είχε ένα πρόβλημα στην λειτουργία καθώς δεν μπορούσε να εμφανίσει σωστά τα αποτελέσματα στην οθόνη.

4.2.2 Visualisation

Σχήμα 4.3: Κεντρικό μενού Monteverdi για Visualisation

4.2.2.1 Viewer

Το συγκεκριμένου υπο-μενού είναι υπεύθυνο για την εμφάνιση των δεδομένων που εισάγει ο χρήστης στην οθόνη του. Έτσι λοιπόν από το παράθυρο που εμφανίζεται και αφού πρώτα έχει φορτωθεί η εικόνα στο σύστημα, ο χρήστης επιλέγει την εικόνα που θέλει να εμφανίσει στην οθόνη του, πατάει το (+) προκειμένου να φορτωθεί στο σύστημα και μετά ok. Στην συνέχεια εμφανίζονται δύο παράθυρα.

- **Standard image viewer**
  Στο συγκεκριμένο εργαλείο, εμφανίζεται η εικόνα του χρήστη. Στο πάνω αριστερά παράθυρο φαίνεται το σύνολο της εικόνας και η περιοχή στην οποία έχει εστιάσει ο χρήστης. Το παράθυρο αυτό χρησιμοποιείται κυρίως για γρήγορη πλοήγηση πάνω στην εικόνα (Scroll window). Στο ακριβώς από κάτω παράθυρο εμφανίζεται η περιοχή η οποία ο χρήστης έχει κάνει zoom, ενώ από κάτω έχουμε το ιστόγραμμα της εικόνας και παρακάτω την περιγραφή του κάθε pixel στο οποίο ο χρήστης ακουμπάει το ποντίκι του. Τέλος, το μεγάλο παράθυρο εμφανίζει το μέρος της εικόνας στο οποίο έχει εστιάσει ο χρήστης (Full resolution window), ενώ το παράθυρο που βρίσκεται μέσα στο τετράγωνο, εμφανίζει την περιοχή που έχει μεγεθυνθεί στα αριστερά.

- **Data Properies**
  Στο συγκεκριμένο παράθυρο βρίσκονται όλες οι επιλογές για την εικόνα όσον αφορά την οριστικοποίησή της. Το παράθυρο αυτό αποτελείται από τέσσερις καρτέλες.
  Η καρτέλα **Data** περιέχει γενικές πληροφορίες που αφορούν την εικόνα και γενικότερα εντολές που χρησιμοποιούνται κυρίως στους vector τύπους δεδομένων.
  Η καρτέλα **Setup** περιέχει τους συνδυασμούς των καναλιών που μπορεί να κάνει ο χρήστης. Έτσι ο χρήστης μπορεί να επιλέξει να εμφανίσει την εικόνα σε grayscale mode, ή να επιλέξει τον συνδυασμό καναλιών που χρειάζεται για την εφαρμογή του. Όλες οι ενέργειές του μπορούν να εμφανιστούν στην οθόνη με το πλήκτρο update channels.

Επίσης στην συγκεκριμένη καρτέλα, ο χρήστης μπορεί να επιλέξει την μέθοδο με την οποία γίνεται το contrast της εικόνας. Από μόνο του το πρόγραμμα έχει προεπιλεγμένη την μέθοδο Linear X% και τα όρια min +2% και max -2%. Τέλος, οι άλλες δύο επιλογές που υπάρχουν στην καρτέλα χρησιμεύουν για να χωρίσουν ή όχι τα παράθυρα που υπάρχουν και αναλύθηκαν στο Standard image viewer.
Η καρτέλα **Histogram** παρουσιάζει τα ιστογράμματα του κάθε καναλιού. Ο χρήστης μπορεί να ελέγξει και να αλλάξει την κατωφλίωση των ιστογραμμάτων απλά αλλάζοντας τις ακραίες θέσεις των γραμμών που περιβάλλουν το ιστόγραμμα.
Τέλος, η καρτέλα **Pixel Description** περιέχει όλες τις πληροφορίες για όποιο pixel της εικόνας θελήσει ο χρήστης. Οι πληροφορίες που παρέχονται στην συγκεκριμένη καρτέλα αφορούν τις διαστάσεις της εικόνας, τις τιμές του συγκεκριμένου pixel στα διάφορα κανάλια που είναι διαθέσιμα, τις τιμές του pixel στα κανάλια που απεικονίζονται, τις συντεταγμένες του σημείου (γεωγραφικό μήκος και γεωγραφικό πλάτος) και τέλος σε περίπτωση που ο χρήστης διαθέτει σύνδεση στο Internet εμφανίζει την περιοχή που υπολογίζεται ότι είναι η εικόνα που εισήγαγε ο χρήστης (χώρα και πόλη). Στις Ελληνικές περιοχές αναγνωρίζει την χώρα αλλά όχι την πόλη από την οποία προέκυψε η εικόνα.

4.2.2.2 SpectralViewer

Το υπο-μενού αυτό χρησιμεύει για να μπορέσει ο χρήστης να προσδιορίσει τις φασματικές υπογραφές για κάθε pixel της εικόνας. Με άλλα λόγια, εμφανίζει για κάθε pixel της εικόνας την τιμή της έντασής του σε κάθε κανάλι. Πάλι ο χρήστης έρχεται αντιμέτωπος με δύο παράθυρα.

- Spectrume analysis module
  Στο παράθυρο αυτό, ο χρήστης μπορεί να περιηγηθεί σε όλα τα pixel της εικόνας, ενώ στο κάτω μέρος του παραθύρου μπορεί να επιλέξει τον κατάλληλο συνδυασμό καναλιών για την απεικόνιση του αρχείου του. Το συγκεκριμένο παράθυρο είναι δομημένο με τον ίδιο περίπου τρόπο με το οποίο είναι δομημένο και το παράθυρο του Viewer

- Curve Display
  Στο παράθυρο αυτό εμφανίζεται το διάγραμμα που δείχνει την τιμή της έντασης σε κάθε ένα κανάλι του κάθε pixel. Ο χρήστης μπορεί να αλλάξει τον τρόπο που εμφανίζεται η κλίμακα στον άξονα των y, όπως επίσης και το zoom στο οποίο φαίνεται το διάγραμμα.

4.2.2.3 Color Mapping

Το συγκεκριμένο εργαλείο κατασκευάζει τον χρωματικό χάρτη του αρχείου που εισάγει ο χρήστης. Ο χρήστης εισάγει την εικόνα και επιλέγει το χρώμα με το οποίο θέλει να γίνει ο χάρτης.

4.2.3 Calibration

Σχήμα 4.4: Κεντρικό μενού Monteverdi για Calibration

Όπως αναφέρθηκε και παραπάνω, το συγκεκριμένο μενού επιλογών χρησιμοποιείται για να γίνει διόρθωση της εικόνας από την ατμόσφαιρα. Η διόρθωση αυτή μπορεί να γίνει τόσο σε δορυφορικά δεδομένα (Optical Calibration), όσο και σε δεδομένα ραντάρ (SAR Calibration). Βέβαια στα δορυφορικά δεδομένα υποστηρίζονται μόνο αρχεία που ανήκουν στους δορυφόρους Ikonos-2, Spot4-5 και Quickbird. Στην περίπτωση της εικόνας που χρησιμοποιήθηκε για την εφαρμογή αυτή όμως, παρόλο που άνηκε στον δορυφόρο Quickbird, το σύστημα δεν την αναγνώρισε και δεν μπόρεσε να κάνει διόρθωση των σφαλμάτων της ατμόσφαιρας λόγο μη πληρότητας δεδομένων στα metadata της.

Λίγο αναλυτικότερα τώρα, για τα οπτικά δεδομένα, η βασική ιδέα είναι η διόρθωση της ανακλαστικότητας από τα παρατηρούμενα φυσικά αντικείμενα. Η όλη διαδικασία μπορεί να χωριστεί σε 3 βασικά βήματα: αρχικά να γίνει άντληση της φωτεινότητας από τις τιμές της εικόνας που εισάγεται, στην συνέχεια μετατρέπεται η φωτεινότητα σε ανακλαστικότητα προκειμένου να παραχθεί μια TOA εικόνα (Top Of Atmosphere) και τέλος προσομοιώνεται η ανακλαστικότητα του ήλιου από το ζευγάρι ατμόσφαιρα- επιφάνεια γης. Στο τελευταίο στάδιο δημιουργείται μια εικόνα TOC (Top of Canopy) η οποία είναι και το τελικό αποτέλεσμα της διόρθωσης της ατμόσφαιρας σε οπτικά δεδομένα.

Τέλος, για τα SAR δεδομένα η διόρθωση λόγω της ατμόσφαιρας είναι πολύ σημαντική και επηρεάζει το τελικό αποτέλεσμα και την ακρίβειά του. Το μοντέλο που χρησιμοποιείται στην συγκεκριμένη περίπτωση επιτρέπει την εκτίμηση της ακρίβειας του αποτελέσματος ποσοτικά. Προς το παρόν, είναι διαθέσιμη μόνο η ατμοσφαιρική διόρθωση για δεδομένα TerraSARX. [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

4.2.4 Filtering

Σχήμα 4.5: Κεντρικό μενού Monteverdi για Filtering

Πρόκειται για το μενού επιλογών που περιέχει τα περισσότερα τηλεπισκοπικά εργαλεία του συστήματος.

4.2.4.1 BandMath

Με το συγκεκριμένο υπο-μενού, ο χρήστης μπορεί να κάνει οποιαδήποτε σύνθετη μαθηματική πράξη μεταξύ των καναλιών που περιέχει η εικόνα. Το μοντέλο αυτό στηρίζεται στην μαθηματική βιβλιοθήκη muParser και περιέχει ένα πλήθος από συναρτήσεις και λειτουργίες. Με το εργαλείο αυτό δηλαδή ο χρήστης μπορεί να κατασκευάσει όλους τους λόγους καναλιών αλλά και να πραγματοποιήσει οποιαδήποτε πράξη σε οποιοδήποτε κανάλι. Έτσι απλά φορτώνει την εικόνα που θέλει στο σύστημα, και στο παράθυρο που δημιουργείται απλά συμπληρώνει την 'φόρμουλα', δηλαδή την πράξη που θέλει να κάνει. Το εργαλείο αυτό προφυλάσσει επίσης και από λάθη στην μαθηματική φόρμουλα, ελέγχοντας την έκφραση κατά την διάρκεια που ο χρήστης την πληκτρολογεί και τον προειδοποιεί για το λάθος. Αφού πραγματοποιηθεί αυτό το

βήμα, αυτομάτως δημιουργείται ο δείκτης, BathMath που περιέχει το αποτέλεσμα της εικόνας που έχει δημιουργηθεί.

4.2.4.2 Thershold

Το υπο-μενού αυτό αποτελεί ένα ακόμα εργαλείο για κατωφλίωση της εικόνας. Και σε αυτή την περίπτωση ο χρήστης εισάγει την εικόνα στο σύστημα και του εμφανίζεται αμέσως ένα παράθυρο στο οποίο μπορεί να κάνει όλες του τις ενέργειες. Έτσι έχει την δυνατότητα να επιλέξει είτε την Generic Threshold είτε την Binary Threshold. Επίσης είναι σε θέση να πραγματοποιήσει κατωφλίωση είτε στο στο πάνω άκρο, είτε στο κάτω άκρο είτε και στα δύο άκρα. Η κατωφλίωση είναι μια πολύ σημαντική διαδικασία της τηλεπισκόπησης, καθώς εμφανίζει με καλύτερο τρόπο τα αποτελέσματα και τα τονίζει έτσι ώστε ο τελικός χρήστης να μπορεί να τα προσδιορίσει εύκολα.

4.2.4.3 Pansharpening

Λόγω του τρόπου κατασκευής των αισθητήρων, είναι πολύ δύσκολο να επιτευχθεί ταυτόχρονα υψηλή ποιότητα σε χωρική αλλά και σε φασματική ανάλυση, καθώς μια καλύτερη χωρική ανάλυση σημαίνει μικρότερο ανιχνευτή,το οποίο με την σειρά του συνεπάγεται μικρότερη οπτική ροή στην επιφάνεια του ανιχνευτή. Από την άλλη μεριά, τα φασματικά κανάλια επιτυγχάνονται μέσα από φίλτρα τα οποία εφαρμόζονται στην επιφάνεια του ανιχνευτή και τα οποία μειώνουν την οπτική ροή έτσι ώστε να αυξήσουν το μέγεθος του ανιχνευτή και να επιτευχθεί η επιθυμητή αναλογία θορύβου στο σήμα.
Για τους παραπάνω λόγους, πολλές δορυφορικές εικόνες υψηλής ευκρίνειας είναι εξοπλισμένες με δύο σετ από ανιχνευτές, οι οποίοι παράγουν δύο ξεχωριστά είδη εικόνων. Το πρώτο είδος είναι οι πολυφασματικές εικόνες (XS) οι οποίες είναι εξοπλισμένες από 3 εώς 8 φασματικά κανάλια, περιέχοντας συνήθως το μπλε, πράσινο, κόκκινο και εγγύς υπέρυθρο κανάλι στην συγκεκριμένη ανάλυση (συνήθως από 2.8 μέτρα εώς 2 μέτρα). Το δεύτερο είδος αποτελούν οι πανχρωματικές εικόνες οι οποίες είναι εικόνες στην κλίμακα του γκρι (grayscale). Οι συγκεκριμένες εικόνες χρησιμοποιούν ανιχνευτές οι οποίοι καλύπτουν ένα μεγάλο τμήμα του φωτεινού φάσματος έτσι ώστε να αυξήσουν την οπτική ροή και να μειώσουν το μέγεθος των pixel. Για το λόγο αυτό η ανάλυση στις παγχρωματικές εικόνες είναι συνήθως περίπου 4 φορές καλύτερη από αυτή των πολυφασματικών εικόνων (φτάνουν τα 46 με 70 cm).
Είναι πολύ συχνό φαινόμενο οι δύο αυτές εικόνες να δίνονται μαζί ως δεδομένα. Τέτοιου είδος εικόνες ονομάζονται bundle. Έτσι και αλλιώς αποτελεί ένα πολύ κοινό πρόβλημα της τηλεπισκόπησης το πως να συνδυάσει τις πανχρωματικές εικόνες με τις πολυφασματικές έτσι ώστε να ταιριάξει την πολύ καλή χωρική πληροφορία που παρέχεται από τις τελευταίες με τον φασματικό πλούτο που παρέχουν οι πρώτες. Η διαδικασία αυτή ονομάζεται pan-sharpening. [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]
Το συγκεκριμένο υπο-μενού λοιπόν του Monteverdi προσφέρει μια βήμα προς βήμα διαδικασία για την συγκεκριμένη λειτουργία. Αρχικά το σετ των εικόνων (παγχρωματικές και πολυφασματικες) φορτώνονται στο σύστημα από το αντίστοιχο μενού. Στην συνέχεια χρησιμοποιείται το μοντέλο του Superimpose προκειμένου να εντοπιστεί και να συγκριθεί η πολυφασματική με την πανχρωματική εικόνα. Ως αποτέλεσμα προκύπτει μια νέα πολυφασματική εικόνα η οποία έχει την ίδια γεωγραφική περιοχή και την ίδια ανάλυση με αυτή της παγχρωματικής εικόνας. Τέλος, ο χρήστης μπορεί να χρησιμοποιήσει το μοντέλο του Simple RCS pan-sharpening χρησιμοποιώντας τόσο την παγχρωματική όσο και την πολυφασματική εικόνα και παράγοντας ακριβώς το ίδιο αποτέλεσμα.

4.2.4.4 Mean shift clustering

Με το υπο-μενού αυτό, το σύστημα πραγματοποιεί κατάτμηση της εικόνας αλλά μπορεί να χρησιμοποιηθεί και ως εξομάλυνση όλων των ακμών. Η κατάτμηση της εικόνας χρειάζεται

για να δημιουργηθούν αντικείμενα στην εικόνα, όπου με την σειρά τους αυτά, ταξινομούνται με κάποιο τρόπο ταξινόμησης. Ο τρόπος με τον οποίο δουλεύει ο συγκεκριμένος αλγόριθμος είναι ο εξής: για κάθε pixel ο αλγόριθμος δημιουργεί ένα μπλοκ από γειτονικά pixel που περιέχονται μέσα στην χωρική ακτίνα και το εύρος χρώματος που επιλέγει ο χρήστης. Στην συνέχεια τα παραπάνω χαρακτηριστικά για το μπλόκ υπολογίζονται και ο αλγόριθμος συνεχίζει με αυτά τα νέα στοιχεία. Αφού φορτωθεί η εικόνα, εμφανίζεται στην οθόνη ένα παράθυρο που περιέχει την περιοχή ενώ κάτω δεξιά έχει μια σειρά από παραμέτρους. Προκειμένου να τρέξει η κατάτμηση αρκεί ο χρήστης να πατήσει το πλήκτρο run. Με το Display Boundaries εμφανίζονται στην οθόνη τα όρια των αντικειμένων που δημιουργούνται στην εικόνα, ενώ με το Display Cluster εμφανίζονται στην οθόνη τα pixel των αντικειμένων ομαδοποιημένα. Όσον αφορά τώρα τις παραμέτρους, αφορούν το μέγεθος των αντικειμένων που χρειάζεται ο χρήστης να δημιουργήσει. Ανάλογα με το πόσο μεγάλη είναι η εικόνα, χρειάζεται κάποιο χρόνο για να πραγματοποιήσει την κατάτμηση. Στο τέλος, και αφού ο χρήστης κάνει την κατάτμηση που θέλει, δημιουργείται ο αντίστοιχος δείκτης που προσδιορίζει την κατατμημένη εικόνα. [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

4.2.4.5 Feature Extraction

Πρόκειται για το υπο-μενού που περιέχει όλα τα φίλτρα που υπάρχουν στο σύστημα και χρησιμοποιεί διάφορες τεχνικές που σκοπό έχουν την εξαγωγή και ανίχνευση πληροφοριών από τις εικόνες. Τα αντικείμενα που εξάγει μπορεί να είναι σημεία, γραμμές κ.τ.λ. αλλά και μετακινήσεις, υφή κ.τ.λ. Κατά τα γνωστά, ο χρήστης εισάγει την εικόνα του στο σύστημα και στην συνέχεια εμφανίζεται ένα παράθυρο που περιέχει τα εξής χαρακτηριστικά. Αρχικά, αποτελείται από τρία μικρά παράθυρα από τα οποία τα δύο πρώτα περιέχουν την εικόνα που εισήγαγε ο χρήστης, ενώ το τρίτο περιέχει μια μικρή επισκόπηση του αποτελέσματος όπως προκύπτει από το φίλτρο που εφαρμόζεται. Στην συνέχεια και όσον αφορά την καρτέλα action, ο χρήστης προσδιορίζει το φίλτρο της αρεσκείας του. Τα φίλτρα που υπάρχουν στο σύστημα είναι τα παρακάτω.

Φίλτρο	**Λεπτομέρειες**
Touzi	—
Harris	—
Spectral Angle	—
Variance	—
Mean	—
Rec. Gradient	—
Texture	PanTex SFS Haralick1 Haralick2
Morphology	Morphological opening Morphological closing
Radiometry Indexes	Vegetation Soil Built up Water
Edge Density	Sobel
Mean Shift	Smooth Clusters Labels
	Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.1 – συνέχεια από προηγούμενη σελίδα

Φίλτρο	Λεπτομέρειες
	Bountries

Πίνακας 4.1: Διαθέσιμα φίλτρα στο Monteverdi

Ανάλογα με το φίλτρο που διαλέγει ο χρήστης, μπορεί να επιλέξει τα κανάλια στα οποία θα το εφαρμόσει, τις παραμέτρους του και απλά να πατήσει το κουμπί add. Με αυτό τον τρόπο όλες οι μετατροπές υπολογίζονται στο σύστημα. Στην συνέχεια και αφού ο χρήστης κατευθυνθεί στην καρτέλα Output, επιλέγοντας κάποιες από τις παραμέτρους, εμφανίζεται στο τρίτο παράθυρο μια επισκόπηση του αποτελέσματος έτσι ώστε να το ελέγξει ο χρήστης, και άμα χρειάζεται να αλλάξει κάποια από τις παραμέτρους.
Τέλος, όπως σε όλες τις ενέργειες που πραγματοποιούνται στο σύστημα, δημιουργείται ο αντίστοιχος δείκτης που περιέχει την φιλτραρισμένη εικόνα.

4.2.4.6 Change Detection

Το συγκεκριμένο υπο-μενού χρησιμοποιείται για να ανιχνεύσει μεταβολές του περιβάλλοντος στην εικόνα. Ο χρήστης, πρέπει να φορτώσει στο σύστημα τις δύο εικόνες έτσι ώστε να αρχίσει την επεξεργασία.

4.2.4.7 Fine Correlation

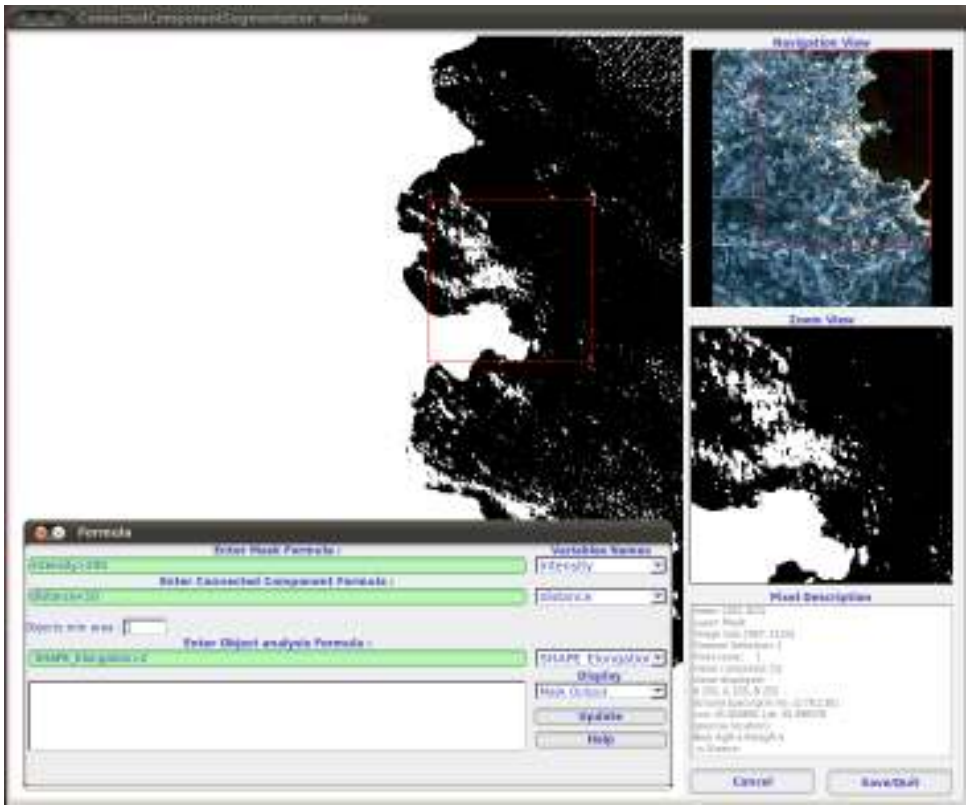
4.2.4.8 Vectorization

Αφορά τις εικόνες τύπου vector.

4.2.4.9 Connected Component Segmentation module

Το συγκεκριμένο μοντέλο επιτρέπει την κατάτμηση και ανάλυση σε αντικείμενα χρησιμοποιώντας κριτήρια που προσδιορίζονται από τον χρήστη σε κάθε βήμα. Το μοντέλο αυτό χρησιμοποιεί πάλι την βιβλιοθήκη muParser ενώ πραγματοποιείται σε τρία βήματα.
Στο πρώτο βήμα διαμορφώνεται η 'μάσκα'. Η συγκεκριμένη 'μάσκα' χρησιμοποιείται από το Connected Component segmentation. Αυτό το βήμα είναι προαιρετικό, καθώς σε περίπτωση που δεν δοθεί καμία 'μάσκα', η όλη διαδικασία γίνεται για όλη την εικόνα. Παράδειγμα ενός τέτοιου κριτηρίου είναι το intensity>200, όπου χρησιμοποιείται το κριτήριο της έντασης και συμμετέχουν όλα τα pixel που έχουν τιμή έντασης πάνω από 200.
Στο δεύτερο βήμα ακολουθεί η κατάτμηση. Η κατάτμηση μπορεί να ακολουθείται από ένα βήμα όπου ό,τι δεν συμμετέχει απορρίπτεται. Για παράδειγμα ο χρήστης μπορεί να χρησιμοποιήσει ένα κριτήριο απόστασης προκειμένου να δεχτεί ή να απορρίψει τα αντίστοιχα pixel.
Σε τελικό στάδιο γίνεται η τελική επιλογή με χρήση κάποιου στατιστικού ή κριτηρίου απόστασης. Όλα τα διαθέσιμα κριτήρια βρίσκονται δίπλα ακριβώς από την έκφραση. Έτσι ο χρήστης μπορεί πιο εύκολα να επιλέξει το κριτήριο που θέλει κάθε στιγμή.

Τα αποτελέσματα εξάγονται σε shape file format. Το παράθυρο που εμφανίζονται τα αποτελέσματα αναβαθμίζεται κάθε φορά που ο χρήστης πατάει το ανάλογο κουμπί. Επίσης, τα αποτελέσματα που μπορούν να εμφανιστούν είναι: η εικόνα που εισήχθηκε στο σύστημα (input image), η εικόνα 'μάσκα' που δημιουργείται από τα κριτήρια (mask output), η εικόνα που προκύπτει από συνένωση της εικόνας με την 'μάσκα' (masked image), η εικόνα που προκύπτει από την εφαρμογή του φίλτρου (Segmentation output), η εικόνα που προκύπτει από το φίλτρο και αποκλείει τα μικρά αντικείμενα (Segmentation after small object rejection) και τέλος η εικόνα που προκύπτει από την αντικειμενοστραφή ανάλυση (Filter output). Τέλος,



Σχήμα 4.6: Παράδειγμα χρήσης του εργαλείου Connected Component Segmentation

αφού ο χρήστης ορίσει όλες τις παραμέτρους που χρειάζεται, απλά κάνει ένα Save and Quit στο παράθυρο και το αποτέλεσμα εξάγεται σε μορφή vector.

4.2.5 SAR

Σχήμα 4.7: Κεντρικό μενού Monteverdi για SAR

Το συγκεκριμένο μενού επιλογών περιέχει όλα τα απαραίτητα εργαλεία για επεξεργασία εικόνων ραντάρ. Τα επιμέρους εργαλεία παρουσιάζονται παρακάτω.

4.2.5.1 Despeckle Image

Οι εικόνες SAR συνήθως περιέχουν κάποιο θόρυβο σαν σημαδάκια πάνω στην εικόνα. Προκειμένου ο θόρυβος αυτός να μειωθεί και οι εικόνες να βελτιωθούν προτείνονται πλήθος από φίλτρα και τεχνικές. Το συγκεκριμένο μοντέλο χρησιμοποιεί μια πολύ διαδεδομένη μέθοδο για τον λόγο αυτό, την Frost and Lee.

4.2.5.2 Compute intensity and log-intensity

Το συγκεκριμένο υπο-μενού υπολογίζει την ένταση και τη log ένταση από τις εικόνες SAR που εισάγονται.

4.2.5.3 Polarimetric Synthesis

Η πόλωση είναι η γραμμική μέτρηση (HH, HV, VH, VV) της πολικότητας η οποία επιτρέπει την μέτρηση πλήθους οπτικών ιδιοτήτων των αντικειμένων. Στην πολικότητα, η βασική μέτρηση είναι ένας πολύπλοκος πίνακας 2x2 που αποδίδεται σε έναν χώρο οκτώ διαστάσεων (Sinclair matrix). Για αμοιβαίους τρόπους όπου HV=VH, ο χώρος αυτός μετατρέπεται σε 5 διαστάσεων όπου οι τρεις αποτελούν τους $\|HH\|, \|HV\|, \|VV\|$ και οι άλλοι δύο τους co-pol HH-VV και cross-pol HH-HV.
Συνοπτικά τα εργαλεία που χρησιμοποιούνται στο συγκεκριμένο υπο-μενού είναι τα ακόλουθα:
Synthesis: Το συγκεκριμένο εργαλείο επιτρέπει την κατασκευή μιας εικόνας που έχει προέρθει από ένα πολικό ραντάρ, έχοντας επιλέξει να μεταδίδουν και να παίρνουν τις πολικότητες. Το συγκεκριμένο μοντέλο ασχολείται με πραγματικά μέρη της εικόνας για τις HH, VV, VH, HV εικόνες. Προς το παρόν η περίπτωση όπου HV=VH δεν έχει κατασκευαστεί ακόμα, αλλά ο χρήστης μπορεί να εισάγει τις δύο τιμές ξεχωριστά.
Conversion: Το συγκεκριμένο εργαλείο χρησιμοποιείται προκειμένου να αλλάξει ο πολύπλοκος πίνακας 2x2 οχτώ διαστάσεων. Τα μοντέλα που υπάρχουν σε αυτό το εργαλείο επιτρέπουν τις αλλαγές μεταξύ των πινάκων καθώς επίσης και των τύπων των εικόνων που εισάγονται.
Analysis: Το συγκεκριμένο εργαλείο χρησιμοποιείται για την παράσταση της κλασικής πολιμετρικής ανάλυσης. Επιτρέπει τον υπολογισμό της πολιμετρικής σύνθεσης (εισαγωγή εικόνας 4 καναλιών, εξαγωγή μονοκάναλης πραγματικής εικόνας και καθορισμός των παραμέτρων από τον χρήστη) αλλά και τον υπολογισμό της H alpha εικόνας (εισαγωγή εικόνας 6 καναλιών και εξαγωγή πραγματικής εικόνας 3 καναλιών). [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

4.2.6 Learning

Σχήμα 4.8: Κεντρικό μενού Monteverdi για Learning

Στο συγκεκριμένου μενού επιλογών βρίσκονται όλες οι ταξινομήσεις που διαθέτει το σύστημα.

4.2.6.1 SVM Classification

Η επιβλεπόμενη ταξινόμηση αποτελεί μια διαδικασία με την οποία κάθε ξεχωριστό στοιχείο ταξινομείται στην αντίστοιχη κλάση με βάση την πληροφορία που έχουν ένα ή περισσότερα χαρακτηριστικά της αλλά και με βάση τις περιοχές εκπαίδευσης που έχει επιλέξει ο χρήστης. Το μοντέλο της επιβλεπόμενης ταξινόμησης που χρησιμοποιεί το σύστημα είναι βασισμένο στη μέθοδο Support Vector Machine, η οποία χρησιμοποιεί την έρευνα μεταξύ δύο ξεχωριστών κλάσεων με βάση τα δείγματα εκπαίδευσής τους. Η μέθοδος μπορεί να εφαρμοστεί και για ταξινομήσεις με παραπάνω από 2 κλάσεις. Σε κάθε περίπτωση, ο χρήστης φορτώνει την εικόνα του στο σύστημα και με βάση το παράθυρο που δημιουργείται, δημιουργεί κλάσεις (κατηγορίες), φτιάχνοντας πολύγωνα για κάθε κλάση. Με αυτό τον τρόπο το σύστημα πραγματοποιεί

μια ταξινόμηση με βάση τα χαρακτηριστικά της κάθε κλάσης, έτσι όπως προκύπτουν από τις περιοχές εκπαίδευσης που επέλεξε ο χρήστης και κάθε pixel ταξινομείται στην αντίστοιχη κατηγορία.

4.2.6.2 SVM Classification

Το υπο-μενού αυτό, πραγματοποιεί την ίδια δουλειά με το παραπάνω, με την μόνη διαφορά ότι ο χρήστης σε αυτή την περίπτωση πρέπει να φορτώσει μαζί με την εικόνα και τις περιοχές εκπαίδευσης. Με άλλα λόγια, χρησιμοποιείται στην περίπτωση που ο χρήστης έχει έτοιμες τις περιοχές εκπαίδευσης της εικόνας και απλά θέλει να τρέξει τον αλγόριθμο.

4.2.6.3 KMeans clustering

Το υπο-μενού αυτό αποτελεί μια μη επιβλεπόμενη ταξινόμηση με χρήση του αλγορίθμου KMeans. Ο χρήστης εισάγει την εικόνα και στο παράθυρο που εμφανίζεται επιλέγει πόσες κλάσεις θέλει να δημιουργήσει ο αλγόριθμος, πόσα δείγματα θα πάρει από την εικόνα και ποιο θα είναι το max nb of iterators. Έτσι, αφού τελειώσει η όλη επεξεργασία του αλγορίθμου, δημιουργείται και ο αντίστοιχος δείκτης, που περιέχει την εικόνα πού έχει ταξινομηθεί.

4.2.6.4 Object Labeling

4.2.7 Geometry

Σχήμα 4.9: Κεντρικό μενού Monteverdi για Geometry

Στην διαδικασία της ψηφιακής τηλεπισκόπησης, μια πολύ κοινή διαδικασία είναι ο χειρισμός δεδομένων που προέρχονται από διαφορετικές πηγές. Το συγκεκριμένο μενού δίνει την δυνατότητα πρόσβασης σε μια μεγάλη σειρά από γεωμετρικές διορθώσεις. Προσφέρει γεωμετρική διόρθωση σε οπτικά αλλά και SAR δεδομένα, χρησιμοποιώντας τα υπάρχοντα μοντέλα (οι πληροφορίες της εικόνας που βρίσκονται στα metadata αυτής διαβάζονται αυτόματα από την εφαρμογή). [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

4.2.7.1 Reproject Image

Το συγκεκριμένο υπο-μενού χρησιμοποιείται προκειμένου ο χρήστης να κάνει ορθή προβολή της εικόνας. Αυτή η εφαρμογή μπορεί να αναγνωρίζει τα metadata και να θέτει αυτόματα τις παραμέτρους.
Αφού λοιπόν εισάγει το αρχείο, εμφανίζεται ένα παράθυρο το οποίο περιέχει 4 επιμέρους καρτέλες. Στην πρώτη καρτέλα (Input Image), το σύστημα ενημερώνει τον χρήστη για τα γενικά χαρακτηριστικά της εικόνας, όπως τις γεωδαιτικές συντεταγμένες των ακραίων σημείων της εικόνας ή του κέντρου της. Στην δεύτερη καρτέλα του παραθύρου (Output Image), ο χρήστης έχει την δυνατότητα να ενημερωθεί για τις εικονοσυντεταγμένες των συγκεκριμένων σημείων, καθώς επίσης και τις καρτεσιανές τους συντεταγμένες σε όποιο από τα παρεχόμενα συστήματα θέλει. Το γεωγραφικό πλάτος και μήκος υπολογίζεται από τα metadata της εικόνας. Επίσης από την συγκεκριμένη καρτέλα, ο χρήστης μπορεί να επιλέξει το σύστημα προβολής για την

εικόνα που θα δημιουργηθεί. Η επόμενη καρτέλα (Setting), δίνει την δυνατότητα στον χρήστη να διαλέξει τον interpolator που θεωρεί ότι είναι καλύτερος για το σύστημά του. Επίσης μπορεί να επιλέξει και χρήση του DEM, δίνει την δυνατότητα δηλαδή στον χρήστη να εισάγει το ψηφιακό μοντέλο του εδάφους για το αρχείο που έχει φορτώσει στο σύστημα. Τέλος, στην τελευταία καρτέλα ο χρήστης μπορεί να κάνει μια επισκόπηση του αποτελέσματος που προκύπτει από τις ενέργειες που έδωσε στο σύστημα.

4.2.7.2 Superimpose two images

Το συγκεκριμένο υπο-μενού μπορεί να χρησιμοποιηθεί για να προβάλει μια εικόνα στην γεωμετρία της άλλης. Ο χρήστης πρέπει να εισάγει τις δύο εικόνες και το σύστημα θα προβάλει την γεωμετρία της δεύτερης εικόνας (Reference Image to Reprojection) στην πρώτη (Image to Reproject).

4.2.7.3 Homologous points extraction

Με το υπο-μενού αυτό, ο χρήστης μπορεί να προσδιορίσει ομόλογα σημεία στις δύο εικόνες χειροκίνητα, και το σύστημα θα δώσει στον χρήστη το λάθος που υπολογίζεται ότι κάνει. Με τον τρόπο αυτό ο χρήστης ελέγχει την γεωμετρία του μετασχηματισμού των δύο εικόνων.

4.2.7.4 GCP to sensor model

Το συγκεκριμένο υπο-μενού είναι παρόμοιο με το παραπάνω, με την μόνη διαφορά πως αυτή την φορά ο χρήστης εισάγει μόνο μια εικόνα, και προσπαθεί να βρει τα σημεία ελέγχου στο έδαφος και σε αυτή την εικόνα. Με λίγα λόγια, το σύστημα με τις δυνατότητες που του παρέχει, μπορεί να προσδιορίσει το σημείο στον αντίστοιχο χάρτη και από εκεί ο χρήστης να το προσδιορίσει με περισσότερη ακρίβεια. Με αυτό τον τρόπο η γεωμετρία της εικόνας διορθώνεται αφού η γεωαναφορά γίνεται με τον σωστό τρόπο. Η συνάρτηση που χρησιμοποιείται για τον λόγο αυτό είναι βασισμένη στο RPC transformation. Παρόλα αυτά, στο σημείο αυτό πρέπει να αναφερθεί πως με την εικόνα της περιοχής της Αχλάδας, ο χάρτης δεν μπορούσε να εστιάσει με τον σωστό τρόπο. Αντίθετα για την περιοχή του Ηρακλείου όπως φαίνεται και παρακάτω το σύστημα προσδιόρισε την περιοχή έτσι όπως έπρεπε. [The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-3.10, 2011]

Σχήμα 4.10: Γεωαναφορά της περιοχής του Ηρακλείου

Υπάρχουν δύο τρόποι για δημιουργία των σημείων GCP. Ο πρώτος γίνεται με σύνδεση στο Internet όπου δυναμικά γίνεται η ταυτοποίηση της περιοχής με τα Open Street Map layers, ενώ ο δεύτερος επιτυγχάνεται χωρίς σύνδεση στο Internet όπου ο χρήστης δίνει χειροκίνητα σημεία ελέγχου, υποδεικνύοντας την θέση τους μέσα στην εικόνα.
Η εφαρμογή μπορεί επίσης να εισάγει και να εξάγει την λίστα με τα σημεία ελέγχου από και σε XML αρχείο. Παράλληλα, άμα τα σημεία ελέγχου βρίσκονται στα metadata της εικόνας, τότε το μοντέλο επιτρέπει την εισαγωγή και την αφαίρεση σημείων από την ήδη υπάρχουσα λίστα, η οποία αυτόματα αναγνωρίζεται.

4.2.7.5 DEM To Image Generator

Το ψηφιακό μοντέλο εδάφους, είναι εικόνα με γεωαναφορά του υψομέτρου. Είναι απαραίτητο εργαλείο για πλήθος εφαρμογών όπως για παράδειγμα, η παραγωγή ορθοκανονικών εικόνων. Το υπο-μενού αυτό είναι ένα ακόμα για την εισαγωγή του ψηφιακού μοντέλου εδάφους σε μια εικόνα. Ο χρήστης εισάγει την εικόνα στην οποία αναφέρεται το DEM, και στην συνέχεια εισάγει το αρχείο του ψηφιακού μοντέλου εδάφους για να προσαρμοστεί πάνω σε αυτή.

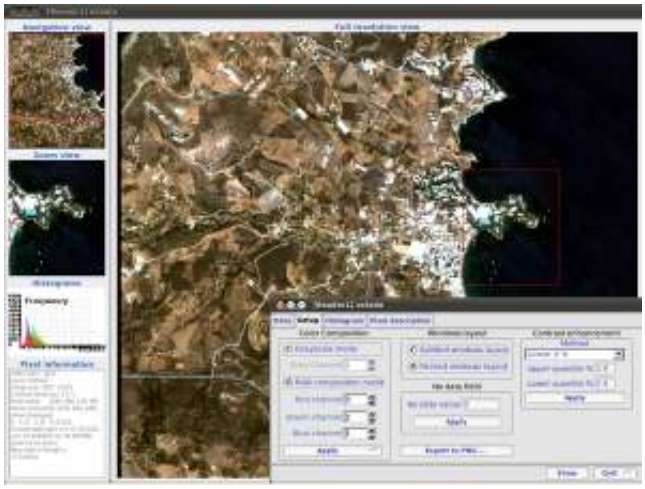
4.2.7.6 VectorData Trasform

Το συγκεκριμένο εργαλείο χρησιμοποιείται για την γεωμετρική διόρθωση μιας εικόνας τύπου Vector με βάση μια εικόνα raster.

4.3 Εμφάνιση εικόνας

Στο σημείο αυτό και αφού αναφέρθηκαν όλα τα διαθέσιμα εργαλεία για την επεξεργασία των δορυφορικών ή ακόμα και των εικόνων radar, ακολουθεί η εφαρμογή των εργαλείων αυτών στην εικόνα της περιοχής Αχλάδα του Ηρακλείου Κρήτης. Η εικόνα αυτή προέρχεται από τον δορυφόρο Quickbird, οπότε περιέχει 4 κανάλια καθώς επίσης είναι μεγάλης διακριτικής ικανότητας. Τέλος στην συγκεκριμένη εικόνα δεν χρειάζεται να κάνουμε καμία γεωμετρική διόρθωση καθώς είναι ήδη γεωμετρικά διορθωμένη.

Από το αντίστοιχο υπο-μενού λοιπόν, φορτώθηκε η εικόνα στο σύστημα ενώ αντίστοιχα, εμφανίστηκε και στην οθόνη του χρήστη. Το λογισμικό αναγνωρίζει αμέσως πως η εικόνα που εισήχθηκε είναι δορυφορική.



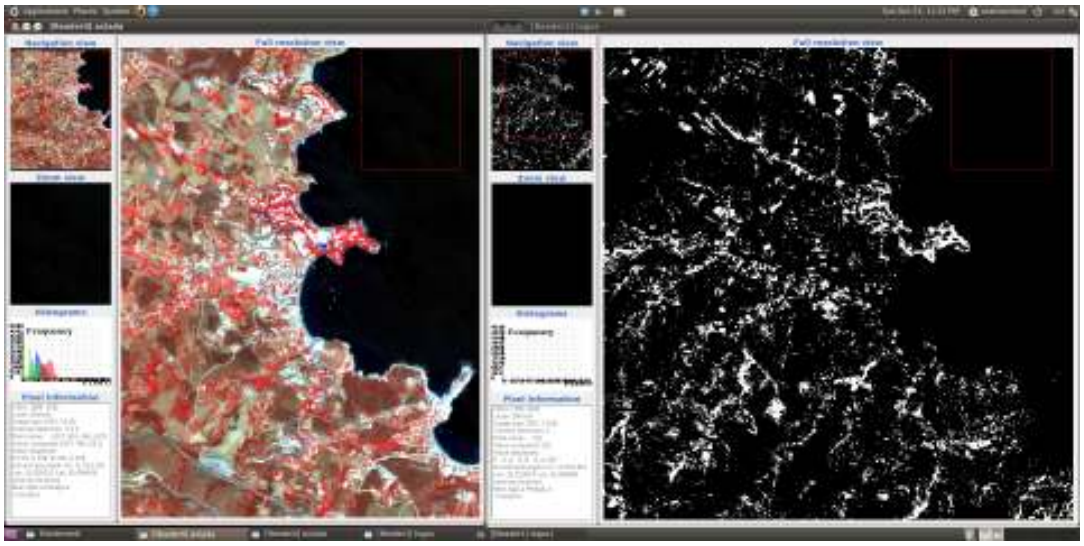
Σχήμα 4.11: Εφαρμογή ανοίγματος και εμφάνισης εικόνας QuickBird για την περιοχή της Αχλάδας

4.4 Εφαρμογή λόγων

Οι λόγοι καναλιών είναι ενισχύσεις, οι οποίες προκύπτουν από την διαίρεση των ψηφιακών τιμών σ΄ ένα ψηφιακό κανάλι, με τις αντίστοιχες τιμές σ΄ ένα άλλο κανάλι. Ένα πολύ μεγάλο πλεονέκτημα των λόγων είναι ότι αναδεικνύουν τα φασματικά χαρακτηριστικά των διαφόρων στοιχείων της εικόνας ανεξάρτητα από την έκταση των συνθηκών φωτισμού της περιοχής που θέλουμε. [Αργιαλάς, 1998]

Προκειμένου να επιλεγεί ο κατάλληλος λόγος, πρέπει να εντοπιστούν τα κανάλια στα οποία η θεματική ενότητα που ενδιαφέρει τον χρήστη έχει την μεγαλύτερη διαφορά ανακλαστικότητας. Στον αριθμητή συνήθως επιλέγεται το κανάλι στο οποίο η θεματική ενότητα που αναζητάτε έχει την μεγαλύτερη τιμή ανακλαστικότητας έτσι ώστε το αποτέλεσμα της να εμφανιστεί με λευκό χρώμα, ενώ όλα τα υπόλοιπα θα εμφανιστούν με μαύρο. Ανάλογα οπότε με την θεματική ενότητα που αναζητά ο χρήστης υπάρχουν και οι αντίστοιχοι λόγοι. Έτσι προκειμένου να εντοπιστεί η βλάστηση, χρησιμοποιείται ο λόγος 4/3, ενώ για να εντοπιστούν οι υδάτινες επιφάνειες χρησιμοποιείται ο λόγος 1/4, όπως επίσης ο ίδιος λόγος μπορεί να χρησιμοποιηθεί και για τις αστικές περιοχές.

Στην συγκεκριμένη περίπτωση εφαρμόστηκε ο δείκτης βλάστησης προκειμένου να εμφανιστούν τα αποτελέσματα που προκύπτουν. Έτσι στο Σχήμα 4.12 εμφανίζονται η αρχική εικόνα στα αριστερά και η εικόνα που προέκυψε από τον λόγο βλάστησης στα δεξιά. Προκειμένου να προκύψει το παρακάτω αποτέλεσμα και να γίνει περισσότερο κατανοητό, πραγματοποιήθηκε και ενίσχυση του ιστογράμματος της εικόνας που προέκυψε.



Σχήμα 4.12: Εφαρμογή λόγου βλάστησης NDVI σε εικόνα QuickBird της περιοχής Αχλάδας

Από το Σχήμα 4.12 προκύπτει πως το αποτέλεσμα της βλάστησης είναι αρκετά ικανοποιητικό καθώς όλες οι περιοχές που περιέχουν πράσινο εμφανίζονται με άσπρο χρώμα, ενώ οι αστικές περιοχές, το οδικό δίκτυο και γενικότερα οι περιοχές που είναι χωρίς βλάστηση εμφανίζονται με μαύρο χρώμα. Τέλος, εφαρμόστηκε και ο λόγος για εξαγωγή της Θάλασσας και της αστικής περιοχής και τα αποτελέσματά του ήταν εξίσου καλά.

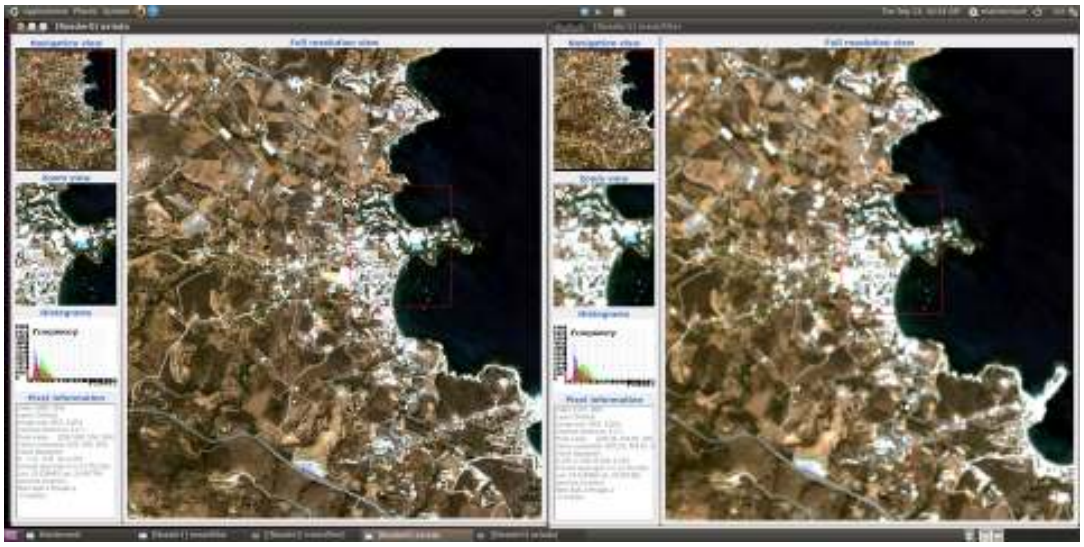
4.5 Εφαρμογή φίλτρων

Η ψηφιακή επεξεργασία φιλτραρίσματος μιας ψηφιακής τηλεπισκοπικής απεικόνισης αποσκοπεί στην ενίσχυση/ βελτίωση της, είτε με την απόκρυψη ή συμπίεση ορισμένων χωρικών συχνοτήτων και γραμμικών χαρακτηριστικών που εμποδίζουν ή δυσχεραίνουν την φωτοερμηνευτική ανάλυση άλλων ενδιαφερόντων στοιχείων, ή με την ενίσχυση των χωρικών συχνοτήτων και των γραμμικών χαρακτηριστικών που κυρίως μας ενδιαφέρουν. [Ρόκος, 2005]

Τα φίλτρα που υπάρχουν στον κώδικα του OTB είναι πάρα πολλά. Στο κεφάλαιο αυτό εφαρμόζονται κάποια ενδεικτικά φίλτρα προκειμένου να γίνει εμφανές ο τρόπος με τον οποίον αυτά δουλεύουν στο γραφικό περιβάλλον του Monteverdi. Στο μενού επιλογών του Monteverdi τα φίλτρα βρίσκονται στο υπο-μενού Feature Extraction.

4.5.1 Φίλτρα χαμηλών συχνοτήτων

Τα φίλτρα ομαλοποίησης χρησιμοποιούνται προκειμένου να απομακρυνθεί ο θόρυβος στην εικόνα καθώς αφαιρούν την πληροφορία υψηλών συχνοτήτων, αποτελούν δηλαδή φίλτρα χαμηλών συχνοτήτων. Χαρακτηριστικό παράδειγμα τέτοιου φίλτρου αποτελεί το φίλτρο του μέσου όρου. Έτσι λοιπόν εφαρμόστηκε στην εικόνα το φίλτρο mean όπως παρέχεται από το πρόγραμμα. Σαν στοιχεία του φίλτρου επιλέχτηκε ένας κύκλος με ακτίνα 2. Έτσι λοιπόν πραγματοποιήθηκε η πράξη της συνέλιξης μεταξύ της αρχικής εικόνας και της μάσκας του φίλτρου και προέκυψε το Σχήμα 4.13.



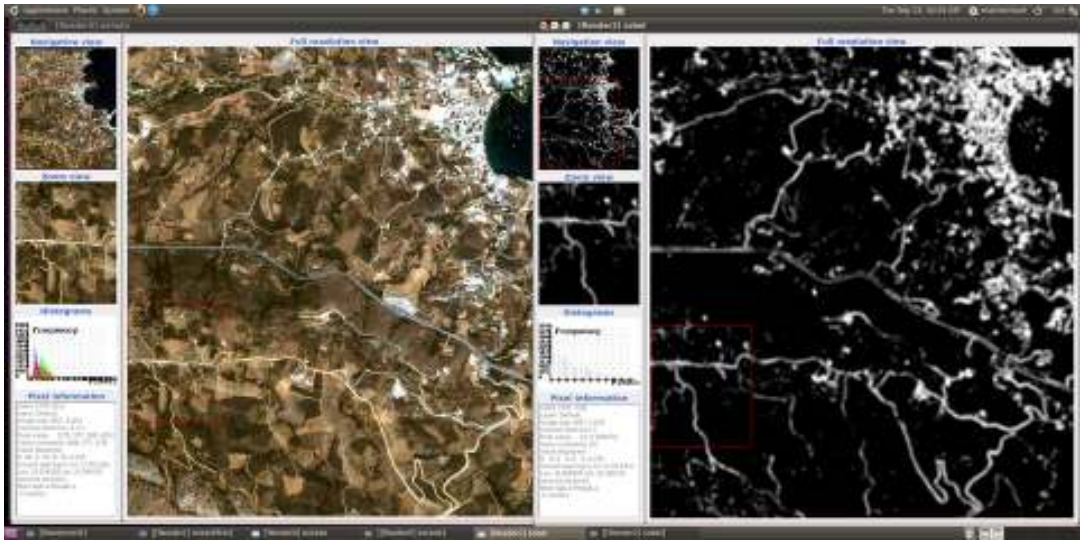
Σχήμα 4.13: Εφαρμογή φίλτρου μέσου όρου με πυρήνα 2x2 σε εικόνα QuickBird

Παρατηρούμε πως η τελική εικόνα μας (Σχήμα 4.13) είναι ομαλοποιημένη με αποτέλεσμα να φαίνεται θολή. Αν η ακτίνα της κυκλικής 'μάσκας' ήταν μεγαλύτερη τότε η τελική εικόνα θα ήταν περισσότερο θολή λόγου της εξομάλυνσης που υπέστη η εικόνα. Τέλος, τα όρια των διαφόρων καλλιεργειών ή δρόμων έγιναν περισσότερο δυσδιάκριτα.

4.5.2 Φίλτρα υψηλών συχνοτήτων

Τα φίλτρα ενίσχυσης των γραμμικών στοιχείων της δορυφορικής εικόνας ανήκουν στην κατηγορία φίλτρων υψηλών συχνοτήτων, δηλαδή τονίζουν τα χαρακτηριστικά υψηλών χωρικών συχνοτήτων, ενισχύοντας έτσι τις επιμήκεις λεπτομέρειες και τα γραμμικά χαρακτηριστικά της εικόνας. [Ρόκος, 2005]

Τα φίλτρα αυτά μπορεί να είναι είτε γραμμικά φίλτρα (δηλαδή η συνέλιξη να πραγματοποιείται με γραμμικό τρόπο), είτε μη γραμμικά. Στην περίπτωση αυτή χρησιμοποιείται το φίλτρο sobel. Παρόλο που το OTB παρέχει πολλά φίλτρα για ανίχνευση ακμών, το γραφικό του περιβάλλον υποστηρίζει μόνο το φίλτρο sobel. Τα αποτελέσματα της χρήσης του συγκεκριμένου φίλτρου είναι τα ακόλουθα:



Σχήμα 4.14: Εφαρμογή φίλτρου sobel με πυρήνα 3x3 σε εικόνα QuickBird

Μπορούμε να παρατηρήσουμε (Σχήμα 4.14) πως οι δρόμοι και γενικότερα όλες οι ακμές της εικόνας εμφανίζονται αρκετά ικανοποιητικά.

4.5.3 Φίλτρα μορφολογίας

Η Μορφολογία αποτελεί μια θεωρία και τεχνική της ανάλυσης και επεξεργασίας γεωμετρικών σχημάτων βασισμένη στις θεωρίες των συνόλων, των δικτύων, την τοπολογία και γενικότερα επιμέρους μαθηματικές διαδικασίες. Οι λειτουργίες της μορφολογίας είναι πολλές αλλά οι κυριότερες είναι οι διαστολή (dilation) και συστολή (erosion). Σε αυτές τις δύο λειτουργίες στηρίζονται και αυτές που αναφέρονται και υπάρχουν μέσα στο περιβάλλον του Monteverdi. Οι λειτουργίες αυτές είναι του μορφολογικού ανοίγματος (opening) και του μορφολογικού κλεισίματος (closing). Οι δύο αυτές λειτουργίες χρησιμοποιούνται για μετακίνηση θορύβου. Οι εφαρμογές τους βέβαια είναι πάρα πολλές. Το μορφολογικό άνοιγμα αποτελεί μια διαστολή μιας συστολής, ενώ το μορφολογικό κλείσιμο αποτελεί μια συστολή μιας διαστολής.

Στο σημείο αυτό, κρίνεται σκόπιμο να αναφερθούν οι βασικοί τύποι των τεσσάρων βασικών μορφολογικών λειτουργιών. Η μαθηματική μορφολογία, αποτελεί στην ουσία πράξεις συνόλων. Για το λόγο αυτό λοιπόν οι λειτουργίες της παρουσιάζονται μέσω αυτών. Αρχικά λοιπόν για τη διαστολή ισχύουν τα παρακάτω:

$$A \oplus B = \bigcup A_b$$

Ομοίως και με την συστολή

$$A \ominus B = \bigcap A_{-b}$$

Αντίστοιχα, το μορφολογικό άνοιγμα δίνεται από τον τύπο

$$A \circ B = (A \ominus B) \oplus B$$

Τέλος, το μορφολογικό κλείσιμο δίνεται από τον τύπο

$$A \bullet B = (A \oplus B) \ominus B$$

Με βάση τις λειτουργίες αυτές, γίνονται και όλες οι άλλες για οποιαδήποτε επεξεργασία εικόνας. Ενδεικτικά, εφαρμόζεται ένα μορφολογικό κλείσιμο στην εικόνα προκειμένου να σχολιαστούν τα αποτελέσματα που προκύπτουν. Η συγκεκριμένη λειτουργία αναμένεται να προσδιορίσει γεωμετρικά σχήματα που ξεχωρίζουν από τα γειτονικά τους. Αντίθετα, η λειτουργία του μορφολογικού κλεισίματος, χρησιμοποιείται προκειμένου να εμφανιστούν οι πιο ομαδοποιημένες περιοχές, απομακρύνοντας ό,τι δεν ανήκει στην ομαδοποιημένη περιοχή της εικόνας.

Όπως φαίνεται και στο Σχήμα 4.15, εφαρμόζοντας το φίλτρο του μορφολογικού opening, εντοπίστηκαν όλα τα μικρά κλειστά σχήματα, όπως είναι οι πισίνες που υπάρχουν στην περιοχή. Επίσης, ξεχώρισε και το οδικό δίκτυο της περιοχής. Το όλο αποτέλεσμα της μορφολογίας προβλήθηκε στο κανάλι 4 της εικόνας και για το λόγο αυτό ξεχώρισαν οι πισίνες της περιοχής.

Σχήμα 4.15: Εφαρμογή μορφολογικού ανοίγματος σε εικόνα Landsat της περιοχής Ηρακλείου

4.6 Εφαρμογή ταξινομήσεων

Σκοπός της διαδικασίας ταξινόμησης εικόνων είναι η αυτόματη κατηγοριοποίηση όλων των στοιχείων μιας εικόνας σε διαφορετικές τάξεις κάλυψης γης ή αντικείμενα. [Αργιαλας, 1998] Όπως αναφέρεται και παρακάτω υπάρχουν δύο είδη ταξινόμησης, η επιβλεπόμενη, στην οποία ο χρήστης πρέπει να δώσει δείγματα από την εικόνα προκειμένου να πραγματοποιηθεί και η μη επιβλεπόμενη, στην οποία ο χρήστης ορίζει απλά πόσες κλάσεις θέλει και στην συνέχεια η

ταξινόμηση γίνεται από μόνη της.

Το περιβάλλον του Monteverdi υποστηρίζει τις ταξινομήσεις που έχουν σαν δομικό στοιχείο τόσο το εικονοστοιχείο, όσο και το αντικείμενο. Για το λόγο αυτό, ο χρήστης μπορεί να πραγματοποιήσει μια ταξινόμηση φορτώνοντας την ίδια την εικόνα ή ακόμα και πραγματοποιώντας μια κατάτμηση πρώτα στην εικόνα και μετά ταξινομώντας τα επιμέρους αντικείμενα που δημιουργούνται σε αυτή. Η κατάτμηση που πραγματοποιεί το συγκεκριμένο περιβάλλον γίνεται με τον αλγόριθμο Mean Shift. Στον αλγόριθμο αυτό, ο χρήστης μπορεί να επιλέξει το μέγεθος των αντικειμένων που χρειάζεται να δημιουργήσει και γενικότερα να επιλέξει ο ίδιος τις παραμέτρους με τις οποίες θα πραγματοποιηθεί η κατάτμηση.

Έχοντας πραγματοποιήσει ή μη κατάτμηση της εικόνας, ο χρήστης μπορεί να προχωρήσει στην ταξινόμηση της περιοχής αρχικά με την μέθοδο της επιβλεπόμενης ταξινόμησης και στην συνέχεια με αυτή της μη επιβλεπόμενης. Η ταξινόμηση έγινε στο σύνολο της εικόνας, χωρίς να χρειαστεί να κοπεί κάποιο κομμάτι της.

4.6.1 Επιβλεπόμενη ταξινόμηση

Η επιβλεπόμενη ταξινόμηση γίνεται σε δύο στάδια. Το πρώτο στάδιο είναι αυτό της επίβλεψης (εκπαίδευσης), ενώ το δεύτερο είναι αυτό της ταξινόμησης. Στο πρώτο στάδιο ο Φωτοερμηνευτής αναγνωρίζει αντιπροσωπευτικές περιοχές εκπαίδευσης και αναπτύσσει μια αριθμητική περιγραφή των φασματικών ιδιοτήτων της κάθε θεματικής κατηγορίας. [Αργιαλάς, 1998]

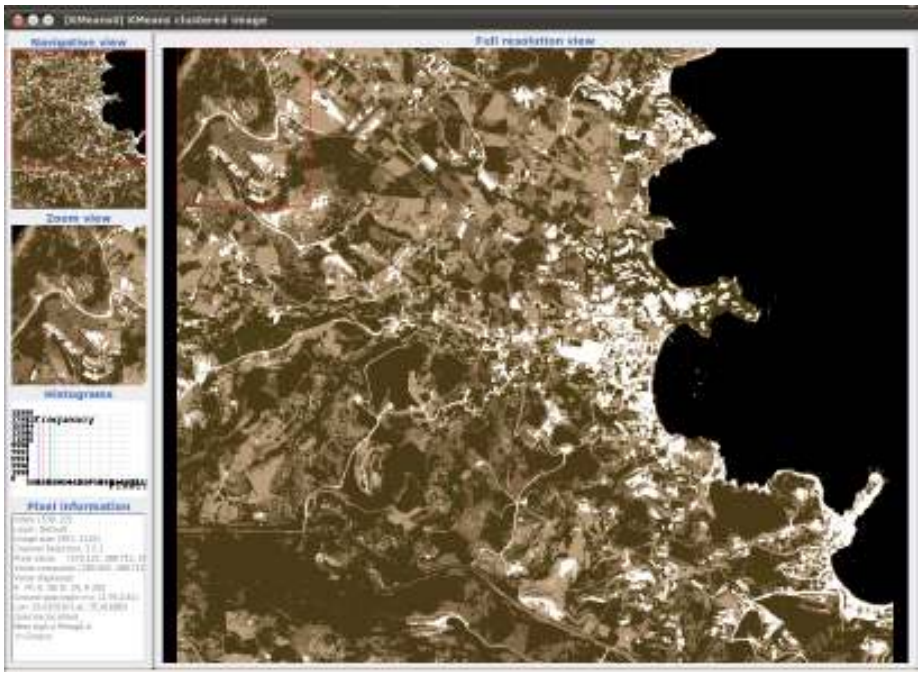
Η επιβλεπόμενη ταξινόμηση που περιέχει το περιβάλλον είναι η SVM Classification, που αναφέρεται σε αρχεία τύπου vector. Για το λόγο αυτό δεν πραγματοποιήθηκε κάποιο παράδειγμα στο συγκεκριμένο τμήμα.

4.6.2 Μη επιβλεπόμενη ταξινόμηση

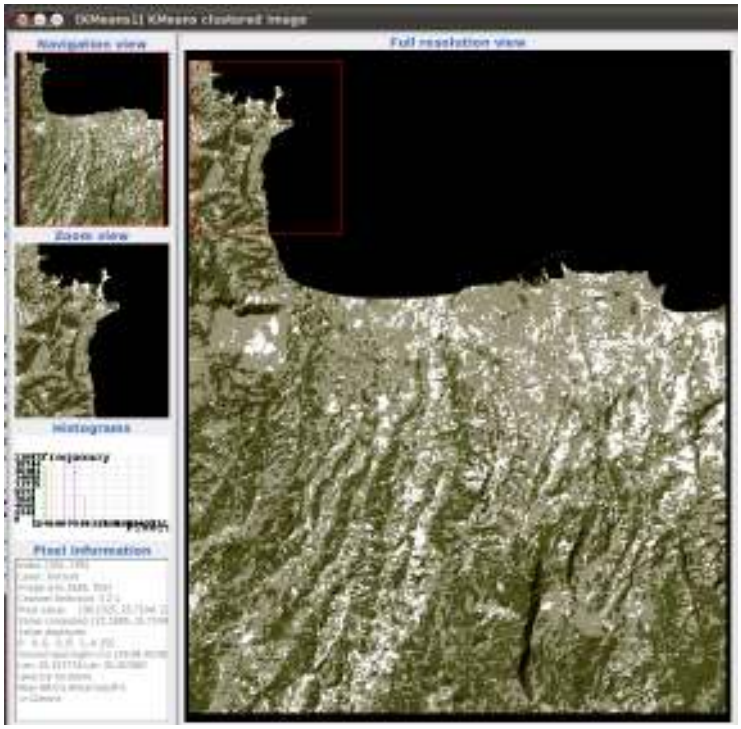
Σε αντίθεση με την επιβλεπόμενη ταξινόμηση, η μη επιβλεπόμενη δεν χρησιμοποιεί δεδομένα επίβλεψης. Η μη επιβλεπόμενη ταξινόμηση περιλαμβάνει αλγορίθμους, οι οποίοι εξετάζουν τα άγνωστα εικονοστοιχεία μιας εικόνας και τα ομαδοποιούν σ΄ έναν αριθμό κατηγοριών με βάση τις φυσικές ομαδοποιήσεις ή συσσωρεύσεις που ενυπάρχουν στις ψηφιακές τιμές της εικόνας. [Αργιαλάς, 1998]

Το περιβάλλον του Monteverdi επιτρέπει την ταξινόμηση τόσο με βάση το εικονοστοιχείο, όσο και με βάση το αντικείμενο. Ενδεικτικά, εφαρμόστηκε το πρώτο είδος ταξινόμησης. Όπως έχει αναφερθεί και παραπάνω ο αλγόριθμος που παρέχει το σύστημα για μη επιβλεπόμενη ταξινόμηση είναι ο KMeans. Ο αλγόριθμος αυτός χρησιμοποιήθηκε τόσο για την εικόνα της περιοχής Αχλάδας που προέρχεται από τον δορυφόρο Quickbird, όσο και για την περιοχή του Ηρακλείου που προέρχεται από τον δορυφόρο Landsat 7. Τα αποτελέσματα των ταξινομήσεων φαίνονται παρακάτω.

Για την ταξινόμηση στην συγκεκριμένη περίπτωση, επιλέχτηκαν 7 κατηγορίες τόσο στην μια εικόνα όσο και στην άλλη. Μπορούμε να παρατηρήσουμε πως η ταξινόμηση που προέκυψε, εμφανίζει καλά αποτελέσματα, με λίγο καλύτερα τα αποτελέσματα από την περιοχή της Αχλάδας. Αυτό οφείλεται κυρίως στο γεγονός ότι η εικόνα από τον δορυφόρο Quickbird έχει καλύτερη ακρίβεια από αυτή του δορυφόρου Landsat, με αποτέλεσμα ο αλγόριθμος να μπορεί να ομαδοποιήσει με καλύτερο τρόπο τις περιοχές με παρόμοια χαρακτηριστικά.



Σχήμα 4.16: Εφαρμογή μη επιβλεπόμενης ταξινόμηση με την μέθοδο KMeans της περιοχής Αχλάδας



Σχήμα 4.17: Εφαρμογή μη επιβλεπόμενης ταξινόμηση με την μέθοδο KMeans της περιοχής Ηρακλείου

Κεφάλαιο 5

Orfeo Toolbox (OTB)

5.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο αναλύθηκε το γραφικό περιβάλλον του OTB και πραγματοποιήθηκαν σε αυτό κάποιες βασικές εφαρμογές τηλεπισκόπησης. Στο κεφάλαιο αυτό, θα ασχοληθούμε με το πηγαίο κώδικα που περιέχει το σύστημα. Το OTB χρησιμοποιεί πολύ δυνατά συστήματα γενικευμένου προγραμματισμού, πολλές κλάσεις είναι ήδη διαθέσιμες, ενώ πολλά εργαλεία έχουν κατασκευαστεί προκειμένου να βοηθήσουν τον χρήστη σε όποια εφαρμογή του. Σε περίπτωση όμως που ο χρήστης θέλει να τα χρησιμοποιήσει όλα αυτά, πρέπει να εξοικειωθεί με τον κώδικα του συστήματος.

Όπως έχει ήδη αναφερθεί, ο κώδικας το OTB χρησιμοποιεί την CMake προκειμένου να μπορέσει να μεταγλωττιστεί με τον σωστό τρόπο. Για το λόγο αυτό ο χρήστης εκτός από τον κώδικα του αλγορίθμου που είναι γραμμένος σε C++, πρέπει να προσθέσει στον ίδιο φάκελο και τον κώδικα της CMake που έχει δημιουργήσει για να τρέξει τον αλγόριθμο. Η CMake δημιουργεί δικά της makefiles τα οποία μπορούν να χρησιμοποιηθούν από τον compiler που διαλέγει ο χρήστης ανάλογα με το περιβάλλον που βρίσκεται. Όλη η διαδικασία γίνεται ανάλογα με την διανομή που έχει ο κάθε χρήστης, από το αντίστοιχο τερματικό. Τα αρχεία της CMake είναι γραμμένα με παρόμοιο τρόπο και ακολουθούν και αυτά ένα πρότυπο που θα αναλυθεί παρακάτω.

Στο σημείο αυτό αναλύεται το πρώτο παράδειγμα στο OTB. Στο πρόγραμμα αυτό ο αλγόριθμος το μόνο που πραγματοποιήθηκε είναι να αντιγράφει την εικόνα εισαγωγής του χρήστη σε όποιο format θέλει ο χρήστης. Όλος ο κώδικας των παραδειγμάτων παρουσιάζεται στο παράρτημα. Το παράδειγμα αυτό αναλύεται σε μεγάλο βαθμό έτσι ώστε ο χρήστης να καταλάβει τον τρόπο που δομούνται τα προγράμματα του OTB.

5.1.1 Τρόπος δόμησης της CMake

Αρχικά αναλύεται ο τρόπος με τον οποίο είναι γραμμένο το αρχείο της CMake. Το συγκεκριμένο αρχείο δεν είναι τίποτα άλλο από ένα αρχείο txt, το οποίο μάλιστα και ονομάζεται με συγκεκριμένο τρόπο για να αναγνωρίζεται από το σύστημα (CMakeLists.txt).

- Στην πρώτη γραμμή του αρχείου, ορίζεται το όνομα του προγράμματος που υλοποιείται και έτσι όπως εμφανίζεται στο Visual Studio για τα Windows, ενώ για τα περιβάλλοντα Linux, Unix η συγκεκριμένη γραμμή δεν έχει καμία απολύτως επίπτωση (παράρτημα, 5.1.1 αρχείο CMake, γραμμή 1).
- Η επόμενη γραμμή αναφέρεται στην ελάχιστη προαπαιτούμενη έκδοση της CMake που χρειάζεται το σύστημα για να μεταγλωττιστεί (παράρτημα, παρουσίαση εικόνας, 5.1.1 αρχείο CMake, γραμμή 3).

- Το επόμενο μπλοκ εντολών ελέγχεται αν το σύστημα έχει βρει τον φάκελο του OTB και σε περίπτωση που δεν μπορέσει να το βρει, αναφέρει στον χρήστη το πρόβλημα, ενώ σε περίπτωση που όλα κυλήσουν σωστά, φορτώνεται το αρχείο UseOTB.cmake το οποίο περιέχει όλες τις πληροφορίες διαμόρφωσης του OTB (παράρτημα, παρουσίαση εικόνας, 5.1.1 αρχείο CMake, γραμμές 5-11).

- Η επόμενη εντολή περιέχει ως πρώτο όρισμα το όνομα του εκτελέσιμου το οποίο παράγεται ως αποτέλεσμα από την CMake, ενώ το δεύτερο όρισμά του είναι το όνομα της πηγής του κώδικα από όπου παράγεται το αποτέλεσμα. Στην ουσία μόνο η γραμμή αυτή και η επόμενη μεταβάλλεται για όλα τα αρχεία που δημιουργεί ο χρήστης για τα επιμέρους προγράμματά του (παράρτημα, παρουσίαση εικόνας, 5.1.1 αρχείο CMake, γραμμή 13).

- Τέλος, η επόμενη εντολή αναφέρεται στις βιβλιοθήκες που χρειάζεται να συνδεθούν προκειμένου να μεταγλωττιστεί το πρόγραμμα. Για λόγους πρακτικούς, το πρώτο όρισμα της συγκεκριμένης εντολής είναι πάλι το όνομα του αρχείου που περιέχει τον κώδικα και είναι το μόνο που αλλάζει ανάλογα το πρόγραμμα (παράρτημα, παρουσίαση εικόνας, 5.1.1 αρχείο CMake, γραμμή14).

Ο χρήστης αρκεί να ακολουθήσει το συγκεκριμένο πρότυπο προκειμένου να φτιάξει τα αρχεία CMake, και κατ΄ επέκταση να μεταγλωττίσει και τα δικά του προγράμματα. Έτσι και αλλιώς η προτυποποίηση η οποία ακολουθείται είναι η ίδια σε κάθε περίπτωση.

5.1.2 Τρόπος δόμησης του αρχείου του κώδικα

Όλος ο κώδικας του OTB έχει δομηθεί με δείκτες και μάλιστα Smartpointer. Αυτό έτσι και αλλιώς ήταν αναμενόμενο καθώς τα δεδομένα εισόδου είναι εικόνες (δηλαδή πίνακες), πολλών διαστάσεων οπότε ο μόνος τρόπος για να μπορέσουν να διαχειριστούν με τον σωστό τρόπο είναι οι δείκτες. Επίσης όλος ο κώδικας είναι δομημένος με templates. Τα πρότυπα ή αλλιώς templates παρέχουν ένα απλό τρόπο για να αναπαρασταθεί ένα ευρύ φάσμα γενικών εννοιών, και απλούς τρόπους για να συνδυαστούν οι έννοιες αυτές. [Stroustrup, 2003] Τα πρότυπα παρέχουν έμμεση υποστήριξη για το γενικευμένο προγραμματισμό, δηλαδή τον προγραμματισμό όπου χρησιμοποιούνται τύποι ως παράμετροι. Έτσι και αλλιώς το OTB στηρίζεται στον γενικευμένο προγραμματισμό οπότε και θα χρησιμοποιούσε σίγουρα τα πρότυπα.

Στο σημείο αυτό αναλύεται ο κώδικας του συγκεκριμένου προγράμματος bit by bit.

- Αρχικά το πρώτο πράγμα που περιείχε ο κώδικας ήταν ο ορισμός των header αρχείων που χρειαζόταν. Τα αρχεία αυτά βρίσκονταν μέσα στο σύστημα του OTB. Τα αρχεία αυτά βρίσκονταν μέσα στον φάκελο του OTB και για το λόγο αυτό ορίζονταν μέσα σε ''..'' και όχι σε <..>όπως συνηθίζεται (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμές 29-32).

- Στην συνέχεια ορίστηκε η βασική συνάρτηση του κώδικα, δηλαδή η main. Στην περίπτωση αυτή και στις περισσότερες περιπτώσεις, η συνάρτηση πήρε ως ορίσματα τα στοιχεία της εικόνας. Το πρώτο όρισμα ήταν ένας μετρητής του πλήθους των παραμέτρων της εικόνας που εισήχθηκε (argc) ενώ το δεύτερο όρισμα ήταν ένας δείκτης που περιείχε όλα τα δεδομένα που εισήχθηκαν στο πρόγραμμα (*argv[]). Έτσι όλα τα στοιχεία που χρειάστηκε ο χρήστης εισήχθησαν στον κώδικα (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμή 34).

- Ακολούθησε ο ορισμός της εικόνας. Από το πρότυπο λοιπόν otb::Image ορίστηκε μια μεταβλητή ImageType η οποία είχε δύο διαστάσεις και τα στοιχεία της ήταν μη αρνητικοί χαρακτήρες unsigned char (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμή 36).

- Επόμενο βήμα αποτέλεσε ο ορισμός του δείκτη που διάβαζε την εικόνα. Με παρόμοιο τρόπο λοιπόν, ορίστηκε ένας καινούργιος δείκτης τύπου ReaderType από το πρότυπο otb::ImageFileReader. Ο δείκτης αυτός δημιουργήθηκε δυναμικά με χρήση του New() (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμές 38-39).

- Με ακριβώς τον ίδιο τρόπο ορίστηκε και ο δείκτης writer ο οποίος ανήκει στο πρότυπο otb::StreamingImageFileWriter. Ο δείκτης αυτός περιλάμβανε την εικόνα που θα δόθηκε από το πρόγραμμα ως αποτέλεσμα (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμές 41-42).

- Αφού λοιπόν ορίστηκαν όλες οι μεταβλητές που χρειάστηκε το πρόγραμμα όπως παρουσιάστηκαν παραπάνω, σειρά είχε να περαστούν οι τιμές των δεδομένων στις μεταβλητές. Έτσι και χωρίς να ελεγχθεί σε αυτό το παράδειγμα η εγκυρότητα των δεδομένων που εισήγαγε ο χρήστης, στον δείκτη αναγνώστη (reader) πέρασε η εικόνα που έδωσσε ο χρήστης σαν είσοδο, ενώ στον δείκτη γραφέα (writer) πέρασε το όνομα της εικόνας που έδωσε ο χρήστης σαν αποτέλεσμα (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμές 44-45).

- Η εντολή αυτή πραγματοποίησε τον λόγο ύπαρξης του προγράμματος. Στην ουσία μέσω μιας διοχέτευση ροής (pipeline) το αποτέλεσμα του αναγνώστη περνάει σαν είσοδο στον γραφέα (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμή 47).

- Τέλος, η όλη διαδικασία ολοκληρώθηκε, και πραγματοποιήθηκε η εκτέλεση της ροής μεσώ της συγκεκριμένης εντολής. Καλώντας την εντολή Update() στο τελευταίο στοιχείο της ροής, ο χρήστης έγινε σίγουρος πως ενημερώθηκαν όλα τα προηγούμενα μέλη της ροής. Με την εντολή αυτή, περάσαν και όλα τα metadata της εικόνας (παράρτημα, παρουσίαση εικόνας, 5.1.2 αρχείο κώδικα γραμμή 48).

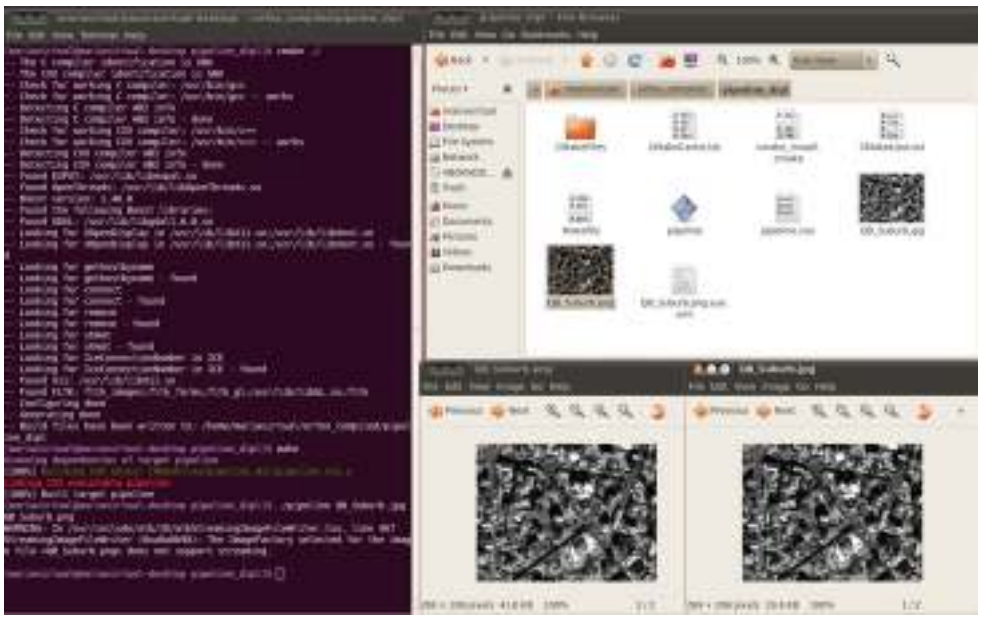
5.1.3 Οργάνωση του συστήματος

Αφού αναφέρθηκε με κάθε λεπτομέρεια ο τρόπος που είναι δομημένος ο κώδικας του παραπάνω προγράμματος αλλά και το αρχείο της CMake, ο χρήστης δεν είχε τίποτα άλλο από το να κατασκευάσει έναν φάκελο στον οποίο και τοποθέτησε τόσο τα δύο αρχεία, όσο και την εικόνα που εισήγαγε. Είναι πολύ σημαντικό όλα τα στοιχεία να βρίσκονται στον ίδιο φάκελο, έτσι ώστε το πρόγραμμα να αναγνωρίζει όλα τα στοιχεία που χρειάζεται. Στην συνέχεια και ανάλογα με την διανομή που έχει ο κάθε χρήστης τρέχει τις εντολές της cmake στο τερματικό του συστήματός του, αφού βρίσκεται μέσα στον φάκελο που έχει δημιουργήσει. Η όλη διαδικασία ολοκληρώνεται σε 3 βήματα όπως φαίνεται και παρακάτω.

- Έτσι όπως φαίνεται και στην παρακάτω εικόνα (Σχήμα 5.1), αρχικά ο χρήστης έτρεξε την εντολή cmake ./. Στην ουσία η εντολή αυτή έψαξε να βρει τους διαθέσιμους μεταγλωττιστές του συστήματος για το αρχείο κώδικα που υπήρχε στον φάκελο. Ο μεταγλωττιστής που χρησιμοποιήθηκε για την γλώσσα C++ είναι ο GCC, ο οποίος πρέπει να υπάρχει μέσα στο σύστημα του χρήστη. Έτσι και αλλιώς το OTB έχει δοκιμαστεί με το συγκεκριμένο μεταγλωττιστή για περιβάλλον Unix/Linux. Στο στάδιο αυτό πραγματοποιήθηκε το configure του κώδικα του προγράμματος. Ο χρήστης παρατήρησε την δημιουργία διαφόρων αρχείων της CMake μέσα στο φάκελο που δημιούργησε.

- Στην συνέχεια ο χρήστης πραγματοποίησε την εντολή make. Με την εντολή αυτή ελέγχθηκε όλος ο κώδικας για το αν περιείχε κάποιο σφάλμα το οποίο και έγινε γνωστό στον χρήστη. Σε περίπτωση που ο κώδικας δεν περιείχε κάποιο συντακτικό λάθος, σύμφωνα με τον μεταγλωττιστή του συστήματος, τότε δημιουργείται μέσα στον φάκελο ένα εκτελέσιμο αρχείο με το όνομα που του είχε δοθεί στο αρχείο της CMake.

- Στο τελικό στάδιο ο χρήστης έτρεξε το αρχείο που του δημιούργησε η CMake δίνοντας μαζί και τα ορίσματα που χρειάστηκε για να τρέχει το πρόγραμμα.



Σχήμα 5.1: Εφαρμογή αντιγραφής εικόνας με χρήση του OTB για την περιοχή της Αχλάδας

Όπως παρατηρείται και παραπάνω, το πρόγραμμα που δημιουργήθηκε όχι απλώς αντιγράφει την εικόνα, αλλά και την μετατρέπει σε όποιο format του ορίζει ο χρήστης. Μπορεί το συγκεκριμένο πρόγραμμα να μην κάνει τρομερά πράγματα, αλλά εισάγει τον χρήστη στον τρόπο με τον οποίο λειτουργεί το OTB προκειμένου να κατασκευάσει στην συνέχεια τα δικά του προγράμματα.

Στο σημείο αυτό, πρέπει να αναφέρουμε πως η εικόνα που εισήχθηκε στο πρόγραμμα ήταν μια εικόνα δύο διαστάσεων (όπως και ορίσαμε στο πρόγραμμα). Σε περίπτωση που την ίδια δουλειά την θέλαμε για εικόνες με πολλές διαστάσεις και κατέπέκταση και πολλά κανάλια, η διαδικασία θα ήταν η ίδια με μόνη διαφορά τον τρόπο με τον οποίο θα ορίζαμε τον δείκτη reader αλλά και writer, καθώς θα έπρεπε να ορίζονται ως VectorImageType και όχι απλά ως ImageType. Η συγκεκριμένη διαφορά θα αναλυθεί και παρακάτω καθώς όλα τα προγράμματα από εδώ και πέρα θα γίνονται με εικόνες format tiff.

5.2 Επεξεργασίες σε πολυκάναλες εικόνες

Σε πολλές περιπτώσεις, ο χρήστης χρειάζεται να δουλέψει σε κάποια από τα κανάλια της εικόνας. Μια τέτοια περίπτωση είναι αυτή της εφαρμογής φίλτρων, όπου πολλά από τα φίλτρα του OTB εφαρμόζονται μόνο σε ένα κανάλι, είτε γιατί είναι έτσι κατασκευασμένος ο αλγόριθμός τους, είτε γιατί τα πολλά κανάλια δεν δίνουν καμιά επιπλέον πληροφορία για τον αλγόριθμο. Για το λόγο αυτό, παρακάτω αναλύθηκαν οι μέθοδοι εξαγωγής μόνο ενός καναλιού από μια πολυκάναλη εικόνα και η εξαγωγή της rgb εικόνας από αυτή.

5.2.1 Εξαγωγή ενός καναλιού από πολυφασματικές εικόνες

Προκειμένου να μπορέσει ο χρήστης να διαχειριστεί και να επεξεργαστεί την εικόνα του, πρέπει να ξέρει να την διαχωρίζει από τα διάφορα επιμέρους στοιχεία της. Έτσι στη συγκεκριμένη

εφαρμογή, ο χρήστης ανέλυσε την πολυκάναλη εικόνα στα επιμέρους κανάλια της και εξήγαγε μια δεύτερη εικόνα που περιείχε μόνο το ένα από αυτά τα κανάλια.

- Πρώτη ενέργεια όπως έγινε σε όλα τα προγράμματα μέχρι τώρα ήταν να ορίσει ο χρήστης τα αρχεία επικεφαλίδες, που χρειάστηκε προκειμένου να πραγματοποιήσει τις λειτουργίες που ήθελε.Τα αρχεία που χρειάστηκε να αναλυθούν στο συγκεκριμένο παράδειγμα είναι δύο. Το αρχείο otbStreamingImageFileWriter.h χρησιμοποιήθηκε, προκειμένου να περάσουν όλα τα ορίσματα που έπρεπε στον γραφέα, που εμφάνισε το τελικό αποτέλεσμα. Το συγκεκριμένο αρχείο περικλείεται σε όλα τα προγράμματα φίλτρων. Ακολούθησε το αρχείο otbMultiToMonoChannelExtractROI.h το οποίο χρησιμοποιήθηκε για να απομονώσει ένα κανάλι από την πολυφασματική εικόνα που δόθηκε ως είσοδος στο πρόγραμμα. (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 29-30).

- Στην συνέχεια, ελέγχθηκε κατά πόσο τα δεδομένα που εισήγαγε ο χρήστης είχαν τον σωστό αριθμό. Στην συγκεκριμένη περίπτωση, ο χρήστης εισήγαγε δύο ονόματα. Το πρώτο όνομα αντιστοιχεί στην εικόνα που θα γίνει η επεξεργασία, ενώ το επόμενο όνομα είναι το όνομα του αρχείου που θα δημιουργηθεί. Σε περίπτωση λάθους το πρόγραμμα ενημερώνει τον χρήστη ότι δεν εισήγαγε σωστά τις τιμές(παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 35-41).

- Στο επόμενο σετ εντολών ορίστηκε αναγνώστης, χρησιμοποιώντας ως τύπο εισόδου στο αντίστοιχο πρότυπο το VectorImageType καθώς η εικόνα ήταν αντίστοιχου τύπου. Αφού ορίστηκε ο δείκτης πήρε και ως είσοδο το όνομα της εικόνας που επεξεργαστηκε (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 45-51).

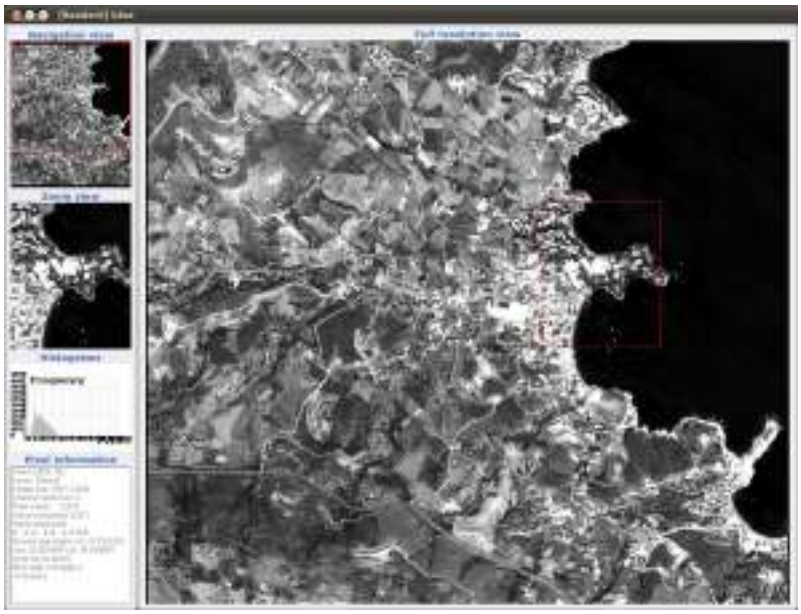
- Στην συνέχεια ορίστηκε ο δείκτης extractChannel, ο οποίος από την αντίστοιχη διαδικασία απομόνωσε το ένα κανάλι από την εικόνα. Ο δείκτης αυτός προήλθε από την αντίστοιχη κλάση MultiToMonoChannelExtractROI (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 53-55).

- Το αρχείο επικεφαλίδα που απομόνωσε ένα από τα κανάλια της εικόνας, μπορεί παράλληλα να απομονώσει για αυτή την δουλειά μια υποπεριοχή της εικόνας. Στην συγκεκριμένη περίπτωση όμως ο χρήστης όρισε ως είσοδο την GetLargestPossibleRegion η οποία απομόνωσε όλη την εικόνα. Σε διαφορετική περίπτωση, ο χρήστης μπορούσε να δώσει τις συντεταγμένες της υποπεριοχής που ήθελε να κάνει την επεξεργασία (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 58-60).

- Επόμενο βήμα αποτέλεσε ο ορισμός του καναλιού που απομονώθηκε, καθώς επίσης και η είσοδο της εικόνας στον αντίστοιχο δείκτη απομόνωσης. Στην συγκεκριμένη περίπτωση, επιλέχθηκε να απομονωθεί το τρίτο κανάλι που περιείχε το μπλε χρώμα (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 61-62).

- Προκειμένου να τελειώσει αυτή η διαδικασία, χρησιμοποιήθηκε το αρχείο StreamingImageFileWriter.h προκειμένου να οριστεί ο δείκτης που λάμβανε το αποτέλεσμα. Ο δείκτης αυτός πήρε ως είσοδο το αποτέλεσμα του δείκτη extractChannel, ενώ το αποτέλεσμα εφαρμόστηκε στο δεύτερο όρισμα του προγράμματος. Στην συνέχεια, ο δείκτης έγινε Update προκειμένου όλη η διοχέτευση της ροής να εκτελεστεί. Μετά από όλη αυτή την διαδικασία προέκυψε μια μονοκάναλη εικόνα την οποία μπόρεσε να επεξεργαστεί ο χρήστης (παράρτημα, 5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα, 65-72).

Τέλος το αποτέλεσμα της εφαρμογής του συγκεκριμένου αλγορίθμου είναι το ακόλουθο Σχήμα 5.2. Παρατηρείται πως όλη η περιοχή παρουσιάζεται με ασπρόμαυρες αποχρώσεις, όπου όποιες περιοχές περιέχουν περισσότερο μπλε χρώμα παρουσιάζονται μαύρες, ενώ αυτές που δεν διαθέτουν καθόλου μπλε απόχρωση παρουσιάζονται άσπρες.



Σχήμα 5.2: Εφαρμογή εξαγωγής του κόκκινου καναλιού (κανάλι 3) από την περιοχή της Αχλάδας

5.2.2 Εξαγωγή rgb εικόνας από την αντίστοιχη πολυκάναλη

Η συγκεκριμένη εφαρμογή ήταν παρόμοια με την παραπάνω με την μόνη διαφορά πως αντί για ένα κανάλι, δημιουργήθηκε μια εικόνα που περιείχε τρία κανάλια και τα κανάλια αυτά πρόβαλαν το έγχρωμο σύνθετο 321(RGB). Το μπλε, το πράσινο και το κόκκινο είναι τα κύρια προσθετικά χρώματα. Η σύνθεση ή προσθήκη αυτών των τριών χρωματικών ερεθισμάτων δημιουργεί την αντίληψη του ενός συγκεκριμένου χρώματος για το κάθε αντικείμενο. [Αργιαλάς, 1999]

Έτσι και αλλιώς μια rgb εικόνα είναι πολύ χρήσιμη καθώς από αυτή μπορεί να προκύψουν πλήθος εφαρμογών. Μια από τις εφαρμογές αυτές είναι ο υπολογισμός του χώρου hsi όπου τα κανάλια κόκκινο, πράσινο και μπλε μετατρέπονται σε κανάλι κορεσμού (hue), απόχρωσης (saturation) και έντασης (intensity). Ο χρήστης μπορεί να χρησιμοποιήσει την μετασχηματισμένη εικόνα για οποιαδήποτε εφαρμογή, όπως για παράδειγμα, την εφαρμογή στο κανάλι της έντασης οποιουδήποτε φίλτρου.

- Τα αρχεία επικεφαλίδες που χρησιμοποιήθηκαν σε αυτή την περίπτωση ήταν ακριβώς τα ίδια με αυτά που χρησιμοποιήθηκαν στο παραπάνω παράδειγμα. Έτσι και αλλιώς η μέθοδος που ακολουθήθηκε, ήταν παρόμοια με την παραπάνω (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 24-28).

- Ακολούθησε ο έλεγχος των δεδομένων στο πρόγραμμα (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 33-42).

- Στο συγκεκριμένο παράδειγμα ο χρήστης όρισε τα δεδομένα εισόδου διαβάζοντάς τα από αρχείο. Έτσι ορίστηκαν δύο δείκτες που ήταν σταθεροί και πήραν σαν ορίσματα τα ονόματα των δύο αρχείων που εισήγαγε ο χρήστης. Παράλληλα ορίστηκε και ο αναγνώστης, έτσι όπως ορίστηκε και στα παραπάνω παραδείγματα (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 44-51).

- Στην συνέχεια, ορίστηκαν και οι δύο δείκτες του extractROIFilter και ο γραφέας με παρόμοιο τρόπο. Όλοι οι τύποι των εικόνων ορίστηκαν ως VectorImageType καθώς

περιείχαν παραπάνω από ένα κανάλι, ενώ ο δείκτης που πραγματοποιήθηκε την απομόνωση των καναλιών πήρε ως τύπο στο πρότυπό του, το PixelType (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 54-59).

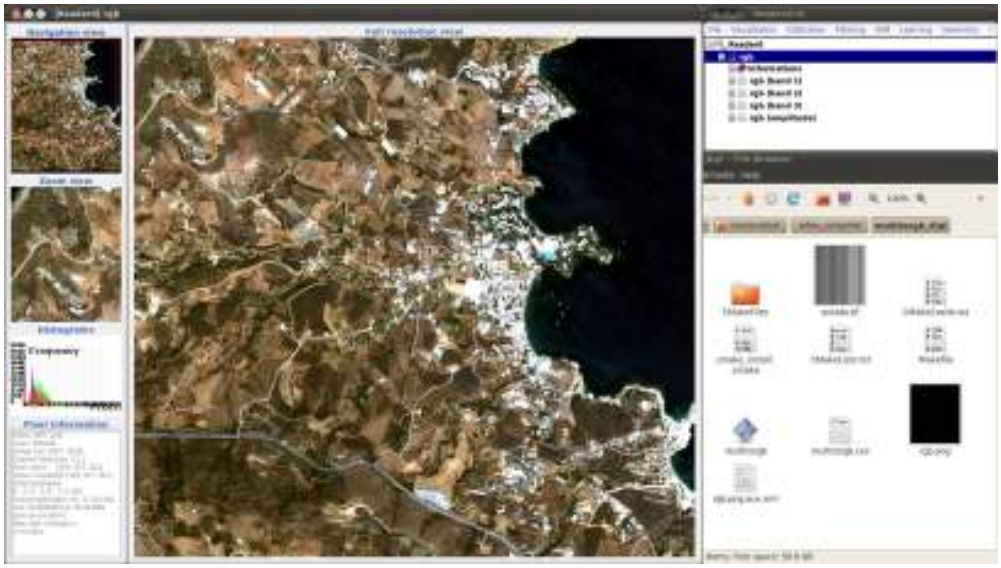
- Στο επόμενο βήμα οι δείκτες πήραν τις τιμές τους από τους δείκτες που διάβασαν τα αρχεία των δεδομένων (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 61-63).

- Ακολούθησε ο ορισμός του δείκτη που απομόνωσε τα κανάλια. Με όμοιο τρόπο, όπως προηγουμένως, ο δείκτης έλαβε ως τιμή του όλη την περιοχή της εικόνας (GetLargest PossibleRegion). Η διαφορά που παρατηρήθηκε ήταν πως αντί για ένα κανάλι ο δείκτης έλαβε μια αλληλουχία καναλιών. Μάλιστα πήρε από το πρώτο μέχρι το τρίτο κανάλι διαδοχικά (SetFirstChannel(1), SetLastChannel(3)), σε περίπτωση που τα κανάλια που ήθελε ο χρήστης να περάσει δεν ήταν διαδοχικά απλά μπορούσε να χρησιμοποιήσει την μέθοδο SetChannel όπου ορίζει ξεχωριστά το κάθε κανάλι εισαγωγής (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 66-71).

- Ο αλγόριθμος συνέχισε με την δημιουργία της ροής, όπου όλες οι παράμετροι πέρασαν στα αντίστοιχα πεδία (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 74-75).

- Τέλος η ροή εκτελέστηκε όπως όλες τις φορές με την μέθοδο Update (παράρτημα, 5.3 Δημιουργία RGB εικόνας από πολυκάναλη, 77).

Το αποτέλεσμα του συγκεκριμένου αλγορίθμου είναι μια καινούργια εικόνα που περιέχει μόνο τρία από τα τέσσερα κανάλια της αρχικής. (Σχήμα 5.3)



Σχήμα 5.3: Εφαρμογή δημιουργίας τρικάναλης εικόνας (RGB στην περιοχή της Αχλάδας

Στο συγκεκριμένο πρόγραμμα έγιναν μια σειρά από αλλαγές προκειμένου αυτό να καταφέρει να δώσει τα σωστά αποτελέσματα. Αρχικά, επιλέχτηκε να πάρει σαν όρισμα όλη την εικόνα και όχι μια υποπεριοχή της. Στην συνέχεια, το μεγαλύτερο πρόβλημα που αντιμετωπίστηκε στον αρχικό κώδικα ήταν ότι με τον τρόπο που οριζόταν η εικόνα, δούλευε μόνο για εικόνες 8bit. Έτσι άλλαξε όλος ο τρόπος ορισμού της αρχικής εικόνας, και προσδιορίστηκε ως VectorImageType. Η διαφορά των δύο προγραμμάτων στο σημείο αυτό ήταν το γεγονός ότι στην πρώτη περίπτωση η εικόνα οριζόταν ως unsigned char (δηλαδή χαρακτήρας

μη μηδενικός), ενώ στην δεύτερη περίπτωση ως unsigned int (δηλαδή ακέραιος αριθμός μη αρνητικός). Επίσης, στην πρώτη περίπτωση ορίστηκε ο τύπος του αρχείου, ενώ στο τελικό πρόγραμμα ορίστηκε ο τύπος της εικόνας. Με λίγα λόγια, παρατηρήθηκε πως τα προγράμματα που ορίστηκαν μέσω του προτύπου (template) αρχείου δεν διαβάστηκαν σωστά 8bit εικόνες και για το λόγο αυτό καλό είναι να αποφεύγονται για το σκοπό αυτό.

5.2.3 Υπολογισμός των κύριων συνιστωσών

Η σημαντική συσχέτιση μεταξύ των καναλιών είναι ένα πρόβλημα, το οποίο συχνά συναντάται στην ανάλυση των πολυφασματικών εικόνων. Ο μετασχηματισμός σε κύριες ή κανονικές συνιστώσες είναι δύο τεχνικές, οι οποίες έχουν αναπτυχθεί ώστε να εξαλείψουν ή να ελαττώσουν τέτοιους είδους πλεονασμούς στα πολυφασματικά δεδομένα. Ο σκοπός των συγκεκριμένων αυτών τεχνικών είναι να συμπιέσει όλη την πληροφορία, η οποία εμπεριέχεται σε μια αρχική πολυφασματική εικόνα n καναλιών, σε λιγότερα από n νέα κανάλια ή διάφορες συνιστώσες. Τέλος, η πρώτη κύρια συνιστώσα PC1 περιλαμβάνει το μεγαλύτερο ποσοστό της ολικής διασποράς της εικόνας, και οι επόμενες συνιστώσες PC2, PC3,...,PCn η κάθε μια περιλαμβάνει ένα μειωμένο ποσοστό της διασποράς της εικόνας, σε σχέση με την προηγούμενη. [Αργιαλάς, 1998]

Με άλλα λόγια, σε περίπτωση που ο χρήστης ήθελε να εφαρμόσει ένα φίλτρο ή μια ταξινόμηση, αντί να το πραγματοποιήσει σε ένα από τα κανάλια της εικόνας, μπορούσε να το εφαρμόσει σε κάποια από τις συνιστώσες που είχαν προκύψει από τον αλγόριθμο καθώς αυτές περιλαμβάνουν το μεγαλύτερο μέρος της πληροφορίας για την εικόνα. Ο αλγόριθμος δεχόταν ως είσοδος την τρικάναλη εικόνα σε png format και τον αριθμό των συνιστωσών που ήθελε ο χρήστης και έδινε ως αποτέλεσμα την εικόνα των κύριων συνιστωσών σε tif format και το αποτέλεσμα της κάθε συνιστώσας σε png format. Αναλυτικότερα τώρα, ο αλγόριθμος, περιείχε τα εξής βήματα:

- Αρχικά, ορίστηκαν τα αρχεία επικεφαλίδες που χρησιμοποιήθηκαν για τον συγκεκριμένο αλγόριθμο. Εκτός από τα βασικά αρχεία που περιέχονταν σε κάθε αλγόριθμο, ο χρήστης όρισε και το αρχείο otbInnerProductPCAImageFilter το οποίο χρησιμοποιήθηκε για την εφαρμογή των κύριων συνιστωσών (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 18-24).

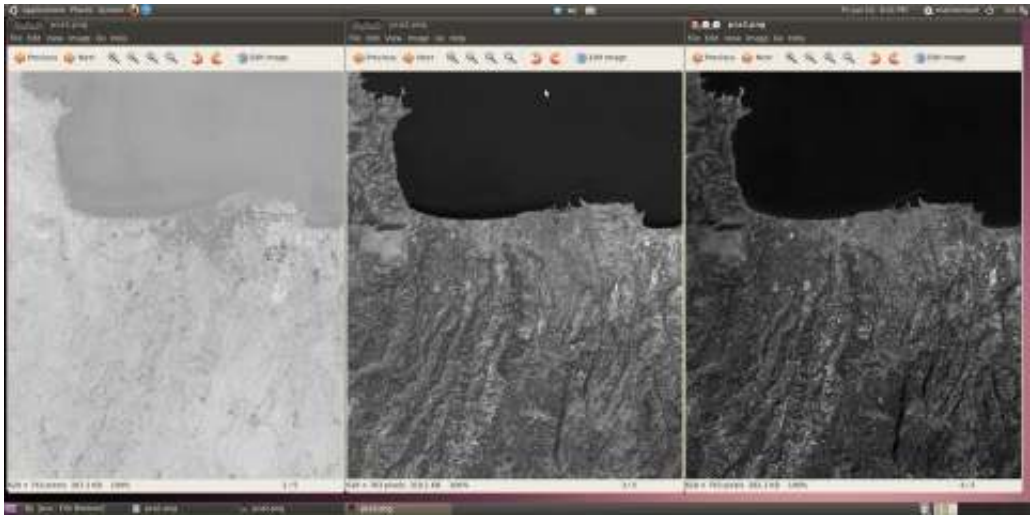
- Ακολούθησε η δήλωση όλων των παραμέτρων του προγράμματος στις αντίστοιχες μεταβλητές -δείκτες. Οι δείκτες που περιελάμβαναν τις δύο εικόνες, η μια που εισήχθηκε στο πρόγραμμα και η άλλη που δημιουργήθηκε από αυτό, ήταν τύπου const char δηλαδή σταθερού χαρακτήρα, κάτι που υποδηλώνει για άλλη μια φορά πως ο αλγόριθμος λειτουργούσε σωστά με εικόνες 8bit (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 34-38).

- Στην συνέχεια ορίστηκαν οι τύποι των εικόνων. Το γεγονός ότι επρόκειτο για πολυκάναλες εικόνες έκανε τον προγραμματιστή να χρησιμοποιήσει την κλάση VectorImage. Αφού ορίστηκαν όλες οι μεταβλητές, ο αναγνώστης έλαβε ως είσοδο την εικόνα (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 40-45).

- Έπειτα, ορίστηκε το φίλτρο των συνιστωσών όπου έλαβε ως ορίσματα στο πρότυπό του τον τύπο της εικόνας που πήρε ως είσοδο και τον τύπο της εικόνας που δημιουργήθηκε έτσι όπως είχαν οριστεί παραπάνω. Ο δείκτης του φίλτρου έλαβε επίσης ως όρισμα και τον αριθμό των συνιστωσών που όρισε ο χρήστης και έπρεπε να είναι λιγότερες ή ίσες με αυτές του αριθμού των καναλιών της εικόνας που εισήχθηκε (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 47-51).

- Όπως σε όλους τους αλγορίθμους μέχρι τώρα, ο τρόπος δομής έγινε μέσω ροών που όρισαν όλες τις παραμέτρους, και έτσι η ροή εκτελέστηκε. Το ίδιο έγινε και σε αυτή την περίπτωση, όπου ορίστηκε αναγνώστης, λαμβάνοντας όλες τις τιμές και στην συνέχεια ο αγωγός εκτελέστηκε (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 54-60).
- Τέλος, το επόμενο μπλοκ εντολών απομόνωσε κάθε συνιστώσα και πραγματοποίησε μια ανακατάταξη στην ένταση του κάθε εικονοστοιχείου. Με λίγα λόγια, αφού ορίστηκαν οι αντίστοιχοι δείκτες, για κάθε συνιστώσα, απομονώθηκε η αντίστοιχη περιοχή και δημιουργήθηκε το νέο κανάλι, εξάγοντας την πληροφορία από την εικόνα και περιέχοντας το σύνολο των συνιστωσών. Στην συνέχεια, οι τιμές της έντασης κατανεμήθηκαν από 0 εώς 255 (λαμβάνοντας δηλαδή όλο το εύρος των πιθανών τιμών) και τέλος ορίστηκε ο καινούργιος αναγνώστης για κάθε συνιστώσα (παράρτημα, 5.4 Δημιουργία κύριων συνιστωσών, 64-92).

Το αποτέλεσμα του παραπάνω αλγορίθμου παρουσιάστηκε στην παρακάτω εικόνα. (Σχήμα 5.5) Ο χρήστης μπόρεσε εύκολα να παρατήσει πως η πρώτη συνιστώσα περιείχε το μεγαλύτερο μέρος της πληροφορίας, ενώ όταν προστέθηκαν και οι άλλες δύο συνιστώσες, η εικόνα περιείχε το μεγαλύτερο μέρος των πληροφοριών της. Μαζί με τις τρεις εικόνες που περιείχαν τις τρεις συνιστώσεις, δημιουργήθηκε και άλλη μια που είχε το αποτέλεσμα και των τριών συνιστωσών σε tif format.



Σχήμα 5.4: Εφαρμογή δημιουργίας κυρίων συνιστωσών στην εικόνα του Ηρακλείου

5.3 Εφαρμογή λόγων

Επόμενη εφαρμογή που πραγματοποιήθηκε στο περιβάλλον του OTB αποτέλεσε η εφαρμογή λόγων. Η χρησιμότητα των λόγων έχει παρουσιαστεί και στο προηγούμενο κεφάλαιο. Ήδη η συγκεκριμένη εφαρμογή υπήρχε σαν παράδειγμα στο OTB, καθώς οι πράξεις καναλιών γενικότερα αποτελούν μια από τις κύριες εφαρμογές τηλεπισκόπησης.

Αρχικά, όπως πραγματοποιήθηκε και στο προηγούμενο παράδειγμα, έπρεπε να δημιουργηθεί ένας φάκελος που περιείχε την εικόνα που έπρεπε να υποστεί επεξεργασία, το αρχείο του πηγαίου κώδικα για την μετατροπή και το αρχείο της CMake. Το αρχείο της CMake ήταν δομημένο με τρόπο που έχει ήδη αναφερθεί, απλά η μόνη αλλαγή που χρειάστηκε να γίνει ήταν στα πεδία του ονόματος του εκτελέσιμου κώδικα.

Το συγκεκριμένο παράδειγμα επιλύθηκε με δύο διαφορετικούς τρόπους όσον αφορά τον τρόπο παρουσίασης των αποτελεσμάτων. Ο πρώτος παρήγαγε δύο διαφορετικές εικόνες, η μία στο format της κύριας, και η δεύτερη σε format ορατό από τον χρήστη δηλαδή σε format jpg. Ο δεύτερος τρόπος εμφάνισε το αποτέλεσμα μέσω ενός viewer.

5.3.1 Ανάλυση του αρχείου του κώδικα για πράξεις μεταξύ καναλιών με παραγωγή δύο εικόνων

Το περιβάλλον του OTB, αλλά και της βιβλιοθήκης ITK που περιείχε, παρείχε πολλά φίλτρα που επέτρεπαν την λειτουργία των βασικών εργαλείων για τα διάφορα layer της εικόνας, όπως είναι οι λόγοι. Όσον αφορά την εφαρμογή των λόγων λοιπόν, η βιβλιοθήκη παρείχε την δυνατότητα να πραγματοποιηθούν πιο γενικές και πολύπλοκες μαθηματικές πράξεις σε εικόνες, με ένα απλό φίλτρο, το otb::BandMathImageFilter.

Στο σημείο αυτό παρουσιάστηκαν αναλυτικά όλες οι ενέργειες που οδήγησαν στο τελικό αποτέλεσμα.

- Όπως σε κάθε πρόγραμμα της γλώσσας C++ αρχικά υπήρχε η εισαγωγή στον κώδικα όλων των απαραίτητων βιβλιοθηκών και αρχείων επικεφαλίδων (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 20-29). Όπως ήταν εμφανές τα αρχεία αυτά αφορούσαν την εισαγωγή της εικόνας, τον ορισμό των δεικτών για ανάγνωση και εγγραφή των εικόνων που δόθηκαν σαν εισαγωγή αλλά και ως αποτέλεσμα, αλλά και της κλάσης που περιείχε τα φίλτρα των λόγων.

- Ακολούθησε ο έλεγχος για αληθή στοιχεία εισαγωγής (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 34-40). Προκειμένου να γίνει έλεγχος για την ορθότητα των στοιχείων που εισήχθηκαν στο πρόγραμμα, υπήρξε η μεταβλητή argc. Η μεταβλητή αυτή μέτραγε πόσα ήταν τα ορίσματα για το συγκεκριμένο πρόγραμμα. Στην συγκεκριμένη περίπτωση ήταν 4 και αποτελούσαν το όνομα του εκτελέσιμου προγράμματος που δημιουργήθηκε μεσώ της CMake, το όνομα της εικόνας που εισήχθηκε, το όνομα της πρώτης εικόνας που δημιουργήθηκε και ήταν στο ίδιο format με την πρώτη, το όνομα της δεύτερης εικόνας που εισήχθηκε και ήταν σε format jpg. Σε περίπτωση που κάποιο από τα στοιχεία δεν εισήχθηκε σωστά (οπότε και η μεταβλητή argc δεν πήρε 4 ορίσματα, τότε η λειτουργία του προγράμματος διακόπτεται και εμφανίζεται μήνυμα λάθους στον χρήστη.

- Ακολούθησε ένα μπλοκ εντολών στις οποίες ορίστηκαν όλες οι μεταβλητές που χρειάστηκαν για το πρόγραμμα. Αρχικά, λοιπόν, ορίστηκε το στοιχείο PixelType, το οποίο εισήχθηκε ως μεταβλητή στα περισσότερα πρότυπα (templates). Έτσι ορίστηκαν οι τύποι των περισσότερων δεικτών. Εύκολα έγινε αντιληπτό πως το πρότυπο του otb::BandMathImageFilter χρησιμοποίησε ως όρισμα το OutputImageType προκειμένου να ορίσει τον τύπο του φίλτρου που χρησιμοποιήθηκε (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 42-51).

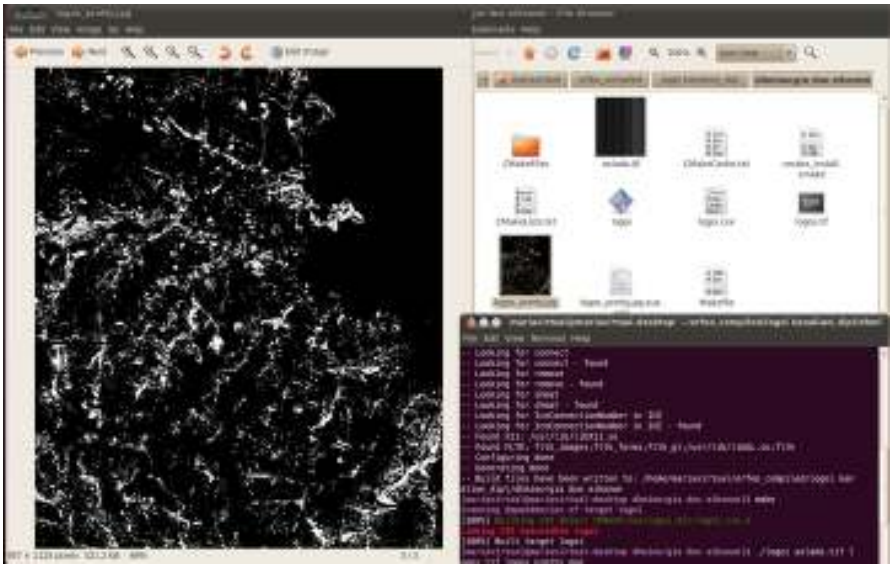
- Στην συνέχεια, ορίστηκαν οι δείκτες που χρειάστηκαν για την υλοποίηση του προγράμματος και έλαβαν τις τιμές τους αντίστοιχα. Οι δείκτες αυτοί δημιουργήθηκαν δυναμικά με την μέθοδο New(), ενώ παράλληλα πήραν και την αντίστοιχη τιμή τους. Ο αναγνώστης πήρε ως όρισμα το όνομα της εικόνας που εισήχθηκε, ενώ ο γραφέας πήρε ως όρισμα το αποτέλεσμα της εφαρμογής του φίλτρου και το εισήγαγε στην πρώτη εικόνα που εξάχθηκε και είχε το ίδιο format με την εικόνα εισαγωγής (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 53-62).

- Προκειμένου να πραγματοποιηθούν οι λόγοι των καναλιών ή οποιαδήποτε άλλη πράξη μεταξύ καναλιών, πρέπει να διαχωριστούν τα επιμέρους κανάλια από την εικόνα. Αυτό έγινε μέσω του προτύπου VectorImageToImageListType. Έτσι, με έναν απλό τρόπο

δημιουργήθηκε ένας δείκτης που πήρε ως όρισμα το αποτέλεσμα του αναγνώστη, δηλαδή όπως αναφέρθηκε παραπάνω την αρχική εικόνα. Στην συνέχεια, και μέσω μιας επαναληπτικής διαδικασίας ο δείκτης filter έλαβε κάθε κανάλι ξεχωριστά (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 64-75). Η όλη διαδικασία χρησιμοποίησε συναρτήσεις που βρίσκονταν μέσα στα αρχεία επικεφαλίδων και ορίστηκαν σε αυτά προκειμένου ο τρόπος παρουσίασης του προγράμματος να γίνει καλύτερος.

- Ακολούθησε η έκφραση με την οποία εφαρμόστηκε το φίλτρο, και πραγματοποίησε την πράξη μεταξύ των καναλιών. Στην περίπτωση αυτή, εφαρμόστηκε ο δείκτης βλάστησης, όπου δηλαδή, εάν ο λόγος (b4)/(b3) για κάθε εικονοστοιχείο της εικόνας ήταν μεγαλύτερος από την τιμή 0.4, τότε το αντίστοιχο εικονοστοιχείο έπαιρνε την τιμή 255, ενώ εάν η συνθήκη δεν ίσχυε, τότε έπαιρνε την τιμή 0. Έτσι, με αυτόν τον τρόπο, η βλάστηση παρουσιάστηκε με άσπρες αποχρώσεις, ενώ όλη η υπόλοιπη εικόνα παρουσιάστηκε με μαύρες αποχρώσεις. Τέλος, για να εκτελεστεί η όλη διαδικασία ο δείκτης πρέπει να γίνει Update (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 77-79).
- Κλείνοντας και αφού ολοκληρώθηκε όλη η διαδικασία της δημιουργίας του λόγου (με παρόμοιο τρόπο πραγματοποιήθηκαν όλοι οι λόγοι στην εικόνα), δημιουργήθηκε η δεύτερη εικόνα η οποία ήταν σε format που μπορούσε να δει ο χρήστης χωρίς χρήση κάποιου viewer. Η εικόνα αυτή ονομάστηκε από τον δημιουργό του προγράμματος PrettyImage και δημιουργήθηκε, τόσο με χρήση του προτύπου ImageFileWriter, όσο και του προτύπου CastImageFilter που βρισκόταν στην βιβλιοθήκη της ITK (παράρτημα, 5.5 Αρχείο κώδικα για λόγους 81-92).

Στην συνέχεια, και αφού αναλύθηκε διεξοδικά όλος το κώδικας, αυτός εκτελείται με την βοήθεια της CMake. Το αποτέλεσμά του είναι αυτό που φαίνεται και παρακάτω. (Σχήμα 5.5)



Σχήμα 5.5: Εφαρμογής λόγου (NDVI) δημιουργώντας δύο ξεχωριστές εικόνες για την περιοχή της Αχλάδας

5.3.2 Ανάλυση του αρχείου του κώδικα για πράξεις μεταξύ καναλιών με εμφάνιση του αποτελέσματος σε viewer

Στο κεφάλαιο αυτό το αποτέλεσμα του προγράμματος ήταν το ίδιο ακριβώς με αυτό που παρουσιάστηκε παραπάνω. Η μόνη διαφορά ήταν ο τρόπος με τον οποίο παρουσιάστηκε το

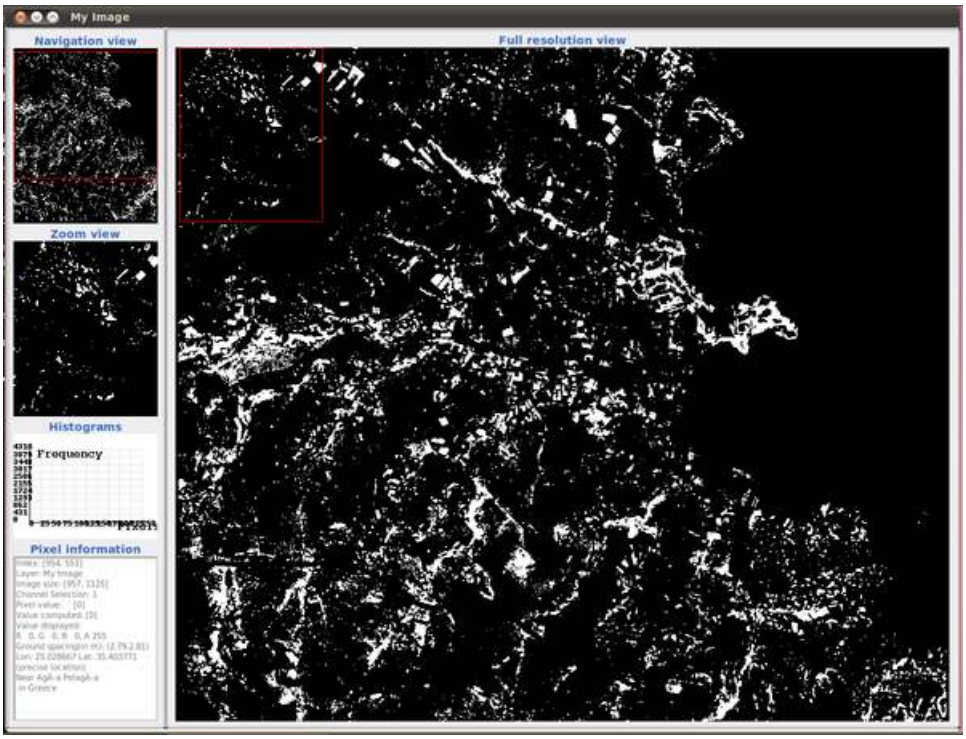
αποτέλεσμα αυτό στον χρήστη. Στην περίπτωση αυτή, λοιπόν, χρησιμοποιήθηκε ένας viewer, δηλαδή το αποτέλεσμα παρουσιάστηκε με γραφικό τρόπο παρόλο που το OTB δεν διαθέτει από μόνο του γραφικό περιβάλλον. Για το λόγο αυτό, το συγκεκριμένο περιβάλλον παρέχει την κλάση otb::ImageViewer η οποία λειτουργεί, όπως όλες οι ροές, και για αυτό αντικαθιστά την otb::ImageFileWriter.

Ο κώδικας που πραγματοποίησε τον λόγο παραμένει ο ίδιος. Η μόνη προσθήκη που έγινε ήταν για την ύπαρξη του viewer και παρουσιάζεται παρακάτω

- Αρχικά πρέπει να εισαχθεί το αντίστοιχο αρχείο επικεφαλίδα για τον viewer(παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 31).
- Το γεγονός ότι στην περίπτωση αυτή δεν παράχθηκαν δύο εικόνες αλλά μόνο μια, οδήγησε στην αλλαγή του ελέγχου εισόδου από argc !=4 σε argc !=3. Αυτό έγινε καθώς το πρώτο όρισμα ήταν το εκτελέσιμο αρχείο που είχε δημιουργήσει η CMake, ενώ το δεύτερο όρισμα ήταν η αρχική εικόνα, και το τελευταίο όρισμα ήταν το τελικό αποτέλεσμα. Σε περίπτωση που τα ορίσματα δεν ήταν σωστά τότε τερματίζεται η λειτουργία του προγράμματος (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 37).
- Στην συνέχεια, εισήχθηκε το αποτέλεσμα που ήθελε ο χρήστης να παρουσιαστεί με γραφικό τρόπο στον αντίστοιχο δείκτη. Ο δείκτης αυτός έλαβε στην ουσία το όνομα του αρχείου που δημιούργησε (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 84).
- Έπειτα δημιουργήθηκε η ροή όπως κάθε φορά. Πρώτο βήμα ήταν ο ορισμός των τύπων των εικονοστοιχείων, της εικόνας και του αναγνώστη. Το πρότυπο της κλάσης του viewer αποτελείτο από το στοιχείο ImageType (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 86-87).
- Αφού ορίστηκαν οι διάφοροι δείκτες, στην συνέχεια δημιουργήθηκαν δυναμικά με την μέθοδο New() και ο καθένας έλαβε την τιμή του. Έτσι ορίστηκαν τα διάφορα αντικείμενα (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 89-90).
- Η μέθοδος έδωσε την δυνατότητα στον χρήστη να δώσει ένα όνομα στο παράθυρο που άνοιγε. Στην συγκεκριμένη περίπτωση το όνομα που δόθηκε ήταν MyImage.(παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 94).
- Επόμενο βήμα, αποτέλεσε η εκτέλεση της ροής έτσι ώστε να παρουσιαστεί το αποτέλεσμα. Ο viewer πήρε ως όρισμα την εικόνα που έδειχνε ο δείκτης lReader (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 94-96).
- Τέλος, προκειμένου να εκτελεστεί η δημιουργία του παραθύρου, χρησιμοποιήθηκε η βιβλιοθήκη της fltk (παράρτημα, 5.6 Αρχείο κώδικα για λόγους με viewer 99).

Στο σημείο αυτό, πρέπει να τονιστεί το γεγονός ότι προκειμένου να λειτουργήσει σωστά η όλη διαδικασία, έπρεπε να προστεθεί στο αρχείο της CMake το αντίστοιχο link για την απεικόνιση. Κατά τα άλλα, το αρχείο της CMake λειτουργεί με τον τρόπο που είχε αναλυθεί παραπάνω(παράρτημα, αρχείο CMake για λόγους2 με viewer 14-15).

Τέλος, το παράθυρο του viewer που δημιουργήθηκε ήταν το ίδιο με αυτό του Monteverdi. (Σχήμα 5.6) Στο πάνω παράθυρο, δηλαδή, παρουσιάστηκε όλη η εικόνα (scroll window) και ανάλογα που κέντραρε το τετράγωνο που υπήρχε μέσα σε αυτή, εμφανίστηκε η αντίστοιχη περιοχή στο κεντρικό παράθυρο. Στο τρίτο παράθυρο (zoom window), παρουσίαζε ζουμαρισμένη μια υποπεριοχή της εικόνας στην οποία ο χρήστης είχε κεντράρει το κόκκινο τετράγωνο μέσα στο κεντρικό παράθυρο. Επίσης, δίνονταν οι πληροφορίες για τις εικονοσυντεταγμένες οποιουδήποτε σημείου.



Σχήμα 5.6: Εφαρμογή λόγου (NDVI) χρησιμοποιώντας viewer

5.4 Εφαρμογή φίλτρων

Στο κεφάλαιο αυτό πραγματοποιήθηκαν κάποια παραδείγματα για φίλτρα. Ο κώδικας του OTB περιέχει μεγάλη πληθώρα από φίλτρα (περισσότερα από 150). Πολλά από τα φίλτρα που περιέχονται στο αντίστοιχο αρχείο βοηθούν απλά για την εκτέλεση των ήδη υπαρχόντων φίλτρων. Τα περισσότερα φίλτρα που χρησιμοποιούνται στο OTB βρίσκονται στην ITK, ενώ όλες οι διαδικασίες που χρειάζονται για να πραγματοποιηθούν όλες οι ροές βρίσκονται στο OTB. Ενδεικτικά κάποια από τα διαθέσιμα φίλτρα που υπάρχουν στο OTB είναι τα ακόλουθα, ενώ όλα τα υπόλοιπα και πιο γνωστά είναι αυτά της ITK:

Φίλτρα	**Αντίστοιχη κλάση**	**Λειτουργία**
Box and Whisker	otbBoxAndWhiskerImageFilter	Η συγκεκριμένη κλάση υπολογίζει τις ακραίες τιμές με την μέθοδο των Box and Whisker
		Συνεχίζεται στην επόμενη σελίδα

Πίνακας 5.1 – συνέχεια από προηγούμενη σελίδα

Φίλτρα	**Αντίστοιχη κλάση**	**Λειτουργία**
Μορφολογικό opening and closing	otbClosingOpeningMorphological Filter	Το συγκεκριμένο φίλτρο πραγματοποιεί ένα μορφολογικό Opening σε grayscale εικόνες και στην συνέχεια πραγματοποιεί ένα μορφολογικό closing. Στην ουσία το φίλτρο αυτό τείνει να απλοποιήσει την εικόνα, σβήνοντας τις λεπτομέρειες που έχουν χαμηλότερη τιμή από αυτή του στοιχείου που χρησιμοποιείται.
Μορφολογικό closing	otbMorphologicalClosingProfile Filter	Το συγκεκριμένο φίλτρο πραγματοποιεί ένα μορφολογικό closing της εικόνας.
Μορφολογικό opening	otbMorphologicalOpeningProfile Filter	Το συγκεκριμένο φίλτρο πραγματοποιεί ένα μορφολογικό opening της εικόνας.
Ανίχνευση πυκνότητας ακμών	otbEdgeDensityImageFilter	Το συγκεκριμένο φίλτρο υπολογίζει την πυκνότητα των ακμών γύρω από ένα pixel.
Ανίχνευση ακμών	otbEdgeDetectorImageFilter	Το φίλτρο αυτό παράγει μία δυαδική εικόνα με όλες τις ακμές της αρχικής εικόνας
Ανίχνευση ακμών με Touzi	otbTouziEdgeDetectorImage Filter	Χρησιμοποιείται κυρίως για την ανίχνευση ακμών σε εικόνες SAR.
Gabor	otbGaborFilterGenerator	Το φίλτρο αυτό εφαρμόζει το φίλτρο Gabor. Χρησιμοποιείται κυρίως για ανίχνευση σχημάτων και εξαγωγή χαρακτηριστικών.

Πίνακας 5.1: Ενδεικτικά φίλτρα στο OTB

5.4.1 Φίλτρα χαμηλών συχνοτήτων

Στο συγκεκριμένο κεφάλαιο αναλύθηκε ο τρόπος με τον οποίο εφαρμόστηκε ένα φίλτρο μέσου όρου. Το φίλτρο του μέσου όρου χρησιμοποιείται κυρίως για την απομάκρυνση του θορύβου. Το φίλτρο παρήγαγε το αποτέλεσμα κάθε εικονοστοιχείου της τελικής εικόνας, υπολογίζοντας τον στατιστικό μέσο όρο από τα αντίστοιχα εικονοστοιχεία της εικόνας που εισάγεται. Το

μέγεθος της περιοχής που δημιουργήθηκε για κάθε μέσο όρο ορίστηκε από τον χρήστη. Υπάρχουν πολλοί τρόποι εφαρμογής του συγκεκριμένου φίλτρου, στην συγκεκριμένη περίπτωση δεν εφαρμόστηκε σε ένα κανάλι της εικόνας αλλά σε όλα τα κανάλια διαδοχικά προκειμένου το αποτέλεσμα να εμφανιστεί με φυσικά χρώματα.

- Αρχικά, όπως αναφέρθηκε, σε κάθε αλγόριθμο ορίστηκαν όλα τα αρχεία επικεφαλίδων. Τα αρχεία που αξίζει να αναφερθούν στο συγκεκριμένο σημείο είναι το itkMeanImageFilter.h, το οποίο όριζε το φίλτρο που χρησιμοποιήθηκε στον συγκεκριμένο αλγόριθμο, και το otbPerBandVectorImageFilter.h, το οποίο εφάρμοζε το φίλτρο διαδοχικά σε όλα τα κανάλια της εικόνας. Το τελευταίο από τα δύο αρχεία, είναι πολύ χρήσιμο για όσα φίλτρα μπορούν να το εφαρμόσουν καθώς εφαρμόζει το φίλτρο σε όλα τα κανάλια της εικόνας και έτσι το αποτέλεσμα μπορεί να απεικονιστεί σε φυσικά χρώματα. Όλα τα υπόλοιπα αρχεία έχουν αναλυθεί σε προηγούμενο κεφάλαιο (παράρτημα, 5.7 Αρχείο κώδικα για φίλτρο μέσου όρου, 26-29).

- Στην συνέχεια, έγινε ο συνηθισμένος έλεγχος των στοιχείων που εισήγαγε ο χρήστης κσι στην συγκεκριμένη περίπτωση ήταν δύο (παράρτημα, αρχείο κώδικα για φίλτρο μέσου όρου, 34-42).

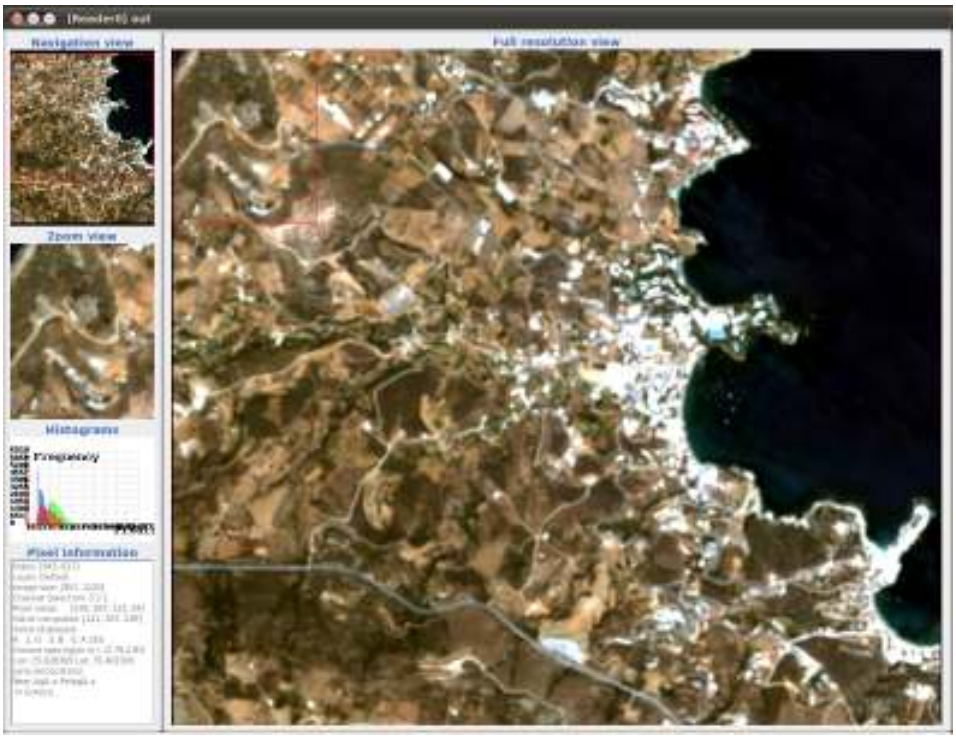
- Ορίστηκε ο αναγνώστης και έλαβε την τιμή του, ενώ παράλληλα ορίστηκαν οι τύποι VectorImageType, ImageType (παράρτημα, 5.7 Αρχείο κώδικα για φίλτρο μέσου όρου,45-53).

- Ακολούθησε η εφαρμογή του φίλτρου του μέσου όρου. Όπως κάθε φορά ορίστηκε ο αντίστοιχος δείκτης που περιείχε το αποτέλεσμα της εφαρμογής του φίλτρου. Στην συνέχεια ορίστηκε από τον χρήστη το μέγεθος του φίλτρου που εφαρμόστηκε. Για την συγκεκριμένη περίπτωση ορίστηκε ένα φίλτρο διαστάσεων 2x2. Με αυτό τον τρόπο το φίλτρο πήρε ως όρισμα όλα τα στοιχεία που χρειαζόταν (παράρτημα, 5.7 Αρχείο κώδικα για φίλτρο μέσου όρου, 55-62).

- Στην συνέχεια, ορίστηκε ο δείκτης vectorFilter. Ο δείκτης αυτός ανήκε στην κλάση PerBandVectorImageType, η οποία, όπως αναφέρθηκε, εφαρμόζει το φίλτρο σε κάθε κανάλι της εικόνας ξεχωριστά και εμφανίζει το αποτέλεσμα σε μια εικόνα. Η κλάση, λοιπόν πήρε ως ορίσματα στο πρότυπό της την εικόνα εισαγωγής, την εικόνα που δημιουργήθηκε και τον τύπο του φίλτρου (παράρτημα, 5.7 Αρχείο κώδικα για φίλτρο μέσου όρου, 65-74).

- Στην συνέχεια, δημιουργήθηκε η ροή των δεδομένων στις αντίστοιχες μεταβλητές. Δημιουργήθηκε ο δείκτης writerVector ο οποίος πήρε ως είσοδο το αποτέλεσμα του δείκτη vectorFilter. Τέλος, η όλη διαδικασία ολοκληρώθηκε με το Update του αντίστοιχου δείκτη (παράρτημα, 5.7 Αρχείο κώδικα για φίλτρο μέσου όρου, 76-79).

Το αποτέλεσμα του φίλτρου φαίνεται παρακάτω. (Σχήμα 5.7) Παρατηρείται μια θολούρα στην όλη εικόνα, πράγμα που υποδηλώνει πως το φίλτρο εφαρμόστηκε με τον σωστό τρόπο. Σε περίπτωση που οι διαστάσεις του φίλτρου ήταν μεγαλύτερες το θόλωμα της τελικής εικόνας θα ήταν μεγαλύτερο.



Σχήμα 5.7: Εφαρμογή του φίλτρου μέσου όρου με πυρήνα 2x2 για την περιοχή της Αχλάδας

5.4.2 Φίλτρα υψηλών συχνοτήτων

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, στα φίλτρα υψηλών συχνοτήτων περιέχονταν και τα φίλτρα ακμών. Έτσι και στην περίπτωση αυτή, εφαρμόστηκε το φίλτρο ανίχνευσης ακμών Sobel. Με παρόμοιο τρόπο, μπόρεσε να εφαρμοστεί οποιοδήποτε φίλτρο ακμών υπήρχε μέσα στον κώδικα του OTB.

- Ορίστηκαν και σε αυτή την περίπτωση όλα τα αρχεία επικεφαλίδες. Αυτό που διαφέρε στην συγκεκριμένη περίπτωση ήταν το itkSobelEdgeDetectionImageFilter.h το οποίο όρισε και το φίλτρο που θα χρησιμοποιηθεί (παράρτημα, 5.8 Αρχείο κώδικα για φίλτρο sobel, 28-35).

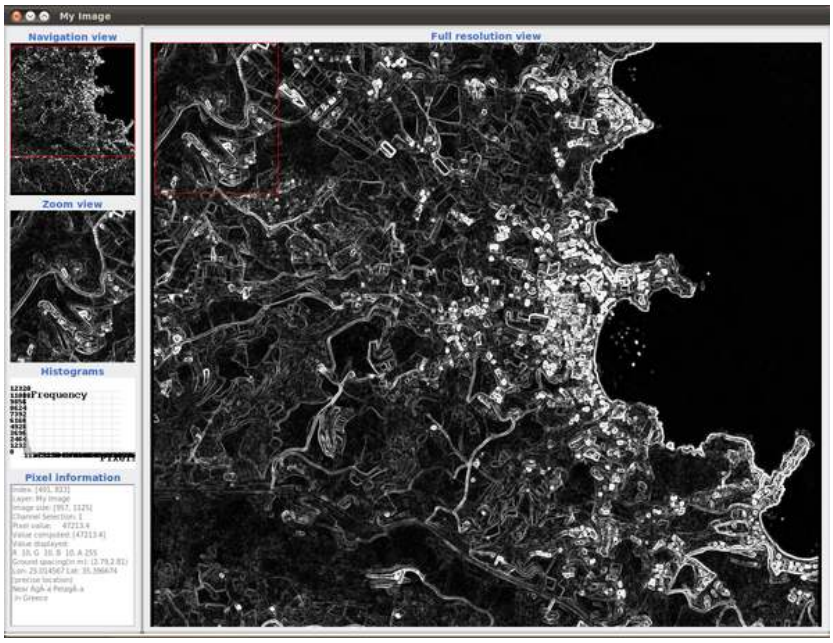
- Ακολούθησε ο ορισμός όλων των δεικτών όπως είχε αναφερθεί μέχρι τώρα. Το μόνο σημείο που χρειάστηκε αρκετή προσοχή ήταν το γεγονός ότι, το ImageType έπρεπε να ήταν τύπου double προκειμένου να αποφευχθούν λάθη στη μεταγλώττιση του προγράμματος (παράρτημα, 5.8 Αρχείο κώδικα για φίλτρο sobel, 39-42).

- Ορίστηκε το φίλτρο που χρησιμοποιήθηκε στον αλγόριθμο. Το φίλτρο πήρε ως ορίσματα στο πρότυπό του το ImageType που αποτέλεσε τύπο double (παράρτημα, 5.8 Αρχείο κώδικα για φίλτρο sobel, 44-46).

- Ακολούθησε ο ορισμός των δεικτών, η δημιουργία της διοχέτευσης ροής και τέλος, η εκτέλεσή της έτσι όπως έχει αναφερθεί μέχρι τώρα (παράρτημα, 5.8 Αρχείο κώδικα για φίλτρο sobel, 48-57).

- Τέλος, το αποτέλεσμα εμφανίστηκε στην οθόνη του χρήστη μέσω του αντίστοιχου viewer. Ο τρόπος με τον οποίο ήταν δομημένος ο κώδικας του viewer έχει αναλυθεί σε προηγούμενο κεφάλαιο (παράρτημα, 5.8 Αρχείο κώδικα για φίλτρο sobel, 61-76).

Από την παραπάνω διαδικασία προέκυψε το παρακάτω αποτέλεσμα. (Σχήμα 5.8) Παρατηρείται πως όλες οι ακμές παρουσιάστηκαν με πολύ καλό τρόπο και το αποτέλεσμα ήταν πολύ ευκρινές. Όλο το οδικό δίκτυο της περιοχής παρουσιάστηκε με μεγάλη λεπτομέρεια, καθώς εκτός από το κύριο οδικό δίκτυο, ξεχώριζε και το δευτερεύων.



Σχήμα 5.8: Εφαρμογή του φίλτρου sobel με πυρήνα 3x3 για την περιοχή της Αχλαδας

Τέλος, ενδεικτικά παρουσιάστηκε και το αποτέλεσμα από την εφαρμογή του φίλτρου GradiantMagnitudeFilter. Το συγκεκριμένο φίλτρο χρησιμοποιείται πολύ συχνά στην τηλεπισκόπηση καθώς βοηθάει κυρίως στον προσδιορισμό των ισοϋψών και τον διαχωρισμό των ομογενών περιοχών. Βέβαια, όπως φαίνεται και παρακάτω, το αποτέλεσμα (που παράχθηκε σε μορφή png) από την εφαρμογή στην εικόνα δεν βοήθησε πολύ στην εξαγωγή κάποιου συμπεράσματος. Ίσως αν πρώτα εφαρμοζόταν στην εικόνα ένα φίλτρο μέσου όρου και στην συνέχεια το συγκεκριμένο φίλτρο, το αποτέλεσμα να ήταν πιο κατανοητό.

Σχήμα 5.9: Εφαρμογη του φίλτρου Gradient Magnitude για την περιοχή της Αχλάδας

5.4.3 Φίλτρα μορφολογίας

Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, τα μορφολογικά φίλτρα έχουν ένα μεγάλο εύρος εφαρμογής. Το περιβάλλον του OTB έχει τα βασικά φίλτρα του dilation και του erosion, με βάση τα οποία δημιουργούνται όλα τα υπόλοιπα. Το συγκεκριμένο περιβάλλον υποστηρίζει τόσο την δυαδική μορφολογία binary morphology όσο και την μορφολογία που το αποτέλεσμα εμφανίζεται σε τόνους του γκρι grayscale morphology. Στην συγκεκριμένη περίπτωση αναλύθηκε η grayscale morphology, όπου το αποτέλεσμα ήταν περισσότερο ορατό στον χρήστη. Οι εικόνες που προέκυψαν από την εκτέλεση του αλγορίθμου, ήταν τόσο η μορφολογική διαστολή (dilation), όσο και η μορφολογική συστολή (erosion) της αρχικής εικόνας. Από την στιγμή λοιπόν, που υπήρχαν οι δύο αυτές βασικές λειτουργίες,όλες οι άλλες προέκυψαν από τον συνδυασμό τους. Αναλυτικότερα τώρα ακολούθησε η ανάλυση του παρακάτω αλγορίθμου.

- Αρχικά, ορίστηκαν, όπως κάθε φορά, τα αρχεία επικεφαλίδες. Ο χρήστης μπόρεσε να παρατηρήσει πως ορίστηκε η otbImage και όχι η VectorImage με αποτέλεσμα ο αλγόριθμος να λειτουργεί σωστά μόνο με μια μονοκάναλη εικόνα, όπως είχε προκύψει από παραπάνω διαδικασία. Επίσης, παρατηρήθηκαν και τα αντίστοιχα αρχεία επικεφαλίδες για την συστολή και την διαστολή καθώς επίσης και για το δομικό στοιχείο που χρησιμοποιήθηκε για την εφαρμογή των φίλτρων (παράρτημα, 5.9 Αρχείο Κώδικα για μορφολογίες, 23-28).

- Στην συνέχεια, έγινε ο καθιερωμένος έλεγχος για τα ορίσματα που έλαβε ο αλγόριθμος, ώστε ο χρήστης να μην εισάγει λάθος αριθμό ορισμάτων. Στον συγκεκριμένο αλγόριθμο, ο χρήστης εισήγαγε και τα όρια με τα οποία πραγματοποιήθηκε η ενίσχυση των τελικών αποτελεσμάτων. Τα προτεινόμενα όρια ήταν το 150 για κατώτερο και το 180 για ανώτατο (παράρτημα, 5.9 Αρχείο Κώδικα για μορφολογίες, 33-40).

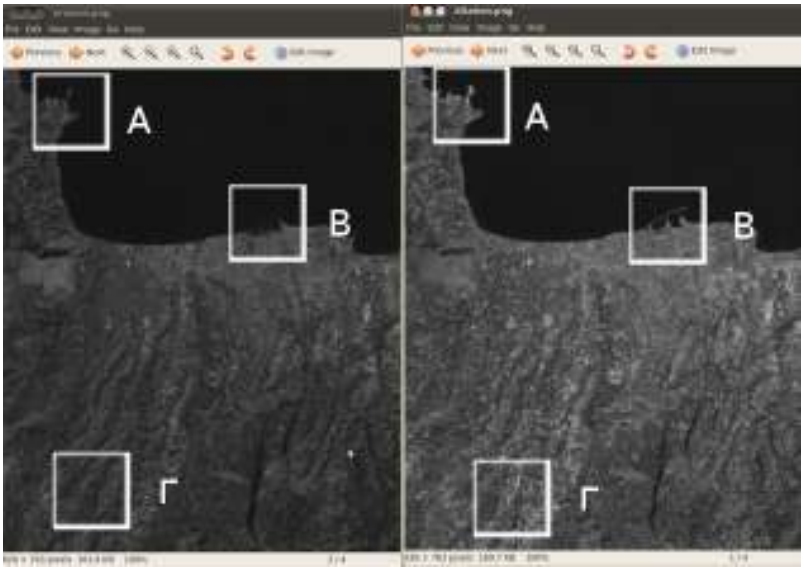
- Στην συνέχεια, ακολούθησε ο ορισμός όλων των μεταβλητών που χρησιμοποιήθηκαν στον αλγόριθμο, με το συνηθισμένο τρόπο έγινε σε όλους τους αλγορίθμους μέχρι τώρα. Επίσης, ορίστηκαν και οι μεταβλητές που έδιναν ως όρισμα στα αντίστοιχα πρότυπα. Όπως μπορεί εύκολα να παρατηρήσει ο χρήστης, ο τύπος των εικόνων ήταν για άλλη μια φορά unsigned char οπότε απευθυνόταν σε 8bit εικόνες. Έπειτα, ορίστηκαν όλες οι μεταβλητές που χρησιμοποιήθηκαν και δεσμεύτηκε χώρος στην μνήμη (παράρτημα, 5.9 Αρχείο Κώδικα για μορφολογίες, 42-67).

- Αφού έγιναν όλες οι παραπάνω διαδικασίες, ορίστηκε το δομικό στοιχείο με το οποίο πραγματοποιήθηκε η εφαρμογή του φίλτρου. Στην περίπτωση του συγκεκριμένου αλγορίθμου, το δομικό στοιχείο αντιστοιχούσε σε ένα κύκλο με ακτίνα 3. Σε περίπτωση που ο χρήστης ήθελε να το μεγαλώσει απλά μπορούσε να αυξήσει τον αριθμό στο structuringElement.SetRadius(1). Αφού λοιπόν, ορίστηκε το δομικό στοιχείο, στην συνέχεια, εισήχθηκαν στα αντίστοιχα κελιά για κάθε φίλτρο (παράρτημα, 5.9 Αρχείο Κώδικα για μορφολογίες, 69-74).

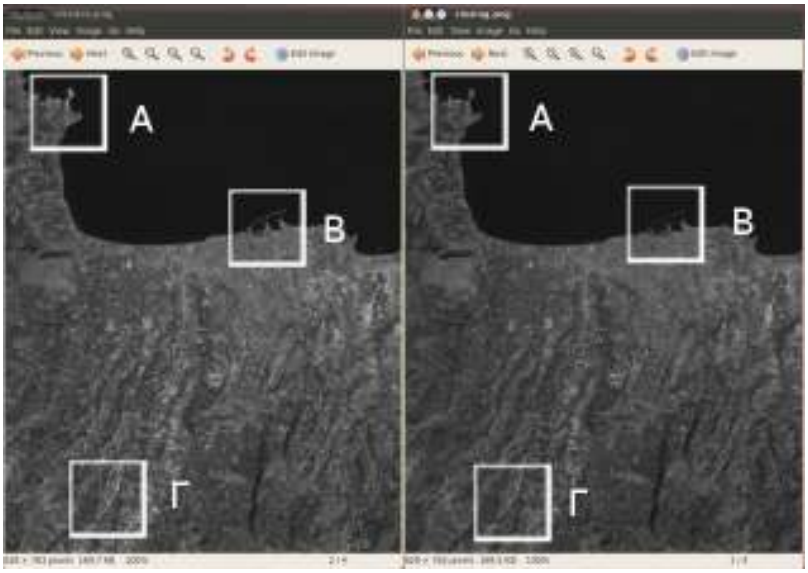
- Τέλος, δημιουργήθηκε η διοχέτευση ροής που οδήγησε στο αποτέλεσμα, αφού περαστούν όλα τα ορίσματα στις αντίστοιχες μεταβλητές. Έπειτα εκτελέστηκε η ροή και το αποτέλεσμα εμφανίστηκε στον χρήστη(παράρτημα, 5.9 Αρχείο Κώδικα για μορφολογίες, 180-211).

Το αποτέλεσμα του αλγορίθμου φαίνεται παρακάτω. (Σχήμα 5.10). Στην αριστερή εικόνα παρατηρείται το αποτέλεσμα της συστολής, με σημειωμένα κάποια χαρακτηριστικά σημεία που παρουσιάζουν αλλαγή. Το δομικό στοιχείο που χρησιμοποιήθηκε ήταν αρκετό για να εξαφανιστούν από την εικόνα τα μικρά στοιχεία, όπως την άκρη του λιμανιού του Ηρακλείου, όπως και

πλήθος άλλων λεπτομερειών. Η δεξιά εικόνα αντιστοιχεί στο αποτέλεσμα της διαστολής, με σημειωμένες τις αντίστοιχες περιοχές. Σε αυτή την περίπτωση προέκυψε το ακριβώς αντίθετο αποτέλεσμα. Όλα τα μικρά στοιχεία ενισχύθηκαν. Έτσι τονίστηκαν όλες οι λεπτομέρειες της εικόνας. Τέλος, ο χρήστης διέκρινε και στις δύο περιπτώσεις το δομικό στοιχείο που χρησιμοποιήθηκε για τα μορφολογικά φίλτρα, καθώς δημιούργησε μια ασυνέχεια στους τόνους της εικόνας.



Σχήμα 5.10: Εφαρμογή του φίλτρου της Μορφολογίας. Αριστερά η εικόνα της συστολής, Δεξιά η εικόνα της διαστολής



Σχήμα 5.11: Αποτέλεσμα εφαρμογής του φίλτρου της Μορφολογίας κλεισίματος. Αριστερά η εικόνα της διαστολής, Δεξιά η εικόνα του κλεισίματος

Στην συνέχεια, κατασκευάστηκε ο αλγόριθμος του μορφολογικού κλεισίματος, όπως προέκυψε από το Σχήμα 5.11. Το μορφολογικό κλείσιμο δημιουργήθηκε όταν σε μια διαστολή εφαρμόστηκε μια συστολή. Το μορφολογικό κλείσιμο χρησιμοποιήθηκε για την λείανση της

εικόνας και ταυτόχρονα την ενίσχυση των μικρών στοιχείων της. Ο χρήστης παρατηρεί πως το δομικό στοιχείο που χρησιμοποιήθηκε, γίνεται αντιληπτό στην εικόνα, αλλά σε μικρότερο βαθμό αυτή την φορά, ενώ η τελική εικόνα δεν έχει μεγάλες διαφορές από την αρχική.

5.5 Εφαρμογή ταξινομήσεων

5.5.1 Μη επιβλεπόμενη ταξινόμηση

Το περιβάλλον του OTB παρέχει την δυνατότητα αξιοποίησης του αλγορίθμου KMeans για μη επιβλεπόμενη ταξινόμηση. Στο σημείο αυτό πρέπει να αναφερθεί ο τρόπος με τον οποίο εφαρμόζεται ο συγκεκριμένος αλγόριθμος. Ο αλγόριθμος KMeans (Κ-μέσων) είναι ένας αλγόριθμος που ομαδοποιεί αντικείμενα βάσει των χαρακτηριστικών των Κ μεριδίων. Υποθέτει ότι τα χαρακτηριστικά του αντικειμένου δημιουργούν ένα χώρο διανυσμάτων και ο σκοπός του είναι να ελαχιστοποιήσει τη συνολική διακύμανση της ομάδας ή τη συνάρτηση τετραγωνικού σφάλματος. [Αργιαλάς, 1999]

$$\text{ΟΤΣ}=\sum_{i=1}^{k}\sum_{x_i \in S_i} \|x - m_i\|^2$$

όπου:

ΟΤΣ= Ολικό τετραγωνικό σφάλμα

k, οι ομάδες που υπάρχουν

m, είναι ο μέσος όρος της i συσσώρευσης ή αλλιώς το κεντροειδές

Τα βασικά βήματα του αλγορίθμου είναι τα εξής:

1. Επιλογή του αριθμού των ομάδων (συσσωρεύσεων). Με λίγα λόγια, γίνεται η αρχικοποίηση του αλγορίθμου. Η επιλογή των κέντρων m σε αυτό το στάδιο είναι αυθαίρετη με την εξαίρεση ότι οποιοδήποτε ζευγάρι δεν μπορεί να αποτελείται από αυτά τα κέντρα. Είναι βέβαια σκόπιμο, η επιλογή των κέντρων αυτών να γίνει κατά τρόπο ομοιόμορφο μέσα από τα δεδομένα, έτσι ώστε να αποφευχθεί η δημιουργία ανωμάλων συσσωρεύσεων και να συντομευτεί η σύγκλιση του αλγορίθμου.

2. Η θέση x του κάθε εικονοστοιχείου της εικόνας εξετάζεται και το εικονοστοιχείο αποδίδεται στη εγγύτερη υποψήφια συσσώρευση. Ο εγγύτερος γείτονας βρίσκεται με βάση την ευκλείδια απόσταση ή την απόλυτη διαφορά των κέντρων.

3. Υπολογίζονται τα κέντρα (μέσοι) των νέων συσσωρεύσεων.

4. Επανάληψη μέχρι να συγκλίνει ο αλγόριθμος. Στην περίπτωση που ο αλγόριθμος δεν συγκλίνει, επαναπροσδιορίζονται με τις τρέχουσες τιμές και ο αλγόριθμος επιστρέφει στο βήμα 2.

Κεφάλαιο 6

Εφαρμογές ανίχνευσης μεταβολών στο OTB

6.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο αναλύθηκαν οι αλγόριθμοι ανίχνευσης μεταβολών που υπάρχουν μέσα στο περιβάλλον του OTB. Η ανίχνευση μεταβολών είναι η διαδικασία αναγνώρισης διαφορών σε αντικείμενα και φαινόμενα, τα οποία παρατηρούνται μέσα από διαχρονικές εικόνες της ίδιας περιοχής. Για την συγκεκριμένη διαδικασία έχουν αναπτυχθεί πληθώρα μεθόδων που χρησιμοποιούν τηλεπισκοπικά δεδομένα. Όλες όμως οι τεχνικές στοχεύουν στην ανάπτυξη αξιόπιστων αυτοματοποιημένων διαδικασιών, για την εξαγωγή των γεωχωρικών πληροφοριών από διάφορους τύπους τηλεπισκοπικών απεικονίσεων. Οι εφαρμογές που έχουν τα αποτελέσματα από την διαδικασία ανίχνευσης μεταβολών είναι πολλά, κάποια από αυτά παρουσιάζονται παρακάτω: [David Grey, 2000]

- Για αλλαγές που πραγματοποιούνται στην γη και γενικότερα στην γήινη επιφάνεια με το πέρας του χρόνου.
- Για αλλαγές που πραγματοποιούνται σε σημαντικά ιστορικά μνημεία, για τοπική τεχνογνωσία των αλλαγών.
- Για παροχή ιστορικών στοιχείων για δάση και οποιοδήποτε άλλο στοιχείο άξιο παρακολούθησης.
- Παρουσίαση των αλλαγών με απτά χαρακτηριστικά όπως σχήμα, μέγεθος, συνεκτικότητα κ.τ.λ. και συσχέτισή τους με στοιχεία όπως το νερό, η άγρια φύση.
- Για την κατανόηση πιθανών αιτιών αλλαγής οποιουδήποτε αντικειμένου.
- Για την κατασκευή και ταχεία αναθεώρηση χαρτών.

Όπως αναφέρθηκε και παραπάνω, οι μέθοδοι ανίχνευσης μεταβολών με τηλεπισκοπικά δεδομένα, που έχουν μελετηθεί και αναπτυχθεί από διαφόρους ερευνητές, είναι ποικίλες και πολυάριθμες. Ορισμένες αποτελούν παραλλαγή ή συνδυασμό κάποιων βασικών και διαδεδομένων μεθόδων. Οι μέθοδοι ανίχνευσης μεταβολών διαχωρίζονται σε φωτοερμηνευτικές και ψηφιακές μεθόδους. Οι φωτοερμηνευτικές είναι λίγες σε αριθμό, αλλά πολύ χρήσιμες, αφού σε αυτές βασίζεται ο αναλυτής για την πρώτη εκτίμηση της μεταβολής και για την μετέπειτα αξιολόγηση των αποτελεσμάτων των ψηφιακών μεθόδων. Οι ψηφιακές μέθοδοι είναι πολυάριθμες και ορισμένες από αυτές αρκετά πολύπλοκες, δίνουν όμως πιο αξιόπιστα αποτελέσματα άμα εφαρμοστούν σωστά.

Εκτός όμως από τον αλγόριθμο, τα αποτελέσματα που θα προκύψουν εξαρτώνται και από τα τηλεπισκοπικά δεδομένα που διαθέτει ο χρήστης. Αρχικά, ανάλογα με την διακριτική ικανότητα αλλά και την φασματική ικανότητα των τηλεπισκοπικών δεκτών και κατ΄ επέκταση των τηλεπισκοπικών δεδομένων, επιλέγεται ο αλγόριθμος, αλλά και η εφαρμογή που θα πραγματοποιηθεί. Στην συνέχεια, τα δεδομένα πρέπει να περάσουν από μια αρχική προεπεξεργασία. Η προεπεξεργασία αυτή αφορά την γεωμετρική και ραδιομετρική διόρθωση τους. Η ανίχνευση μεταβολών απαιτεί ακριβή χωρική εγγραφή των εικόνων. Σε περίπτωση που υπάρχει έστω και ένα πολύ μικρό σφάλμα στη γεωμετρία της εικόνας, αυτό μπορεί να μεταφερθεί σε εσφαλμένα αποτελέσματα σε όλο το εύρος των εικόνων. Επίσης, η ραδιομετρική διόρθωση είναι χρήσιμη για το διαχωρισμό των μεταβολών ενδιαφέροντος από άλλες, που οφείλονται στις ατμοσφαιρικές συνθήκες, τις συνθήκες φωτισμού, τη γωνία λήψης του δορυφόρου, την εδαφική υγρασία κ.τ.λ.

Τέλος, η αξιολόγηση της ακρίβειας των παραγόμενων μεταβολών είναι μια πολύ δύσκολη διαδικασία, καθώς, απαιτεί τον καθορισμό των αλλαγών που έγιναν στην περιοχή ενδιαφέροντος. Οι αλλαγές αυτές όμως μπορεί να αλλάξουν από μέρα σε μέρα οπότε η όλη διαδικασία είναι πολύ δύσκολη. Προκειμένου να αντιμετωπιστεί ως ένα βαθμό το πρόβλημα, δημιουργούνται συγκεντρωτικοί πίνακες που περιέχουν στατιστικά στοιχεία, όπως αποκλίσεις και έτσι προκύπτει μια θεωρητική αντιμετώπιση του αποτελέσματος. Ένας άλλος τρόπος αξιολόγησης του παραγόμενου αποτελέσματος είναι, ψηφιοποιώντας τις αλλαγές φωτοερμηνευτικά και απλά συγκρίνοντας το τελικό αποτέλεσμα. [Καραλής, 2008]

Συνοψίζοντας όλα τα παραπάνω, τα βασικά στάδια μιας εφαρμογής ανίχνευσης μεταβολών είναι τα παρακάτω:

1. **Συλλογή τηλεπισκοπικών δεδομένων**: Οι εικόνες που θα χρησιμοποιηθούν στην διαδικασία παίζουν πολύ σημαντικό ρόλο. Εκτός από το γεγονός ότι πρέπει να είναι διαθέσιμες στην αρχή και στο τέλος της περιόδου αλλαγής, καλό θα είναι να έχουν την ίδια χωρική, φασματική και ραδιομετρική ανάλυση.

2. **Επεξεργασία εικόνων**: Στο σημείο αυτό γίνεται μια σωστή γεωμετρική και ραδιομετρική διόρθωση των εικόνων. Καλό είναι από τις δύο εικόνες να έχει επιλεχθεί η μια ως κύρια ημερομηνία και όλες οι διορθώσεις να γίνουν με βάση αυτή.

3. **Επιλογή και εφαρμογή αλγορίθμου**: Έχοντας επεξεργαστεί όλα τα δεδομένα ο χρήστης πρέπει να επιλέξει τον κατάλληλο αλγόριθμο για την εφαρμογή του και να τον εφαρμόσει.

4. **Στατιστική ανάλυση**: Αφού έχει προκύψει το αποτέλεσμα με χρήση κάποιου αλγορίθμου, δημιουργούνται συγκεντρωτικοί πίνακες που περιέχουν στατιστικά και ποιοτικά στοιχεία για αυτό. Στο στάδιο αυτό μπορεί να γίνει και μια οπτική ανάλυση του αποτελέσματος.

5. **Εξαγωγή αποτελέσματος**: Έχοντας γίνει και η αξιολόγηση, στο σημείο αυτό το αποτέλεσμα μπορεί να χρησιμοποιηθεί για οποιαδήποτε χρήση. Με αυτό τον τρόπο η όλη διαδικασία ολοκληρώνεται.

6.2 Ανίχνευση μεταβολών στο ΟΤΒ

Οι αλγόριθμοι που περιέχει το ΟΤΒ έχουν να κάνουν με επεξεργασία ενός ζευγαριού εικόνων που περιέχουν πληροφορία πριν και μετά το φαινόμενο που μελετάται. Στο συγκεκριμένο περιβάλλον, εκτός από μια σειρά αλγορίθμων που απευθύνονται σε οπτικά δεδομένα, υπάρχει και μια σειρά για δεδομένα που προέρχονται από ραντάρ. Έτσι και αλλιώς τα προβλήματα που

έχουν οι οπτικοί δέκτες και έχουν να κάνουν με την εξάρτησή τους από καιρικές συνθήκες και από τις συνθήκες φωτισμού καλύπτονται από τους ενεργητικούς δέκτες. Βέβαια, τα κύρια προβλήματα που έχουν σχέση με τους αλγορίθμους ανίχνευσης μεταβολών είναι τεσσάρων ειδών.

1. Στην περίπτωση των δεδομένων ραντάρ, ο θόρυβος που διαθέτουν, κάνει την εικόνα δύσκολη στην επεξεργασία.

2. Η γεωμετρική διόρθωση των εικόνων που χρησιμοποιούνται μπορεί να παράξει εικόνες που είναι δύσκολο να συγκριθούν.

3. Επίσης, το χρονολογικό κενό μεταξύ των δύο λήψεων και η αλλαγή στα στοιχεία του δέκτη (π.χ. αλλαγή στο καλιμπράρισμα του δέκτη) μπορεί να προκαλέσει ποικιλία προβλημάτων που είναι δύσκολο να αντιμετωπιστούν.

4. Τέλος, η φυσική εξέλιξη της περιοχής που παρακολουθείται δεν πρέπει σε καμία περίπτωση να συγχέεται με αλλαγές που πρέπει να ανιχνευτούν από τους αλγορίθμους.

Τρεις είναι οι μέθοδοι που χρησιμοποιούνται στο ΟΤΒ και έχουν σχέση με ανίχνευση μεταβολών: [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

- **Σύγκριση ταξινομήσεων**: Με την μέθοδο αυτή, δημιουργούνται για κάθε εικόνα από ένας θεματικός χάρτης και στην συνέχεια, συγκρίνονται οι δύο αυτοί χάρτες. Τα αποτελέσματα δηλαδή, προκύπτουν από την σύγκριση των δύο αυτών χαρτών.

- **Κοινή ταξινόμηση**: Με την συγκεκριμένη μέθοδο, οι μεταβολές προκύπτουν από μια κοινή ταξινόμηση και των δύο εικόνων. Η ταξινόμηση δηλαδή δεν γίνεται για κάθε εικόνα ξεχωριστά αλλά οι δύο εικόνες συνδυάζονται σε μία και η ταξινόμηση γίνεται στην τελική εικόνα.

- **Άλγεβρα εικόνων**: Η μέθοδος αυτή, αναφέρεται σε εφαρμογή διάφορων αλγορίθμων (όπως διαφοράς, λόγου κ.τ.λ.) και ενίσχυση του αποτελέσματος, έτσι ώστε να είναι εμφανείς οι αλλαγές που υπάρχουν. Από τον αλγόριθμο παράγεται ο θεματικός χάρτης με τις αλλαγές.

Το κεφάλαιο αυτό θα πραγματευτεί την τρίτη μέθοδο, δηλαδή τους απλούς αλγορίθμους καθώς αυτοί είναι που παρουσιάζουν το μεγαλύτερο ενδιαφέρον και έχουν ξεχωριστό τρόπο παρουσίασης. Έτσι και αλλιώς στην συνέχεια, θα δημιουργηθούν ξεχωριστοί αλγόριθμοι που να παράγουν εικόνες ανίχνευσης μεταβολών.

6.2.1 Γενικότερο πλαίσιο αλγορίθμων στο ΟΤΒ

Όπως όλοι οι αλγόριθμοι στο ΟΤΒ έτσι και οι αλγόριθμοι ανίχνευσης μεταβολών, λειτουργούν με βάση τις αρχές του γενικευμένου προγραμματισμού. Με λίγα λόγια, λειτουργούν με χρήση διαφόρων προτύπων, έτσι ώστε η όλη διαδικασία να γίνει αυτοματοποιημένη και να λειτουργεί με πλήθος εικόνων. Προκειμένου να γίνουν σωστά όλες οι διαδικασίες καλό είναι να αναλυθούν λίγο τα πρότυπα που χρησιμοποιούνται. [OTB doxygen, 2011]

Αρχικά, στην συγκεκριμένη οικογένεια αλγορίθμων, χρησιμοποιείται ένα φίλτρο που λαμβάνει ως όρισμα στο πρότυπό του τις δύο εικόνες και ένα διεκπεραιωτή (functor) που περιέχει τον αλγόριθμο, και παράγει μια εικόνα με το επιθυμητό αποτέλεσμα. Τα πρότυπα αυτά είναι τέσσερα διαφορετικά και ανάλογα με την εφαρμογή ο χρήστης εφαρμόζει κάθε φορά από ένα:

- **BinaryFunctorImageFilter**: Αποτελεί ένα φίλτρο παρεχόμενο από την ITK το οποίο απλά λαμβάνει στο πρότυπό του τις δύο εικόνες και τον αλγόριθμο και δίνει ως έξοδο το τελικό αποτέλεσμα. Οι εικόνες μπορεί να είναι οποιοδήποτε τύπου και μεγέθους.

- **BinaryFunctorNeighborhoodImageFilter**: Το συγκεκριμένο φίλτρο πραγματοποιεί την ίδια ακριβώς δουλειά με την μόνη διαφορά ότι παρέχει την δυνατότητα να παραχθούν αλγόριθμοι που χρησιμοποιούν μια γειτονία από εικονοστοιχεία. Διαθέτουν δηλαδή και ένα στοιχείο radius το οποίο εφαρμόζει το φίλτρο στην ανάλογη γειτονιά. Τέλος, είναι παρεχόμενο από το OTB.

- **BinaryFunctorNeighborhoodJoinHistogramImageFilter**: Πρόκειται για ένα παρεχόμενο από το OTB φίλτρο, το οποίο παρέχει όλα τα παραπάνω αλλά παράλληλα χρησιμοποιεί για την παραγωγή του αποτελέσματος και το ιστόγραμμα των εικόνων.

- **BinaryFunctorNeighborhoodVectorImageFilter**: Ακολουθεί την παραπάνω λογική με την μόνη διαφορά πως είναι το κατάλληλο φίλτρο για vector εικόνες. Για άλλη μια φορά, το συγκεκριμένο φίλτρο παρέχεται από το OTB.

Ένα ακόμα αρχείο επικεφαλίδα (header) που χρησιμοποιείται είναι αυτό των επαναληπτών (iterator). Προκειμένου ο αλγόριθμος να αποκτήσει πρόσβαση στα εικονοστοιχεία μια εικόνας χρησιμοποιούνται οι επαναλήπτες, καθώς προσφέρουν έναν γρήγορο και απλό τρόπο για επεξεργασία εικόνων. Και στην συγκεκριμένη περίπτωση, υπάρχει μια σειρά από αρχεία, από τα οποία ο χρήστης μπορεί να επιλέξει ανάλογα με την εφαρμογή του το ανάλογο. Επιγραμματικά τα αντίστοιχα αρχεία που υπάρχουν είναι τα ακόλουθα: itk::ImageIterator, itk::ImageIteratorWithIndex, itk::ConstNeighborhoodIterator. Τα αρχεία αυτά λειτουργούν με παρόμοιο τρόπο όπως τα παραπάνω, με την μόνη διαφορά πως παίρνουν ως όρισμα την εικόνα και της δημιουργούν τον ανάλογο επαναλήπτη.

Έτσι, κάθε αλγόριθμος ανίχνευσης μεταβολών αποτελείται από τρία ξεχωριστά αρχεία. Τα δύο πρώτα είναι αρχεία επικεφαλίδες. Το πρώτο, ορίζει το πρότυπο που απαιτείται για την εκτέλεση του αλγορίθμου και παίρνει τα αρχικά ορίσματα, ενώ το δεύτερο ορίζει τον functor, ο οποίος περιέχει το εκτελεστικό κομμάτι του αλγορίθμου. Τέλος, πρέπει να υπάρχει το εκτελέσιμο αρχείο cxx, το οποίο να δημιουργεί όλες τις διοχετεύσεις ροών που χρειάζονται για να εκτελεσθεί με το σωστό τρόπο το πρόγραμμα. Όλα αυτά βέβαια πρέπει να συνοδεύονται με το αντίστοιχο αρχείο της CMake.

Στο σημείο αυτό, καλό είναι να αναφερθούν συγκεντρωτικά οι αλγόριθμοι ανίχνευσης μεταβολών που διαθέτει το συγκεκριμένο εργαλείο. Βέβαια, ένα σημείο που πρέπει να τονιστεί είναι πώς όλοι οι αλγόριθμοι που θα αναφερθούν και δουλεύουν με οπτικά δεδομένα επεξεργάζονται μόνο μονοκάναλες εικόνες. Έτσι, ο χρήστης θα πρέπει να έχει φροντίσει πριν τις αξιοποιήσει να εξάγει ένα κανάλι τους ή να εφαρμόσει ένα αλγόριθμο μέσου όρου ή σταθμευμένου μέσου όρου στα εικονοστοιχεία τους.

Αλγόριθμος	**Λειτουργία**
Αλγόριθμος διαφοράς	Πραγματοποιεί τη αλγεβρική διαφορά των δύο εικόνων, ανά κανάλι, ενώ στις περιοχές που ανιχνεύεται η μεταβολή τα εικονοστοιχεία έχουν τιμή ανάλογη με αυτή, ενώ αυτά που δεν παρουσιάζουν αλλαγή η τιμή τους είναι 0. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]
Αλγόριθμος λόγου	Πραγματοποιεί το λόγο των δύο εικόνων, ανά κανάλι. Έχει πολύ καλά αποτελέσματα, τόσο σε οπτικά, όσο και σε δεδομένα ραντάρ καθώς με τον λόγο εξαλείφεται ο θόρυβος.
Αλγόριθμος Kullback-Leibler	Πρόκειται για αλγόριθμο που στηρίζεται στην διαφορά μεταξύ των τοπικών τυχαίων σημείων. Λειτουργεί κυρίως για δεδομένα από ραντάρ ενώ μπορεί να επεξεργαστεί και συνδυασμό εικόνων ραντάρ και οπτικών. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011] [Γρεγοιρε Μερσιερ ανδ Θορδι Ινγλαδα, 2008]
Αλγόριθμος τοπικής συσχέτισης	Ο αλγόριθμος υπολογίζει την πιθανότητα να υπάρχει μια γραμμική σχέση μεταξύ δύο τυχαίων τιμών. Στην ουσία συγκρίνει τις διασπορές των δύο εικόνων και παράγει το αντίστοιχο αποτέλεσμα. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]
Αλγόριθμος LHMI (Localized Harmonic Motion Imaging)	Πρόκειται για έναν αλγόριθμο που συσχετίζει τις εντροπίες για κάθε υποπεριοχή των εικόνων και παρουσιάζει τα αποτελέσματα. Χρησιμοποιείται για ανίχνευση των πολύ μεγάλων μεταβολών
Αλγόριθμος CBAMI (Cumulant based approximation to Mutual Information)	Αποτελεί μια μέθοδο η οποία έγκειται στην εφαρμογή διαδοχικών παλινδρομήσεων ώστε στην συνέχεια, με μια εφαρμογή διαφοράς να αναδειχθούν οι πραγματικές αλλαγές.

Πίνακας 6.1: Διαθέσιμοι αλγόριθμοι ανίχνευσης μεταβολών στο ΟΤΒ

6.2.2 Δεδομένα επεξεργασίας

Όπως αναφέρθηκε παραπάνω προκειμένου να εφαρμοστούν οι αλγόριθμοι ανίχνευσης μεταβολών με το σωστό τρόπο, πρέπει να γίνουν πρώτα οι γεωμετρικές και ραδιομετρικές διορθώσεις των εικόνων. Στην παρούσα διπλωματική αυτό δεν χρειάστηκε να γίνει, καθώς οι εικόνες που χρησιμοποιήθηκαν ήταν ήδη γεωμετρικά και ραδιομετρικά διορθωμένες. Έτσι και αλλιώς δίνεται περισσότερη έμφαση στους αλγορίθμους ανίχνευσης μεταβολών που παρέχονται από το περιβάλλον ΟΤΒ.

Οι εικόνες που χρησιμοποιήθηκαν είναι ένα ζευγάρι Ikonos που παρουσιάζουν τμήμα της Αττικης Οδού που εκτείνεται από τον κόμβο της Εθνικής Οδού (έξοδος 8) έως τον κόμβο της Λ. Κηφισίας (έξοδος 11). Η πρώτη εικόνα ελήφθη το έτος 2000, δηλαδή πριν την έναρξη των

έργων, και η δεύτερη το έτος 2007, οπότε και είχαν ολοκληρωθεί οι εργασίες. Μια παρουσίαση των δύο εικόνων γίνεται στην εικόνα Σχήμα 6.1.

Στο σημείο αυτό κρίνεται σκόπιμο να παρουσιαστούν κάποια στοιχεία για τις δύο εικόνες.

	Εικόνα έτους 2000	**Εικόνα έτους 2007**
Sensor Type	IKONOS-2	IKONOS-2
Processing Level	Orthorectified	Standard Geometrically Corrected
Map Projection	Transverse Mercator	UTM Zone 34 North
Datum	GGRS87	WGS84
Product Pixel Size	1.00 meters	1.00 meters
Product Map Units	meters	meters
File Format	GeoTIFF	GeoTIFF
Scan Azimuth	0.00 degrees	0.00 degrees
Sun Angle Azimuth	129.7444 degrees	130.1260 degrees
Sun Angle Elevetion	63.96003 degress	68.45852 degrees
Acquisition Date/Time	2000-05-17 08:57	2007-07-05 09:19 GMT
Percent Cloud Cover	0	0

Πίνακας 6.2: Στοιχεία από τα μεταδεδομένα των εικόνων

Σχήμα 6.1: Εμφάνιση των δύο εικόνων Ikonos που χρησιμοποιήθηκαν. Δεξιά εικόνα 2000, Αριστερά εικόνα 2007

Από τα παραπάνω στοιχεία προκύπτουν τα εξής συμπεράσματα για τις δύο εικόνες:

- Τα δεδομένα προέρχονται από ίδιου τύπου δορυφόρο.
- Η ημερολογιακή διαφορά των λήψεων είναι 49 ημέρες.
- Η ώρα λήψης είναι παραπλήσια.

- Η χωρική ανάλυση είναι η ίδια.
- Η γωνία λήψης του δορυφόρου είναι η ίδια.
- Η γωνία ανύψωσης του ηλίου κατά τη χρονική στιγμή της λήψης είναι παραπλήσια.
- Οι εικόνες δεν καλύπτονται από νέφη.
- Το σύστημά αναφοράς τους είναι διαφορετικό αλλά και οι δύο ανάχθηκαν στο WGS 84.
- Στην εικόνα του 2007 που δεν είχε υποστεί ορθοαναγωγή, αυτή πραγματοποιήθηκε, προκειμένου να αποκτήσει την απεικόνιση χάρτη και να αφαιρεθούν από αυτή οι γεωμετρικές παραμορφώσεις που οφείλονται σε παραμέτρους όπως το τοπογραφικό ανάγλυφο ή τα συστηματικά σφάλματα των δορυφόρων.

Στην συνέχεια και αφού οι εικόνες διορθώθηκαν από όλα τα απαραίτητα σφάλματα και ανάχθηκαν στα ίδια συστήματα πληρούσαν όλες τις απαραίτητες προϋποθέσεις για την επεξεργασία τους με τους αλγορίθμους ανίχνευσης μεταβολών. Σύμφωνα με τα προηγούμενα, οι αλγόριθμοι στο ΟΤΒ λειτουργούν με μονοκάναλες εικόνες σε png τύπο. Έτσι με τον αλγόριθμο που αναφέρθηκε σε προηγούμενο κεφάλαιο, πραγματοποιήθηκε η εξαγωγή του κόκκινου καναλιού (κανάλι 3) και η αποθήκευσή του σε png format. Με αυτό τον τρόπο τα δεδομένα ήταν έτοιμα για χρήση από τους αντίστοιχους αλγορίθμους.

6.3 Αλγόριθμοι ανίχνευσης μεταβολών

6.3.1 Αλγόριθμος διαφοράς

Ένας από τους πιο διαδεδομένους αλγορίθμους ανίχνευσης μεταβολών είναι αυτός της διαφοράς εικόνων. Όμως ο συγκεκριμένος αλγόριθμος προκειμένου να μειώσει το θόρυβο στην εικόνα που παράγεται, αντί να πραγματοποιήσει την διαφορά της τιμής κάθε εικονοστοιχείου της μιας εικόνας με το αντίστοιχο, πραγματοποιεί την διαφορά των τοπικών μέσων όρων μιας περιοχής κάθε εικόνας. Με τον τρόπο αυτό οι μεγαλύτερες διαφορές αφαιρούνται και εμφανίζονται στην τελική εικόνα. Όπως έχει προαναφερθεί, ο αλγόριθμος παίρνει ως είσοδο δύο εικόνες μονοκάναλες σε png format. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

$$I_r(i,j) = I_2(i,j) - I_1(i,j)$$

Εκτενέστερα, ακολουθεί η ανάλυση του αλγορίθμου που δημιουργεί ο χρήστης για την σωστή εκτέλεση του κώδικα.

- Όπως κάθε φορά ορίστηκαν όλα τα αρχεία επικεφαλίδες που συνδέονταν και περιείχαν τους αλγορίθμους του κώδικα. Στην συγκεκριμένη περίπτωση εκτός από όλα τα γνωστά αρχεία επικεφαλίδων που χρησιμοποιήθηκαν σε κάθε εκτελέσιμο αρχείο, ορίστηκε και το αρχείο που περιείχε τον αλγόριθμο της διαφοράς (otbMeanDifferenceImageFilter.h). (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 26-32)
- Στην συνέχεια, έγινε έλεγχος των στοιχείων που εισήχθηκαν στο αρχείο. Ο χρήστης εκτός από τις δύο εικόνες, έπρεπε να εισάγει την εικόνα που παράγεται, αλλά και την ακτίνα με την οποία υπολογίστηκε η περιοχή στην οποία πραγματοποιήθηκε ο μέσος όρος. Ο αλγόριθμος πραγματοποίησε την διαφορά μέσων όρων στις δύο εικόνες για να εξαλείψει τον θόρυβο.(παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 37-43)

- Ακολούθησε ο ορισμός όλων των τύπων μεταβλητών που χρειάστηκαν για την εκτέλεση του κώδικα. Εκτός από τις δύο εικόνες που ορίστηκαν από την κλάση otb::Image, ορίστηκαν και οι τύποι της εικόνας που προέκυψαν από τον αλγόριθμο της διαφοράς ChangeImageType, αλλά και η εικόνα που παρουσιάστηκε τελικώς στον χρήστη OutputImageType. Τέλος, ορίστηκαν οι τύποι των δεικτών αναγνωστών και των γραφέων. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 45-57)

- Στην συνέχεια ορίστηκαν οι δύο διαδικασίες που έγιναν προκειμένου η τελική εικόνα να παρουσιαστεί στον χρήστη. Ορίστηκε δηλαδή, ο δείκτης που προερχόταν από την κλάση itk::AbsImageFilter και έπαιρνε την απόλυτη τιμή του αποτελέσματος. Ο αλγόριθμος της διαφοράς πραγματοποίησε την διαφορά χωρίς να ελέγξει αν το αποτέλεσμα της αλγεβρικής αυτής πράξης είναι αρνητικό ή θετικό. Με την απόλυτη τιμή το αποτέλεσμα χρησιμοποιήθηκε χωρίς πρόσημο. Επίσης ορίστηκε και η κλάση itk::RescaleIntensityImageFilter, η οποία πραγματοποίησε ενίσχυση του ιστογράμματος μετασχηματίζοντας τις τιμές της έντασης που είχαν προκύψει έτσι ώστε να πάρουν όλο το εύρος των τιμών (0-255) και το αποτέλεσμα να είναι πιο ευκρινές. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 58-59)

- Έπειτα, το πρότυπο του αλγορίθμου της διαφοράς πήρε ως όρισμα τις εικόνες που εισήχθηκαν καθώς και το όνομα της εικόνας που παράχθηκε. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 61)

- Στο σημείο αυτό, δημιουργήθηκαν δυναμικά όλοι οι απαραίτητοι δείκτες που έλαβαν τις αντίστοιχες τιμές, ενώ ορίστηκαν και οι δείκτες που έλαβαν τα ορίσματα των δεδομένων που εισήχθηκαν. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 63-76)

- Ακολούθησε ο ορισμός των ορίων που πραγματοποίησε η ενίσχυση του ιστογράμματος. Τα όρια αυτά ήταν οι ακραίες τιμές δηλαδή από το 0 εώς το 255. Επίσης, ορίστηκε και η ακτίνα που χρησιμοποιήθηκε το φίλτρο για να οριστεί γειτονιά εικονοστοιχείων που προσδιορίστηκε ο μέσος όρος.(παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 77-78, 80)

- Κατά τα γνωστά ορίστηκαν οι ροές που εκτελέστηκαν. Το σημείο που παρουσιάζει ενδιαφέρον είναι αυτό στο οποίο εμφανίστηκε με γραφικό τρόπο η διαδικασία μεταγλώττισης του κώδικα. Το γεγονός ότι οι εικόνες που προορίζονταν για τους συγκεκριμένους αλγορίθμους ήταν αρκετά μεγάλες και απαιτούσαν αρκετό χρόνο για επεξεργασία, οδήγησε στην γραφική απεικόνιση μέσω ενός προτύπου της διαδικασίας εκτέλεσής του. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 82-90)

- Τέλος, ελέγχθηκε με την μέθοδο try, catch αν ο δείκτης που περιείχε το αποτέλεσμα μπόρεσε να γίνει update και να λάβει την τιμή του. Σε περίπτωση που αυτή η ενέργεια για οποιοδήποτε λόγο δεν πραγματοποιηθεί, εμφανίζεται ένδειξη λάθος στον χρήστη. (παράρτημα, 6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων, 92-101)

Στο Σχήμα 6.2 εμφανίστηκε το αποτέλεσμα της εκτέλεσης του αλγορίθμου. Όλες οι περιοχές που δεν είχαν υποστεί κάποια αλλαγή παρουσιάζονταν με μαύρο χρώμα καθώς η διαφορά των τιμών τους ήταν 0, ενώ αντίθετα οι περιοχές που παρουσίαζαν μεγάλη αλλαγή εμφανίζονταν με άσπρο χρώμα καθώς η διαφορά των τιμών τους έδωσε ένα μεγάλο νούμερο που με την ενίσχυση του ιστογράμματος έφτασε τις τιμές κοντά στο 255. Από την προαναφερθείσα εικόνα φάνηκε πως η περιοχή της Αττικής Οδού παρουσιάστηκε μα φωτεινούς τόνους, καθώς ο αλγόριθμος λειτούργησε με το σωστό τρόπο και αναγνώρισε την διαφορά που υπήρχε στην περιοχή. Βέβαια με φωτεινούς τόνους παρουσιάστηκαν και άλλες περιοχές

Σχήμα 6.2: Αποτέλεσμα του αλγορίθμου διαφοράς για τις εικόνες Ikonos στο μπλέ κανάλι

της εικόνας, καθώς οι αλλαγές εντοπίστηκαν και εκεί. Στο σημείο αυτό κρίνεται σκόπιμο να τονιστεί πως σε περίπτωση που ο χρήστης ήθελε να δώσει μεγαλύτερη ένταση στις περιοχές που δεν μεταβλήθηκαν τότε θα πραγματοποιούσε μια κατωφλίωση με ανώτατο όριο στην τιμή 128 καθώς τότε όλα τα εικονοστοιχεία θα αναγκάζονταν να λάβουν τιμές σε αυτή την περιοχή, ενώ αν ήθελε να δώσει έμφαση στις περιοχές που παρουσιάζουν μεταβολή θα πραγματοποιούσε κατωφλίωση με κατώτερο όριο το 128 έτσι ώστε οι αλλαγές να ενισχύονταν στους φωτεινούς τόνους.

6.3.2 Αλγόριθμος λόγου

Ο συγκεκριμένος αλγόριθμος λειτουργεί με παρόμοιο τρόπο όπως αυτός της διαφοράς. Αντί όμως για την αλγεβρική διαφορά, πραγματοποιεί τον λόγο των δύο εικόνων. Για τους λόγους που αναφέρθηκαν παραπάνω, χρησιμοποίησε τον μέσο όρο μιας γειτονιάς εικονοστοιχείων στην εικόνα και ο λόγος πραγματοποιήθηκε σε αυτούς. Στην συνέχεια το αποτέλεσμα κανονικοποιηθηκε μεταξύ 0 και 1. Έτσι οι αλλαγές θα παρουσιάζονταν με άσπρες αποχρώσεις ενώ τα αμετάβλητα στοιχεία με μαύρες (στον τύπο που φαίνεται και παρακάτω υπάρχει μια αφαίρεση). Ο αλγόριθμος αυτός παρήγαγε πολύ καλά αποτελέσματα και σε δεδομένα ραντάρ καθώς ο λόγος εξαλείφει τον συστηματικό θόρυβο. Το μοντέλο που χρησιμοποιήθηκε από τον αλγόριθμο είναι το ακόλουθο: [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

$$I_r(i,j) = 1 - \min(\frac{I_2(i,j)}{I_1(i,j)}, \frac{I_1(i,j)}{I_2(i,j)})$$

όπου:

Ir(i,j)= η τιμή του εικονοστοιχείου στην τελική εικόνα [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

Αναλυτικότερα τώρα, τα κύρια σημεία του αλγορίθμου ήταν τα ακόλουθα:

- Ορισμός όλων των αρχείων επικεφαλίδων με κύριο αυτό του λόγου (otbMeanRatioImage Filter.h). Όλα τα άλλα αρχεία ήταν παρόμοια όπως σε κάθε εκτελέσιμο αρχείο. (παράρτημα, 6.2 Αρχείο Κώδικα για αλγόριθμο λόγου εικόνων, 22-27)

- Στην συνέχεια, πραγματοποιήθηκε ο έλεγχος των στοιχείων που εισήχθησαν στο πρόγραμμα. Ο χρήστης έπρεπε πάλι να εισάγει την ακτίνα στην οποία πραγματοποιήθηκε ο αλγόριθμος του μέσου όρου από τις αρχικές εικόνες. Έπεται ο ορισμός όλων των μεταβλητών που χρησιμοποιήθηκαν από τον κώδικα. (παράρτημα, 6.2 Αρχείο Κώδικα για αλγόριθμο λόγου εικόνων, 29-72)

- Το σημείο που πρέπει να τονιστεί ήταν η δημιουργία του RescalerType όπου λάμβανε ως όρισμα στο πρότυπό του την εικόνα που προήλθε από τον αλγόριθμο και την εικόνα που δόθηκε στον χρήστη ως αποτέλεσμα. Στην συνέχεια, έλαβε και τα ορίσματά τους το πρότυπο του αλγορίθμου του λόγου. Ως είσοδο ορίστηκαν οι δύο αρχικές εικόνες και η εικόνα που προέκυψε από την επεξεργασία του αλγορίθμου. (παράρτημα, 6.2 Αρχείο Κώδικα για αλγόριθμο λόγου εικόνων, 73-74)

- Ακολούθησε η δημιουργία των δεικτών που έλαβαν τα δεδομένα και τα αποτελέσματα, η εισαγωγή των τιμών στους αντίστοιχους δείκτες, ο ορισμός της ακτίνας, η δημιουργία των ροών, η οπτικοποίηση της εξέλιξης της μεταγλώττισης και ο έλεγχος για την σωστή εισαγωγή του αποτελέσματος. (παράρτημα, 6.2 Αρχείο Κώδικα για αλγόριθμο λόγου εικόνων, 76-96)

Σχήμα 6.3: Αποτέλεσμα της χρήσης του αλγορίθμου λόγου για τις εικόνες Ikonos στο μπλέ κανάλι

Σχολιάζοντας το παραπάνω αποτέλεσμα, (Σχήμα 6.3), ο χρήστης μπορεί να παρατηρήσει πως οι μεταβολές των δύο εικόνων προέκυψαν με πολύ καλά αποτελέσματα. Η περιοχή της Αττικής Οδού παρουσιάστηκε με φωτεινούς τόνους και έγινε εμφανές πως πρόκειται για μια αλλαγή. Οι περιοχές που δεν παρουσίασαν κάποια αλλαγή εμφανίστηκαν σκούρες, με μαύρες αποχρώσεις, καθώς πήραν τιμές έντασης πολύ κοντά στο 0. Πολύ σημαντικό σημείο στον αλγόριθμο αυτό αποτέλεσε το γεγονός ότι παρήγαγε πολύ καλά αποτελέσματα και για δεδομένα ραντάρ.

6.3.3 Αλγόριθμος τοπικής συσχέτισης

Όπως αναφέρθηκε παραπάνω, ο συγκεκριμένος αλγόριθμος μετράει την πιθανότητα να υπάρχει μια γραμμική σχέση μεταξύ δύο τυχαίων αριθμών. Ο αλγόριθμος αφού υπολογίσει τον μέσο όρο για μια περιοχή εικονοστοιχείων που ορίζει ο χρήστης, υπολογίζει την απόκλιση των τιμών της περιοχής από την μέση τιμή και για τις δύο εικόνες (διασπορά), τέλος παράγει το

αποτέλεσμα που προκύπτει από το άθροισμα του γινομένου της διαφοράς της μέσης τιμής από την τιμή στην συγκεκριμένη θέση για τις δύο εικόνες προς το γινόμενο των διασπορών για την συγκεκριμένη γειτονία επί το μέγεθος της εικόνας. [The ORFEO Tool Box Software Guide Updated for OTB-3.10, June 30, 2011]

$$I_r(i,j) = 1 - (\sum_{(I_1(i,j),I_2(i,j))} \frac{(I_1(i,j) - m1)(I_2(i,j) - m2)}{s_1 * s_2 * n})$$

όπου:

Ir(i,j)= η τιμή του εικονοστοιχείου στην τελική εικόνα

m1, m2 = ο μέσος όρος των δύο εικόνων στην γειτονιά που ορίζει ο χρήστης

s1, s2 = η διασπορά στην γειτονία που ορίζει ο χρήστης

n = το μέγεθος της εικόνας Από τον παραπάνω αλγόριθμο προέκυψε πως όλες οι περιοχές

που παρουσίαζαν αλλαγές εμφανίστηκαν με άσπρο χρώμα, ενώ οι υπόλοιπες παρουσιάστηκαν με φωτεινές αποχρώσεις. Στο σημείο αυτό αναλύθηκε το εκτελέσιμο πρόγραμμα που δημιουργήθηκε για το συγκεκριμένο κώδικα και υπακούει σε όλα αυτά που αναλύθηκαν στους προηγούμενους αλγορίθμους και ακολουθεί τον τρόπο δόμησης του αλγορίθμου λόγων.

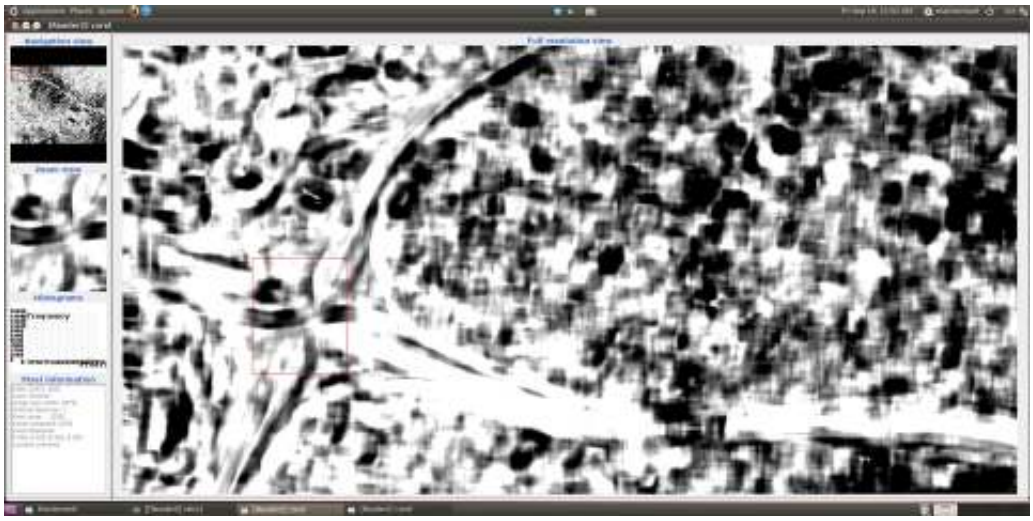
- Ορίστηκε το αρχείο επικεφαλίδα που περιείχε τον κώδικα του αλγορίθμου. Το αρχείο αυτό ήταν το otbCorrelationChangeDetector.h. Όλα τα άλλα αρχεία επικεφαλίδων παραμένουν ακριβώς τα ίδια. (παράρτημα, 6.3 Αρχείο Κώδικα για αλγόριθμο διασποράς εικόνων, 29)

- Ακολούθησε ο έλεγχος των δεδομένων εισόδου και ο ορισμός όλων των απαραίτητων μεταβλητών. Οι μεταβλητές αυτές ήταν οι ίδιες όπως σε όλα τα εκτελέσιμα αρχεία για ανίχνευση μεταβολών (παράρτημα, 6.3 Αρχείο Κώδικα για αλγόριθμο διασποράς εικόνων, 37-56)

- Στην συνέχεια, έγινε η κανονικοποίησης του ιστογράμματος γραμμικά μεταξύ των τιμών 0 και 1 έτσι ώστε το αποτέλεσμα να λάβει όλο το εύρος των τιμών του ιστογράμματος. Αυτό επιτεύχθηκε μέσω της κλάσης itk::ShiftScaleImageFilter. (παράρτημα, 6.3 Αρχείο Κώδικα για αλγόριθμο διασποράς εικόνων, 58)

- Αφού ορίστηκε το φίλτρο, παίρνοντας ως τιμές στο πρότυπό του, τις δύο εικόνες που εισήγαγε ο χρήστης και την εικόνα που παράχθηκε, ακολουθήθηκε η δημιουργία όλων των δεικτών που έλαβαν τα δεδομένα, το πέρασμα των τιμών στους δείκτες αυτούς και η δημιουργία όλων των ροών για την εκτέλεση του αλγορίθμου. Όλα αυτά ολοκληρώθηκαν με τον έλεγχο της σωστής εισόδου τιμής στον writer. (παράρτημα, 6.3 Αρχείο Κώδικα για αλγόριθμο διασποράς εικόνων, 61-101)

Το αποτέλεσμα που προέκυψε από τον αλγόριθμο δεν ήταν εύκολο στην παρατήρηση (Σχήμα 6.4). Αρχικά παρατηρήθηκε πως δεν υπήρχε μεγάλη διάκριση μεταξύ των μεγάλων και των μικρών αλλαγών στην εικόνα. Ο αλγόριθμος δεν μπόρεσε να ξεχωρίσει τις περιοχές που είχαν μια μικρή μεταβολή ώστε να παρουσιαστούν με σκούρους τόνους, αλλά οποιαδήποτε μικρή αλλαγή εμφανίστηκε με φωτεινούς τόνους. Έστω και μια μικρή αλλαγή σε μια γειτονιά εικονοστοιχείων που όρισε ο χρήστης, μεταφράστηκε σε φωτεινούς τόνους σε όλη την αντίστοιχη γειτονία. Επίσης κάποια μικρά σφάλματα που υπήρχαν στην γεωμετρία της εικόνας, μεταφράστηκαν από τον αλγόριθμο ως αλλαγές και δημιούργησαν περισσότερο θόρυβο στο αποτέλεσμα. Το πρόβλημα αυτό μπόρεσε να αντιμετωπιστεί με σωστή ενίσχυση του ιστογράμματος.



Σχήμα 6.4: Αποτέλεσμα της χρήσης του αλγορίθμου τοπικής συσχέτισης για τις εικόνες Ikonos στο μπλέ κανάλι

6.3.4 Αλγόριθμος LHMI

Ο αλγόριθμος LHMI (Localized Harmonic Motion Imaging) αποτελεί έναν από τους παρεχόμενους αλγορίθμους από το περιβάλλον του OTB για ανίχνευση μεταβολών. Ο αλγόριθμος λαμβάνει τις δύο εικόνες, υπολογίζει το δισδιάστατο ιστόγραμμά τους (στον έναν άξονα βρίσκεται η μια εικόνα και στον άλλο η άλλη) και από εκεί υπολογίζει την εντροπία ως προς κάθε άξονα για κάθε υποπεριοχή που ορίζει ο χρήστης με την ακτίνα που ορίζει. Ως εντροπία ορίζεται η μεταβλητή που υπολογίζεται από το άθροισμα της συχνότητας παρατήρησης της τυχαίας τιμής και του λογαρίθμου της. Στην συνέχεια υπολογίζει την μεταβλητή της συσχετισμένης εντροπίας που προκύπτει από το δισδιάστατο ιστόγραμμα και το τελικό αποτέλεσμα προκύπτει από τον τύπο:

$$\mathrm{Ir(i,j)}=\frac{jointEntropy}{(entropyX+entropyY)}$$

όπου:

Ir(i,j)= η τιμή του εικονοστοιχείου στην τελική εικόνα

jointEntropy = η εντροπία που προκύπτει από το συσχετισμένο ιστόγραμμα

entropyX, entropyY = η εντροπία ανά άξονα

Το εκτελέσιμο αρχείο του συγκεκριμένου αλγορίθμου παρουσιάστηκε στο παράρτημα (παράρτημα, 6.4 Αρχείο Κώδικα για αλγόριθμο LHMI εικόνων). Είναι δομημένο με τον ίδιο τρόπο όπως το εκτελέσιμο αρχείο του λόγου των εικόνων και της διασποράς οπότε δεν παρουσιάζεται κάποια ανάλυση. Το μόνο που κρίνεται σκόπιμο να αναφερθεί είναι πως το αρχείο επικεφαλίδα που χρησιμοποιήθηκε και για την συγκεκριμένη περίπτωση ήταν το otbLHMIChangeDetector.h οπότε και το φίλτρο είναι το ανάλογο.

Από το παρακάτω σχήμα παρατηρούνται τα εξής: (Σχήμα 6.5)

- Οι πολύ μεγάλες αλλαγές που προέκυψαν από τον αλγόριθμο παρουσιάστηκαν με άσπρο χρώμα (σαν καμμένο). Έτσι στις περιπτώσεις που αρχικά δεν υπήρχε κάποιο κτίριο, κάποια γέφυρά ή γενικότερα κάποιο αντικείμενο, αυτό παρουσιάστηκε με πολύ άσπρο χρώμα.

- Οι μικρότερες αλλαγές εμφανίστηκαν με ανοιχτές γκρι αποχρώσεις. Έτσι στην περίπτωση της Αττικής Οδού που προϋπήρχε η σχεδίαση της και στις δύο εικόνες, η αλλαγή που εντοπίστηκε από τον αλγόριθμο παρουσιάστηκε με γκρι χρώμα. Παρατηρήθηκε λοιπόν πως ο συγκεκριμένος αλγόριθμος δεν παρήγαγε πολύ καλά αποτελέσματα για τις συγκεκριμένες εικόνες και την συγκεκριμένη εφαρμογή.
- Οι περιοχές που δεν παρουσίασαν κάποια αλλαγή παρουσιάστηκαν με μαύρο χρώμα.

Σχήμα 6.5: Αποτέλεσμα της χρήσης του αλγορίθμου LHMI για τις εικόνες Ikonos στο μπλέ κανάλι

6.3.5 Αλγόριθμος CBAMI

Η μέθοδος CBAMI (Cumulant based approximation to Mutual Information) είναι μια ανθρωποκεντρική μέθοδος εντοπισμού αλλαγών. Πρόκειται για ένα εννοιολογικό πλαίσιο που περιγράφει, εξηγεί και προβλέπει πιθανές αλλαγές σε συνήθειες και συμπεριφορές. Λαμβάνει υπόψη τις ανησυχίες των ερωτώμενων και με βάση αυτές καταλήγει σε συμπεράσματα. Η εφαρμογή της στην ψηφιακή επεξεργασία εικόνων και ιδιαίτερα στις εφαρμογές ανίχνευσης μεταβολών έγκειται στην εφαρμογή φίλτρων και προσαρμογών, στις ψηφιακές τιμές, ώστε στην συνέχεια με μια εφαρμογή διαφοράς να αναδειχθούν οι πραγματικές «ουσιώδεις» αλλαγές.[http://www.nationalacademies.org/rise/backg4a.htm, 2011].

Το εκτελέσιμο αρχείο που χρησιμοποιήθηκε για την υλοποίηση του κώδικα ήταν δομημένο με τον ίδιο τρόπο όπως και στις πιο πάνω περιπτώσεις. (παράρτημα, 6.5 Αρχείο Κώδικα για αλγόριθμο CBAMI εικόνων). Στον συγκεκριμένο όμως αλγόριθμο, δεν χρησιμοποιήθηκε καθόλου ενίσχυση καθώς, με την χρήση του rescaler που εφαρμόστηκε και στους άλλους αλγορίθμους, το αποτέλεσμα αλλοιωνόταν. Επίσης, χρησιμοποιήθηκαν τα αντίστοιχα αρχεία επικεφαλίδες και το φίλτρο που πραγματοποίησε τον αλγόριθμο του CBAMI. Το βασικό πρόβλημα που αντιμετωπίστηκε ήταν ότι ο αλγόριθμος χρησιμοποίησε όλους τους πόρους του συστήματος, καθ΄όλη την διάρκεια που γινόταν η μεταγλώττιση.

Από το παρακάνω σχήμα, (Σχήμα 6.6) ο χρήστης δεν μπόρεσε να εξάγει πολλά συμπεράσματα από την παρατήρηση του. Αρχικά, τονίστηκαν αλλαγές που οφείλονταν κυρίως από γεωμετρικά σφάλματα της εικόνας, καθώς παρουσιάστηκαν με άσπρο (καμμένο) χρώμα. Επίσης οι λίγο πιο μικρές αλλαγές όπως η ασφαλτόστρωση της Αττικής Οδού δεν ανιχνεύονται με πολύ

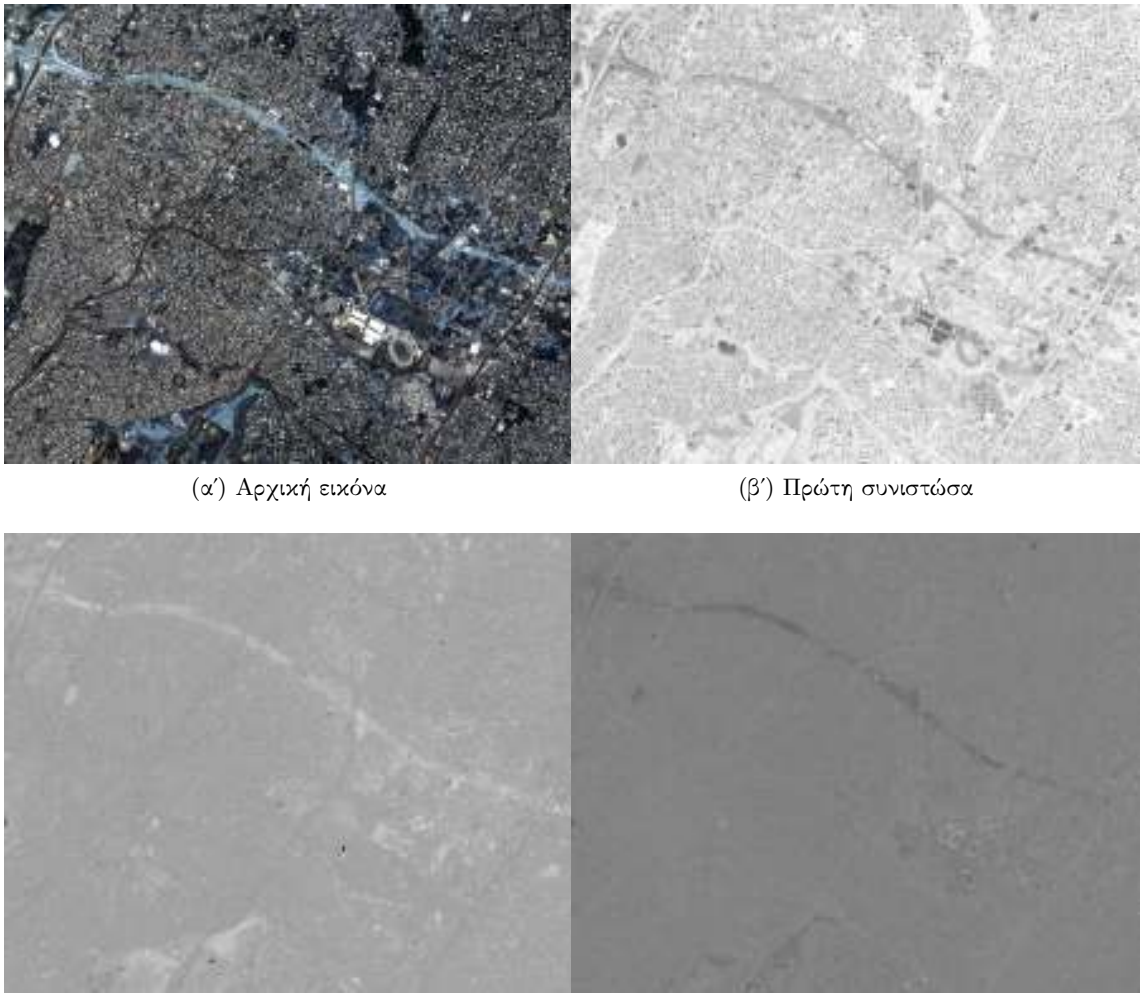
καλό τρόπο. Όλες οι περιοχές που δεν παρουσιάζουν κάποια αλλαγή, εμφανίζονται με σκούρο μαύρο χρώμα. Γενικότερα, ο αλγόριθμος δεν ενδείκνυται για ανίχνευση μικρών αλλαγών, αλλά μπορεί να χρησιμοποιηθεί με καλύτερα αποτελέσματα για την ανίχνευση μεγάλων αλλαγών.

Σχήμα 6.6: Αποτέλεσμα της χρήσης του αλγορίθμου CBAMI για τις εικόνες Ikonos στο μπλε κανάλι

6.3.6 Αποτελέσματα από εφαρμογή αλγορίθμων στις δύο πρώτες συνιστώσες PCA

Τα προηγούμενα αποτελέσματα προέκυψαν όπως αναφέρθηκε και παραπάνω από επεξεργασία του κόκκινου καναλιού όλης της εικόνας. Εκτός από εξαγωγή ενός καναλιού, ο χρήστης θα μπορούσε να εφαρμόσει τους παραπάνω αλγορίθμους διαδοχικά στις συνιστώσες της εικόνας όπως προκύπτουν από την μέθοδο του PCA. Επίσης, θα μπορούσε να μετασχηματίσει τον χώρο της εικόνας από RGB σε HSI και να εφαρμόσει τους αλγορίθμους στην συνιστώσα της έντασης. Στο συγκεκριμένο κεφάλαιο, παράχθηκαν από την εικόνα οι κύριες συνιστώσες της και οι αλγόριθμοι εφαρμόστηκαν στις δύο πρώτες.

Οι κύριες συνιστώσες παράχθηκαν με βάση το εκτελέσιμο αρχείο που παρουσιάστηκε στο παράρτημα, (5.4 Δημιουργία κύριων συνιστωσών). Έτσι από την εικόνα του 2000 της αττικής οδού προέκυψαν τα παρακάτω αποτελέσματα: (Σχήμα 6.7)

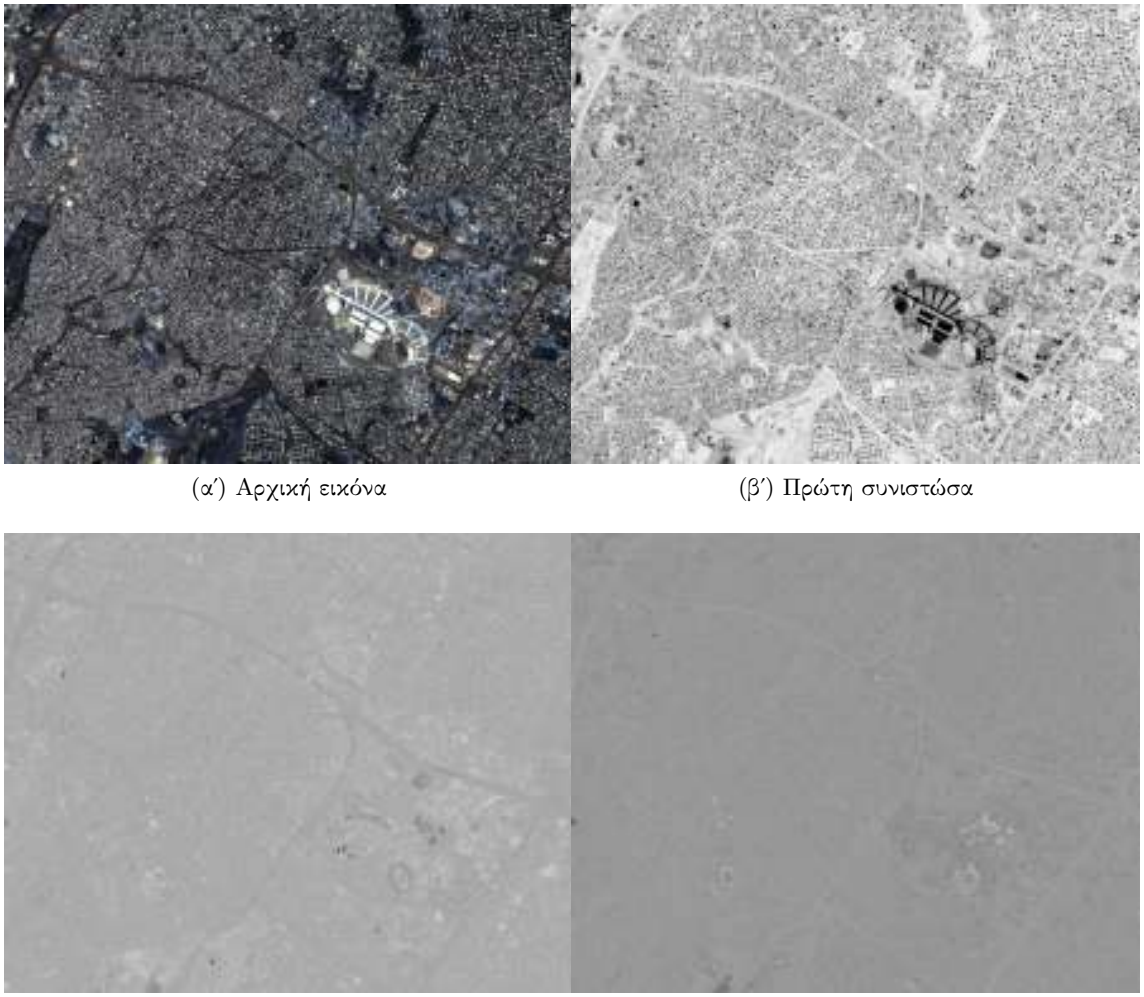


(α′) Αρχική εικόνα (β′) Πρώτη συνιστώσα

(γ′) Δεύτερη συνιστώσα (δ′) Τρίτη συνιστώσα

Σχήμα 6.7: Κύριες συνιστώσες της εικόνας του έτους 2000

Η ίδια μέθοδος εφαρμόστηκε και για την εικόνα του 2007 της ίδιας περιοχής. Έτσι και στην περίπτωση αυτή προέκυψαν τα εξής αποτελέσματα: (Σχήμα 6.8)



(α′) Αρχική εικόνα (β′) Πρώτη συνιστώσα

(γ′) Δεύτερη συνιστώσα (δ′) Τρίτη συνιστώσα

Σχήμα 6.8: Κύριες συνιστώσες της εικόνας του έτους 2007

Από τα παραπάνω αποτελέσματα, (Σχήματα 6.7, 6.8) εφαρμόστηκαν οι αλγόριθμοι του κεφαλαίου στις δύο πρώτες συνιστώσες. Όπως αναφέρθηκε και παραπάνω, οι δύο πρώτες συνιστώσες ήταν αυτές που περιείχαν την περισσότερη πληροφορία από την εικόνα, οπότε όποιες αλλαγές θα φαίνονταν σε αυτές. Κυρίως μάλιστα η πρώτη συνιστώσα ήταν αυτή που περιείχε το μεγαλύτερο μέρος της πληροφορίας. Ακολουθούν λοιπόν τα αποτελέσματα όπως προέκυψαν από τους αλγορίθμους.

(α′) Διαφορά καναλιών στη πρώτη συνιστώσα (β′) Διαφορά καναλιών στη δεύτερη συνιστώσα

Σχήμα 6.9: Αποτελέσματα αλγορίθμου διαφοράς από την χρήση αποτελεσμάτων PCA

(α′) Λόγος εικόνων στη πρώτη συνιστώσα (β′) Λόγος εικόνων στη δεύτερη συνιστώσα

Σχήμα 6.10: Αποτελέσματα αλγορίθμου λόγου από την χρήση αποτελεσμάτων PCA

(α′) Τοπική διασπορά στη πρώτη συνιστώσα (β′) Τοπική διασπορά στη δεύτερη συνιστώσα

Σχήμα 6.11: Αποτελέσματα αλγορίθμου τοπικής συσχέτισης από την χρήση αποτελεσμάτων PCA

(α′) Αλγόριθμος LHMI στη πρώτη συνιστώσα (β′) Αλγόριθμος LHMI στη δεύτερη συνιστώσα

Σχήμα 6.12: Αποτελέσματα αλγορίθμου LHMI από την χρήση αποτελεσμάτων PCA

(α′) Αλγόριθμος CBAMI στη πρώτη συνιστώσα (β′) Αλγόριθμος CBAMI στη δεύτερη συνιστώσα

Σχήμα 6.13: Αποτελέσματα αλγορίθμου CBAMI από την χρήση αποτελεσμάτων PCA

Από τα παραπάνω αποτελέσματα, (Σχήματα 6.9, 6.10, 6.11, 6.12, 6.13) φαίνεται πως οι εφαρμογές ανίχνευσης μεταβολών στην συγκεκριμένη εφαρμογή, μέσω PCA δεν παρουσίασαν τόσο καλά αποτελέσματα. Το γεγονός αυτό οφείλεται στο ότι οι δύο πρώτες συνιστώσες περιείχαν μέρος της αρχικής πληροφορίας της εικόνας. Οι μεταβολές όμως που υπήρχαν στην εικόνα αποτελούσαν ένα πολύ μικρό κομμάτι της, οπότε ήταν πολύ πιθανό να μην απεικονίζονται σε κάποια από τις δύο συνιστώσες. Έτσι και αλλιώς η μεταβολή που ανιχνεύτηκε στην συγκεκριμένη εφαρμογή αναφέρθηκε σε αλλαγή στο υλικό εδάφους κατά μήκος της Αττικής Οδού, μια μεταβολή δηλαδή που δεν ήταν τόσο μεγάλη ούτε εμφανής. Για το λόγο αυτό, η εξαγωγή ενός καναλιού της εικόνας που περιείχε όλη την πληροφορία της στο συγκεκριμένο κανάλι, και η εφαρμογή των αλγορίθμων ανίχνευσης μεταβολών σε αυτό, παρήγαγε πολύ καλύτερα αποτελέσματα.

Κεφάλαιο 7

Προγραμματιστική Εφαρμογή

7.1 Προγραμματισμός OTB στην C++

Η βιβλιοθήκη του OTB είναι οργανωμένη σύμφωνα με τον **γενικευμένο προγραμματισμό** (generic programming) και κατ΄ επέκταση δομείται με βάση αυτόν. Ο γενικευμένος προγραμματισμός είναι μια μέθοδο οργάνωσης βιβλιοθηκών που αποτελούνται από γενικά στοιχεία κώδικα. Η βασική ιδέα είναι η δημιουργία κώδικα που είναι ικανός να δομεί προγράμματα με έναν αποτελεσματικό και εύκολο στην προσαρμογή τρόπο. Πλεονάζουσας σημασίας του γενικευμένου προγραμματισμού είναι οι **περιέκτες** (containers) τα οποία ΄κρατάνε΄ τα δεδομένα, οι **επαναλήπτες** (iterators) προκειμένου να αποκτά πρόσβαση στα δεδομένα και οι γενικευμένοι αλγόριθμοι που περιέχουν τα παραπάνω στοιχεία και δημιουργούν αποτελεσματικούς, θεμελιώδης αλγορίθμους. Ο γενικευμένος προγραμματισμός πραγματοποιείται στην C++ μέσω του προγραμματισμού σε πρότυπα templates και χρησιμοποιεί την STD(Standart Template Library) βιβλιοθήκη.

Τεχνικές **γενικευμένου προγραμματισμού** χρησιμοποιούνται και στην ITK. Το πλεονέκτημα της συγκεκριμένης μεθόδου είναι το γεγονός ότι σχεδόν όλοι οι τύποι δεδομένων υποστηρίζονται, απλά δηλώνοντας τον κατάλληλο τύπο στο αντίστοιχο πρότυπο. Για παράδειγμα, στο OTB ο χρήστης μπορεί να δημιουργήσει εικόνες που αποτελούνται με όλους σχεδόν τους τύπους εικονοστοιχείων. Αντίθετα, ο τύπος ανάλυσης προσδιορίζεται κατά την διάρκεια της μεταγλώττισης, έτσι ο μεταγλωττιστής μπορεί να αποδώσει τον κώδικα κατά το καλύτερο τρόπο.

Εκτός όμως από τον γενικευμένο προγραμματισμό, το OTB αποτελείται και από άλλα σημαντικά στοιχεία. Ένα από αυτά είναι η **συμπερίληψη** αρχείων και ο **ορισμός κλάσεων**. Στην ITK και στο OTB ο ορισμός κλάσεων γίνεται με το μέγιστο δύο αρχεία. Ένα αρχείο επικεφαλίδα (header .h) και ένα εκτελέσιμο αρχείο (.cxx) σε περίπτωση που δεν πρόκειται για κλάση πρότυπο και (.txx) σε περίπτωση που είναι κλάση πρότυπο. Τα αρχεία επικεφαλίδας χρησιμοποιούνται για ορισμούς μεταβλητών και κλάσεων. Στην συνέχεια, ένα ακόμα σημαντικό δομικό στοιχείο του OTB αποτελούν τα **χωρικά αντικείμενα** (spatial objects). Χρησιμοποιώντας ένα κοινό βασικό περιβάλλον, τα χωρικά αντικείμενα είναι ικανά να αναπαραστήσουν περιοχές του χώρου με μια ποικιλία τρόπων. Το συγκεκριμένο εργαλείο έχει πολύ μεγάλη χρήση στους αλγορίθμους που πραγματοποιούν κατάτμηση εικόνας.

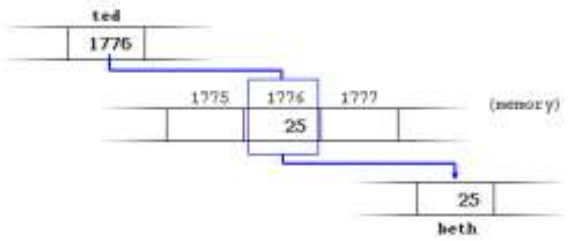
Επόμενο πολύ σημαντικό στοιχείο του OTB είναι οι **δείκτες** και μάλιστα οι smart pointers. Οι δείκτες χρησιμοποιούνται για διαχείριση μνήμης και θα αναλυθούν με περισσότερη λεπτομέρεια στην συνέχεια. Έπειτα, το OTB χρησιμοποιεί την μέθοδο των **εξαιρέσεων** (exceptions). Με την συγκεκριμένη μέθοδο γίνεται με αξιόπιστο και βολικό τρόπο η μεταβίβαση των πληροφοριών ενός σφάλματος, από το αρχικό σημείο εντοπισμού προς ένα σημείο

όπου είναι διαθέσιμες αρκετές πληροφορίες για την ανάκαμψη του προβλήματος. Στην συνέχεια, ένα ακόμα δομικό στοιχείο του OTB, αποτελεί, η δημιουργία μιας αλληλουχίας γεγονότων(pipeline) που ολοκληρώνονται με την μέθοδο εκτέλεσης (Execute()). Το στοιχείο αυτό ευθύνεται για την σωστή εκτέλεση όλων των ροών διοχέτευσης ενεργειών στους αλγορίθμους. Τέλος, το OTB χρησιμοποιεί την τεχνική **πολυμορφισμού** (multi-threading), η οποία, κατά την διάρκεια που εκτελείται ο αλγόριθμος, κατακερματίζει την εικόνα σε διάφορες περιοχές προκειμένου να εφαρμόσει τον αλγόριθμο διαδοχικά σε όλες. Με τον τρόπο αυτό αποφεύγονται πολλά λάθη και η διαδικασία εκτέλεσης του αλγορίθμου γίνεται πολύ πιο γρήγορη σε επεξεργασία.

Στο σημείο αυτό κρίνεται σκόπιμο να αναλυθούν κάποια βασικά στοιχεία που χρησιμοποιούνται σε όλους τους αλγορίθμους του OTB και η πλήρης κατανόησή τους.

7.1.1 Δείκτες

Για έναν τύπο T, ο T* είναι ο τύπος ΄δείκτης σε T΄. Με άλλα λόγια, μια μεταβλητή τύπου T* μπορεί να περιέχει τη διεύθυνση ενός αντικειμένου του τύπου T. [Stroustrup, 2003]



Σχήμα 7.1: Παράδειγμα δεικτών.[Soulie Juan, 2007]

Από το παραπάνω σχήμα (Σχήμα 7.1) προκύπτει πως στην περίπτωση που υπάρχει μια μεταβλητή έστω andy=25 και βρίσκεται στην θέση μνήμης 1776, τότε beth = andy ενώ andy= *ted.

Οι δείκτες έχουν πολύ μεγάλη εφαρμογή και χρήση σε διαχείριση πινάκων, καθώς ο χρήστης δεν χρειάζεται να φορτώνει στην μνήμη του συστήματός του όλο τον πίνακα, αλλά απλά την θέση στην οποία βρίσκεται το πρώτο του στοιχείο. Έτσι στην περίπτωση του OTB που όλες οι εικόνες μεταφράζονται σε πίνακες πολλών διαστάσεων οι δείκτες είναι απαραίτητοι. Από την φύση του ο αντικειμενοστραφής προγραμματισμός αντιπροσωπεύει και λειτουργεί με δεδομένα που προέρχονται μέσα από μια ποικιλία τύπων αντικειμένων και κλάσεων. Όταν μια κλάση λοιπόν παράγει κάποιο δεδομένο, η διανομή της μνήμης γίνεται έτσι ώστε να μπορέσει να αποθηκεύσει και να διαχειριστεί τα δεδομένα που παράγονται μέσω δεικτών.

Στο περιβάλλον του OTB οι δείκτες χρησιμοποιούνται με την βοήθεια της· κλάσης itk::Smart Pointer. Οι ΄έξυπνοι΄ δείκτες έχουν ακριβώς την ίδια χρήση με αυτή των απλών δεικτών, με την μόνη διαφορά πως δεν μπορούν να σβηστούν από την μνήμη, αν δεν έχει σβηστεί πρώτα το αντικείμενο που δείχνουν. Με αυτό τον τρόπο μπορούν να αποφευχθούν πολλά λάθη κατά την διάρκεια του προγραμματισμού. Οι ΄έξυπνοι΄ δείκτες επίσης, μέσω της συγκεκριμένης κλάσης, μπορούν να κατανεμηθούν κατά την διάρκεια του προγράμματος και επίσης μπορούν να σβηστούν αυτόματα όταν κλείνει η περιοχή στην οποία αυτοί έχουν δημιουργηθεί. Όλες οι λειτουργίες του OTB γίνονται με χρήση δεικτών.

7.1.2 Επαναλήπτες (iterators)

Οι επαναλήπτες είναι η συνθετική ύλη που ενώνει τους περιέχοντες τύπους με τους αλγορίθμους. Παρέχουν μια αφηρημένη θεώρηση των δεδομένων, ώστε κάποιος που γράφει έναν αλγόριθμο να μη χρειάζεται να ασχολείται με τις συγκεκριμένες λεπτομέρειες χιλιάδων δομών δεδομένων. Αντίστροφα, το καθιερωμένο μοντέλο προσπέλασης δεδομένων που παρέχεται από τους επαναλήπτες απαλλάσσει τους παρέχοντες τύπους από την υποχρέωση να διαθέτουν ένα πιο εκτεταμένο σύνολο πράξεων προσπέλασης [Stroustrup, 2003].

Ο επαναλήπτης είναι μια αφαίρεση της ιδέας του δείκτη σε στοιχείο ακολουθίας. Οι βασικές του έννοιες είναι:

- Το τρέχον στοιχείο (αναφέρεται με τους τελεστές * και ->όπως οι δείκτες)
- Μετάβαση στο επόμενο στοιχείο (βηματική αύξηση, συμβολίζεται με τον τελεστή ++)
- Ισότητα (συμβολίζεται με τον τελεστή ==)

Οι επαναλήπτες είναι πολύ σημαντικοί στην χρήση του OTB καθώς προσφέρουν τον ποιο γρήγορο και αποδοτικό τρόπο, έτσι ώστε ο χρήστης να αποκτήσει πρόσβαση στα εικονοστοιχεία της εικόνας. Για το λόγο αυτό υπάρχουν μια ποικιλία κλάσεων που περιέχουν διάφορους τύπους επαναληπτών έτσι, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο. Για όλες τις επεξεργασίες εικόνας (φίλτρα, πράξεις καναλιών, ανίχνευση μεταβολών κ.τ.λ.) και γενικότερα όλες τις λειτουργίες που γίνονται με εικονοστοιχεία, χρησιμοποιούνται οι επαναλήπτες.

7.1.3 Κλάσεις και αντικείμενα

Οι κλάσεις αποτελούν μια προέκταση των δομών δεδομένων, αλλά αντί να αποθηκεύουν δεδομένα, μπορούν να αποθηκεύουν και συναρτήσεις. Τα περιεχόμενα της κλάσης ανάλογα με την πρόσβαση που αφήνουν στον χρήση χωρίζονται σε τρεις κατηγορίες:

- Ιδιωτικά μέλη (private) : τα συγκεκριμένα μέλη γίνονται προσβάσιμα μόνο από από μέλη της ίδιας κλάσης ή από ΄φίλους΄ της.
- Προστετευμένα μέλη (protected) : τα μέλη αυτά είναι προσβάσιμα μόνο από μέλη της ίδιας της κλάσης και ΄φίλων΄ της, αλλά παράλληλα και από παράγωγες κλάσεις τους.
- Κοινά μέλη (public): τα μέλη αυτά είναι προσβάσιμα από οπουδήποτε μέσα στο πρόγραμμα.

Όλες οι δομές που χρησιμοποιούνται στο περιβάλλον του OTB οργανώνονται με κλάσεις. Η κάθε κλάση μπορεί να αποτελείται από μεταβλητές, συναρτήσεις ή ακόμα και άλλες υποκλάσεις.

Η C++ μέσω των δομών της δημιουργεί αντικείμενα. Τα αντικείμενα είναι μια συνεχής περιοχή μνήμης. Τα αντικείμενα μπορούν να παραχθούν με πολλούς τρόπους. Μερικά από αυτά είναι τοπικές μεταβλητές, μερικά καθολικές μεταβλητές, μερικά αποτελούν μέλη άλλων κλάσεων κ.τ.λ. Τα αντικείμενα είναι δομικό μέλος του προγραμματισμού σε C++.

7.1.4 Πρότυπα

Τα πρότυπα (templates) παρέχουν έναν απλό τρόπο για να αναπαριστάται ένα ευρύ φάσμα γενικών εννοιών, και απλούς τρόπους για να συνδυαστούν οι έννοιες αυτές. Οι κλάσεις και οι

συναρτήσεις που προκύπτουν μπορούν να φτάσουν τον πιο εξειδικευμένο χειροποίητο κώδικα σε αποδοτικότητα χρόνου εκτέλεσης και χώρου μνήμης. [Stroustrup, 2003]

Τα πρότυπα παρέχουν μια έμμεση υποστήριξη για το γενικευμένο προγραμματισμό, για το λόγο αυτό χρησιμοποιούνται τόσο πολύ από το OTB. Επίσης όπως και οι κλάσεις υποστηρίζουν τον πολυμορφισμό, την ύπαρξη δηλαδή εικονικών (virtual) μελών αλλά και ο χειρισμός των αντικειμένων να γίνεται μέσω δεικτών και αναφορών. Σε αντίθεση με τις κλάσεις όμως, ο πολυμορφισμός στα πρότυπα πραγματοποιείται κατά την διάρκεια της μεταγλώττισης κάτι που είναι καλύτερο για αρκετούς λόγους. Στο περιβάλλον του OTB επίσης, τα πρότυπα ορίζονται με κλάσεις. Είτε όμως χρησιμοποιούνται τύποι typename αντί κλάσεις, το αποτέλεσμα θα ήταν το ίδιο.

7.1.5 Ιδιότητες δομών

7.1.5.1 Κληρονομικότητα

Όλος ο κώδικας του OTB λειτουργεί με βάση την κληρονομικότητα. Η κληρονομικότητα είναι αυτή που αφήνει την δημιουργία κλάσεων παραγώγων από κλάσεων βάσης. Με αυτό τον τρόπο όλες οι μεταβλητές και οι συναρτήσεις που υπάρχουν στην κλάση βάσης 'κληρονομούνται' στην κλάση παράγωγο. Έτσι δεν χρειάζεται να οριστούν πάλι μεταβλητές ή λειτουργίες που έχουν οριστεί ήδη. Τα περισσότερα πρότυπα του OTB κληρονομούν ιδιότητες και λειτουργίες από πρότυπα της ITK.

7.1.5.2 Πολυμορφισμός

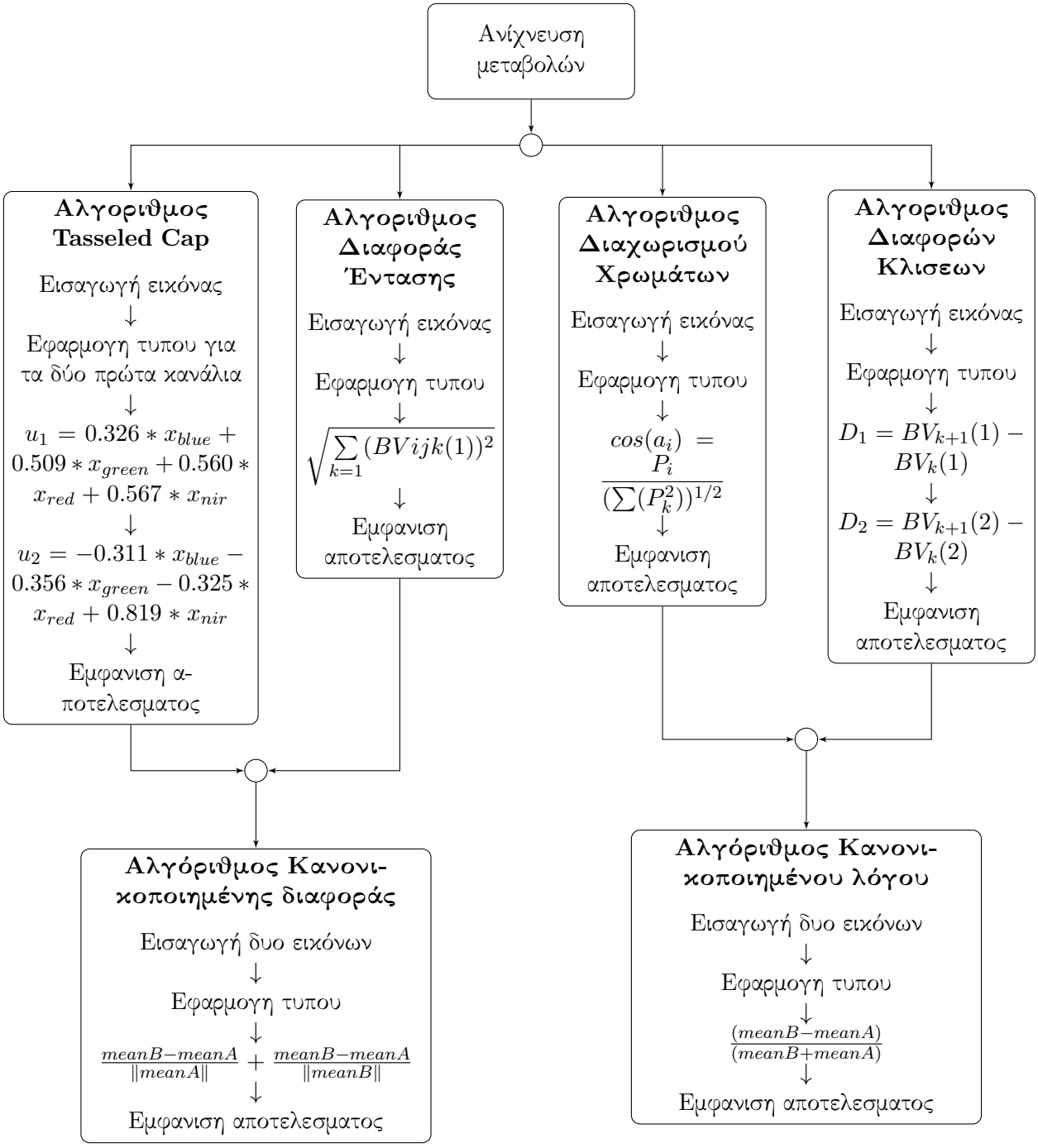
Ο πολυμορφισμός, όπως και η κληρονομικότητα ευνοούν τον γενικευμένο προγραμματισμό. Ιδιαίτερα, ο πολυμορφισμός επιτρέπει την δημιουργία προγραμμάτων που επεξεργάζονται αντικείμενα κλάσεων που είναι μέρος της ίδιας ιεραρχίας κλάσεων, όπως π.χ. αν αυτά τα αντικείμενα είναι αντικείμενα της κλάσης βάσης της ιεραρχίας.[Deitel, 2003]

Για να επιτευχθεί πολυμορφική συμπεριφορά στην C++, οι συναρτήσεις μέλη που καλούνται πρέπει να είναι εικονικές (virtual), και ο χειρισμός των αντικειμένων πρέπει να γίνεται με δείκτες ή αναφορές. Όταν ένα αντικείμενο χειρίζεται άμεσα (και όχι μέσω δείκτη ή αναφοράς), τότε ο ακριβής τύπος του είναι γνωστός στο μεταγλωττιστή και έτσι δε χρειάζεται η χρήση πολυμορφισμού κατά την εκτέλεση. Κάτι τέτοιο όμως απαιτεί μεγάλα αποθέματα μνήμης, και δημιουργεί προγράμματα που δεν είναι πολύ αποδοτικά.

7.2 Εφαρμογές ανίχνευσης μεταβολών

7.2.1 Διαδικασία σχεδιασμού αλγορίθμων

Πριν αρχίσει οποιαδήποτε αναφορά και ανάλυση στους αλγορίθμους που δημιουργήθηκαν, δημιουργήθηκε ένα διάγραμμα ροής, το οποίο απεικονίζει την όλη διαδικασία. Όπως παρουσιάστηκε λοιπόν και από το Σχήμα 7.2, η διαδικασία ανίχνευσης μεταβολών έγινε σε δύο ξεχωριστά στάδια. Στο πρώτο στάδιο, εφαρμόστηκε ένας αλγόριθμος μετασχηματισμού εικόνας, στις δύο εικόνες, ενώ σε επόμενο στάδιο, σχεδιάστηκαν οι αλγόριθμοι κανονικοποιημένου λόγου και διαφοράς που πραγματοποιούν την τελική ανίχνευση μεταβολών. Οι αντίστοιχοι αλγόριθμοι, παρουσιάστηκαν στα αντίστοιχα κεφάλαια που ακολουθούν.



Σχήμα 7.2: Διάγραμμα ροής διαδικασίας που ακολουθείται

7.3 Εφαρμογές ανίχνευσης μεταβολών

7.3.1 Αλγόριθμος Διαφοράς Έντασης

7.3.1.1 Θεωρία

Στο σημείο αυτό, κρίθηκε σκόπιμο να αναλυθεί ο τρόπος με τον οποίο έχουν δομηθεί οι επιμέρους αλγόριθμοι που σχεδιάστηκαν για την εκτέλεση της συγκεκριμένης εφαρμογής. Αρχικά, ο αλγόριθμος χωρίστηκε σε δύο στάδια. Το πρώτο αποτέλεσε η επεξεργασία της αρχικής εικόνας με χρήση του αλγορίθμου διαφοράς έντασης (όπως παρουσιάζεται και στο διάγραμμα ροής (Σχήμα 7.3)) και η παραγωγή μιας μονοκάναλης εικόνας που τα εικονοστοιχεία της έχουν ως τιμές το αποτέλεσμα της εφαρμογής του αλγορίθμου. Στην συνέχεια, οι δύο εικόνες που παράχθηκαν, χρησιμοποιήθηκαν στο πρόγραμμα change detection όπου πραγματοποιήθηκε η κανονικοποιημένη διαφορά τους.

Η διαφορά έντασης στην τιμή φωτεινότητας για κάθε εικονοστοιχείο υπολογίζεται λαμβάνοντας υπ΄όψιν όλα τα κανάλια της εικόνας, σύμφωνα με τον τύπο:

$$\mathrm{M(i,j)(1)}=\sqrt{\sum_{k=1}(BVijk(1))^2}$$

όπου:

M(i,j)(1) = η τιμή έντασης του εικονοστοιχείου (i,j) στην εικόνα 1
BVijk(1) = η τιμή φωτεινότητας στην εικόνα 1
i = ο αριθμός της γραμμής
j = ο αριθμός της στήλης
k = ο αριθμός του καναλιού
n = συνολικός αριθμός καναλιών

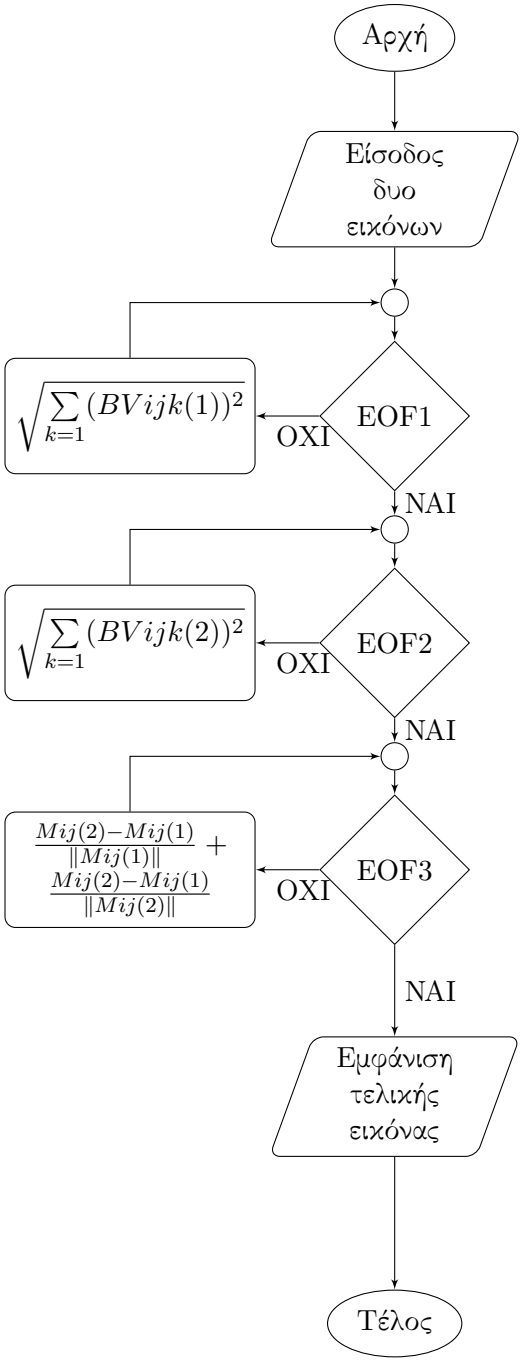
Στην συνέχεια, υπολογίζεται η τιμή έντασης Mij(2) της εικόνας 2, οπότε η σχετική διαφορά θα είναι:

$$\frac{Mij_2 - Mij_1}{\|Mij_1\|} + \frac{Mij_2 - Mij_1}{\|Mij_2\|}$$

Με τον τρόπο αυτό παρέχεται ένα μέτρο της μεταβολής μεταξύ όλων των καναλιών των εικόνων. Μια τέτοια προσέγγιση χρησιμοποιείται στις περιπτώσεις όπου λαμβάνει χώρα αλλαγή της φωτεινότητας των φατνίων σε όλα τα φασματικά κανάλια, όπως για παράδειγμα εάν μετακινηθεί ένα όχημα ανοιχτού χρώματος από ένα ασφαλτοστρωμένο υπαίθριο σταθμό αυτοκινήτων. [Imagine Delta Cue Users Guide, 2007]

7.3.1.2 Προγραμματισμός

Στο σημείο αυτό και πριν αναφερθεί ο κώδικας που δημιουργήθηκε για την υλοποίηση του αλγορίθμου, κρίνεται σκόπιμο να παρουσιαστεί το διάγραμμα ροής του αλγορίθμου με βάση το οποίο έγινε όλη η σχεδίασή του. (Σχημα 7.3)



Σχήμα 7.3: Διάγραμμα ροής αλγορίθμου Διαφοράς Ένταση

Δημιουργήθηκε λοιπόν, ένα αρχείο .cxx το οποίο με χρήση επαναληπτών κατάφερνε να αποσπάσει από όλη την εικόνα την τιμή κάθε εικονοστοιχείου για όλα τα κανάλια της, και στην συνέχεια εφάρμοσε τον αντίστοιχο τύπο. Αναλυτικότερα ο κώδικας του αρχείου παρουσιάζεται παρακάτω:

- Αρχικά ορίστηκαν όλα τα αρχεία επικεφαλίδες που χρειάζονταν για τον αλγόριθμο. Η εικόνα που εισήχθηκε είναι πολυκάναλη, για το λόγο αυτό χρησιμοποιήθηκε και το otb VectorImage.h, ενώ η εικόνα που παράχθηκε έχει μόνο ένα κανάλι και για το λόγο αυτό χρησιμοποιήθηκε το otbImage.h. Επίσης, χρησιμοποιήθηκαν τα αρχεία επικεφαλίδες για τους επαναλήπτες ανάλογα με τον τύπο της εικόνας που απευθύνονται. Τέλος, ορίστηκε και το αρχείο επικεφαλίδα math.h, καθώς εφαρμόστηκαν μαθηματικοί τύποι για την υλοποίηση του αλγορίθμου. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 18-25)

- Στην συνέχεια, ελέγχθηκε ο αριθμός των στοιχείων που εισήχθηκαν στο αρχείο. Στην περίπτωση αυτή εισήχθηκαν δύο ορίσματα, η αρχική εικόνα που περιείχε τις πληροφορίες, και αυτή που παράχθηκε.(Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 34-39)

- Έπειτα, ορίστηκαν όλοι οι τύποι δεδομένων που χρειάστηκε ο αλγόριθμος. Έτσι ορίστηκαν οι διαστάσεις των εικόνων, ο τύπος των εικονοστοιχείων, οι τύποι των εικόνων, του αναγνώστη (reader) και των επαναληπτών. Στο σημείο αυτό, κρίνεται σκόπιμο να αναφερθεί πως οι επαναλήπτες δεν ήταν του ίδιου τύπου. Η εικόνα που έχει πολλά κανάλια, χρησιμοποίησε τον σταθερό επαναλήπτη δηλαδή τον itkImageRegionConstIterator. Ο τύπος αυτός επαναλήπτη, έχει πολύ μεγάλη ταχύτητα και είναι συνήθως η πρώτη επιλογή για εφαρμογές που το μόνο που χρειάζεται ο χρήστης είναι να διαβάσει την εικόνα. Με τον συγκεκριμένο τύπο, ο χρήστης μπορούσε να ορίσει και μια μικρότερη περιοχή της εικόνας για επεξεργασία σε περίπτωση που το ήθελε. Αντίθετα, η μονοκάναλη εικόνα που παράχθηκε, χρησιμοποίησε το αρχείο itkImageRegionIterator.h το οποίο αποτελεί ένα αρχείο βάσης για κάποιους άλλους τύπους επαναληπτών και επιτρέπει τόσο την ανάγνωση, όσο και την εγγραφή των εικονοστοιχείων της εικόνας. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 42-56)

- Ακολούθησε, η δημιουργία του αναγνώστη, το πέρασμα του ορίσματος της εικόνας που εισήχθηκε και ο έλεγχος για σωστή εκτέλεση της διοχέτευσης ροής. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 59-72)

- Επόμενο βήμα, αποτέλεσε η δημιουργία του επαναλήπτη της εικόνας που παράχθηκε και η δημιουργία μιας εικόνας με μηδενικά, έτσι ώστε τα εικονοστοιχεία της να έχουν μια αρχική τιμή. Παρατηρήθηκε πως οι διαστάσεις της τελικής εικόνας είναι ίδιες με αυτές της αρχικής, και στον επαναλήπτη πέρασε σαν όρισμα όλη η εικόνα. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 75-79)

- Έπειτα, ορίστηκε ο επαναλήπτης της εικόνας που εισήχθηκε, σε αυτή την περίπτωση ορίστηκε η εικόνα στο σύνολό της και στην συνέχεια εξάχθηκε ο αριθμός των καναλιών της εικόνας. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφορά έντασης .cxx, 83-85)

- Προκειμένου να λειτουργήσει σωστά η επαναληπτική διαδικασία που λάμβανε όλα τα εικονοστοιχεία της εικόνας, οι δύο επαναλήπτες ορίστηκαν από την αρχή της εικόνας που προσδιορίζουν. Αυτό γίνεται με την εντολή GoToBegin() η οποία αποτελεί μέλος του αντίστοιχου τύπου επαναλήπτη. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 89-90)

- Από το σημείο αυτό, άρχισε όλη η διαδικασία υπολογισμού των εικονοστοιχείων της παραγόμενης εικόνας. Για όλα τα εικονοστοιχεία και για όλα τα κανάλια, πραγματοποιήθηκε το άθροισμα του γινομένου των στοιχείων και στην συνέχεια ο επαναλήπτης της παραγόμενης εικόνας πήρε την τιμή της ρίζας αυτού του αποτελέσματος. Κρίνεται σκόπιμο να αναφερθεί, πως τα εικονοστοιχεία της εικόνα που παράχθηκε αρχικά ορίστηκαν ως διπλής ακρίβειας (double) αφού χρησιμοποιήθηκαν για πράξεις, και στην συνέχεια με την μέθοδο static cast ορίστηκαν ως PixelType. Έτσι και αλλιώς το PixelType ήταν τύπου διπλής ακρίβειας οπότε δεν υπήρχε κάποιο πρόβλημα ασυμβατότητας τύπων. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 100-109)

- Τέλος, ορίστηκε ο τύπος του γραφέα (writer), δημιουργήθηκε η ροή ώστε να παραχθεί η τελική εικόνας και ελέγχθηκε κατά πόσο η ροή εκτελέστηκε με το σωστό τρόπο. (Παράρτημα, 7.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς έντασης .cxx, 113-128)

Το παραπάνω πρόγραμμα χρησιμοποιήθηκε και για τις δύο εικόνες ξεχωριστά και παρήγαγε δύο καινούργιες εικόνες οι οποίες εισήχθηκαν στο εκτελέσιμο αρχείο της διαφοράς έτσι όπως τροποποιήθηκε για τις ανάγκες του αλγορίθμου. Το εκτελέσιμο αρχείο που δημιουργήθηκε βρίσκεται στο κεφάλαιο (Παράρτημα, Εκτελέσιμο αρχείο για διαφορά εικόνων και ανίχνευση μεταβολών). Δεν κρίνεται σκόπιμο να αναλυθεί σε λεπτομέρεια, καθώς ακολούθησε τον τρόπο δόμησης της διαφοράς εικόνων που αναφέρθηκε στο κεφάλαιο 6. Απλά, άλλαξαν τα αρχεία επικεφαλίδες ώστε ο αλγόριθμος διαφοράς που χρησιμοποιήθηκε να είναι αυτός που δημιουργήθηκε για το σκοπό αυτό, ενώ δεν χρειάστηκε να εφαρμοστεί κάποιο είδος ενίσχυσης στο τελικό αποτέλεσμα.

Το κομμάτι που κρίνεται σκόπιμο να αναφερθεί, είναι τα αρχεία επικεφαλίδας που τροποποιήθηκαν και περιείχαν τον αλγόριθμο της ανίχνευσης μεταβολών. Έτσι λοιπόν, αναλύεται αρχικά το αρχείο otbNormalizedMeanDifferenceImageFilter.h που περιείχε όλον τον αλγόριθμο της διαφοράς.

- Στο συγκεκριμένο αρχείο ορίστηκαν όλες οι μεταβλητές που χρειάστηκε ο κώδικας για να υλοποιηθεί. Όλα αυτά έγιναν μέσω προτύπων και κλάσεων, ενώ όλα τα μέλη ορίστηκαν ως κοινά public), προστατευόμενα (protected) και ιδιωτικά (private). Επίσης, όλοι οι αλγόριθμοι της ανίχνευσης μεταβολών λειτούργησαν όπως αναφέρθηκε και στο κεφάλαιο 6, μέσω προτύπων που χρησιμοποιούν εκτελεστές (Functors) και ορίζονται σε ξεχωριστά αρχεία επικεφαλίδες.

- Αρχικά λοιπόν, ορίστηκαν τα αρχεία επικεφαλίδες που είναι απαραίτητα για όλες τις δηλώσεις. Το πρώτο αρχείο είναι αυτό που όρισε το πρότυπο με το οποίο εφαρμόστηκε ο αλγόριθμος της ανίχνευσης μεταβολών. Το συγκεκριμένο πρότυπο δίνει την δυνατότητα να εισαχθούν δύο εικόνες και να χρησιμοποιηθεί σε κάθε περίπτωση μια γειτονιά εικονοστοιχείων για την επεξεργασία της. Το δεύτερο αρχείο που ορίστηκε, ήταν αυτό στο οποίο δημιουργήθηκε ο συγκεκριμένος εκτελεστής. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 20-21)

- Εκτός από τον χώρο που βρισκόταν το αρχείο (δηλαδή το otb), ορίστηκε και το πρότυπό του με όλα τα ορίσματα. Το πρότυπο ήταν απαραίτητο να οριστεί προκειμένου στην συνέχεια η κλάση να πάρει τα σωστά ορίσματα. Παρατηρούμε πως οι εικόνες εισήχθηκαν ως κλάσεις κάτι που αποτελεί τρόπο δήλωσης προτύπων για το περιβάλλον του OTB. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 27-29)

- Στην συνέχεια, ορίστηκε η κλάση που δημιουργήθηκε και έχει ως δημόσια όλα τα μέλη της κλάσης παραγωγής του εκτελεστή. Η εντολή ITK EXPORT αποτελεί μια προδιαγραφή για να λειτουργήσει με το σωστό τρόπο η κλάση. Παρατηρείται επίσης, πως η

κλάση λάμβανε τις δυο εικόνες και την εικόνα που παράγεται, αλλά ορίζει παράλληλα και το εκτελεστή, χρησιμοποιώντας ως ορίσματα για τις εικόνες το ανάλογο τύπο επαναλήπτη. Στην συγκεκριμένη περίπτωση, ο επαναλήπτης που χρησιμοποιήθηκε, λάμβανε μια περιοχή της εικόνας προκειμένου να πραγματοποιήσει οποιαδήποτε πράξη πάνω σε αυτή. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 30-37)

- Στην συνέχεια, ακολούθησε η δήλωση των δημοσίων (public) μελών της κλάσης. Στο συγκεκριμένο σετ εντολών, δόθηκαν απλά ονόματα σε όλες τις κλάσεις και τις μεταβλητές που χρησιμοποιήθηκαν, ανάλογα με την χρήση τους. Έτσι η ίδια η κλάση ορίστηκε ως Self, η κλάση που πραγματοποίησε την όλη διαδικασία και όρισε το πρότυπο BinaryFunctorNeighborhoodImageFilter ορίστηκε ως υπερκλάση, ενώ όλοι οι δείκτες που δημιουργήθηκαν από την κλάση και έδειχναν στα αντικείμενά της ορίστηκαν ως σταθεροί ή μη σταθεροί. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 38-57)

- Ακολούθησε η δήλωση των προστατευμένων (protected) μελών της κλάσης. Αυτά δεν ήταν άλλα, από τον κατασκευαστή (constractor) και τον καταστροφέα (destractor) της κλάσης που κατασκευάστηκε. Με αυτό τον τρόπο, τα δύο αυτά μέλη δεν μπορούσαν να επεξεργαστούν έξω από την κλάση, αλλά διασφαλίστηκε η χρήση τους από φιλικές σε αυτή κλάσεις. Παρατηρήθηκε επίσης, πως ο καταστροφέας ορίστηκε ως εικονικός (virtual), κάτι που επιτρέπει την σωστή χρήση της κλάσης από άλλα προγράμματα, σύμφωνα με τους κανόνες της κληρονομικότητας των κλάσεων. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 59-61)

- Τέλος, ακολούθησε η δήλωση των ιδιωτικών (private) μελών του προτύπου τα οποία χρησιμοποιήθηκαν μόνο από αυτό. Τα μέλη αυτά δεν ήταν άλλα, από τον κατασκευαστή αντιγραφής της κλάσης και τον τελεστή = που πραγματοποιεί την ίδια δουλειά. Με αυτό τον τρόπο διασφαλίστηκε για ακόμα μια φορά πως η κλάση δεν μπορούσε να αντιγραφεί με κανένα τρόπο. (παράρτημα, 7.2 Αρχείο επικεφαλίδας για διαφορά εικόνων, 63-65)

Έπειτα, και αφού ορίστηκαν και δηλώθηκαν όλες οι κλάσεις και οι μεταβλητές, δημιουργήθηκε το αρχείο που περιείχε τον εκτελεστή (Functor) και όριζε την πράξη της ανίχνευσης μεταβολών που πραγματοποιήθηκε σε όλη την εικόνα. Το αρχείο αυτό αποτέλεσε ένα αρχείο επικεφαλίδας όπως και το προηγούμενο με το όνομα otbNomarlizedMeanDifference.h. Αναλυτικά το αρχείο περιείχε τα παρακάτω:

- Παρατηρήθηκε πως ο τρόπος δόμησης του συγκεκριμένου αρχείου ακολούθησε τον τρόπο δόμησης του προηγούμενου. Ορίστηκε ο χώρος που βρίσκεται, δηλαδή το otb και στην συνέχεια ορίστηκε ότι πρόκειται για τον εκτελεστή. Ακολούθησε ο ορισμός του προτύπου με τις τρεις εικόνες και στην συνέχεια, ορίστηκε η κλάση του δημιουργήθηκε. (παράρτημα, 7.4 Αρχείο επικεφαλίδας για διαφορά εικόνων Functor, 21-27)

- Ακολούθησε η δήλωση των κατασκευαστή και του καταστροφέα για την συγκεκριμένη κλάση. (παράρτημα, 7.4 Αρχείο επικεφαλίδας για διαφορά εικόνων Functor, 31-32)

- Στην συνέχεια, ορίστηκε η συνάρτηση που περιείχε όλη την διαδικασία. Η συνάρτηση αυτή, καλείτο από την BinaryFunctorNeighborhoodImageFilter, όπου και χρησιμοποιήθηκε έτσι και αλλιώς ο εκτελεστής. Στην συνάρτηση, έγινε ένας επαναπροσδιορισμός (overload) στον τελεστή παρένθεση. (παράρτημα, 7.4 Αρχείο επικεφαλίδας για διαφορά εικόνων Functor, 33)

- Έπειτα, ορίστηκαν οι τύποι των μεταβλητών που συνέπιπταν με τον τύπο της εικόνας που παράχθηκε και πραγματοποιήθηκε όλη η διαδικασία. Η επανάληψη που ορίστηκε

μέσα στην συνάρτηση έγινε για την περιοχή που όρισε ο χρήστης δίνοντας την ακτίνα στο εκτελέσιμο πρόγραμμα. (παράρτημα, 7.4 Αρχείο επικεφαλίδας για διαφορά εικόνων Functor, 35-44)

- Η συνάρτηση στο τέλος επέστρεψε το αποτέλεσμα που προέκυψε από την κανονικοποιημένη διαφορά των μέσων όρων που υπολογίστηκαν μέσα στην επανάληψη. Το αποτέλεσμα 'αναγκάστηκε' (μέσω του static cast) να γίνει τύπου TOutput αφού έτσι και αλλιώς αποτελεί τιμή του εικονοστοιχείου της νέας εικόνας.

- Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί πως ο εκτελεστής περιείχε μόνο την συνάρτηση που πραγματοποίησε την ζητούμενη πράξη των εικονοστοιχείων. Η περισσότερη δουλειά έγινε από το πρότυπο BinaryFunctorNeighborhoodImageFilter όπου και δημιουργήθηκαν οι επαναλήπτες των εικόνων και έγιναν όλες οι απαραίτητες δηλώσεις για να καλέσει το πρότυπο την συγκεκριμένη συνάρτηση για όλη την εικόνα.

Έχοντας γίνει όλες αυτές οι ενέργειες, ο χρήστης το μόνο που είχε να κάνει είναι να τρέξει το εκτελέσιμο αρχείο και να υπολογίσει τα τελικά αποτελέσματα.

7.3.1.3 Εκτέλεση - Αποτελέσματα

Στο σημείο αυτό παρουσιάστηκαν τα αποτελέσματα που προέκυψαν από κάθε εκτέλεση των επιμέρους προγραμμάτων. Αρχικά λοιπόν από την εκτέλεση του πρώτου αρχείου προέκυψαν οι εξής δύο εικόνες. (Σχήμα 7.4)

(α΄) Άθροισμα καναλιών για εικόνα 2000 (β΄) Άθροισμα καναλιών για εικόνα 2007

Σχήμα 7.4: Αποτελέσματα αλγορίθμου διαφοράς έντασης

Στην συνέχεια, οι δύο εικόνες εισήχθηκαν στο εκτελέσιμο αρχείο που περιέχει τον αλγόριθμο ανίχνευσης μεταβολών και έτσι παράχθηκε το παρακάτω αποτέλεσμα.

Από το Σχήμα 7.5 προέκυψε πως η ανίχνευση της μεταβολής της Αττικής Οδού έγινε με πολύ καλά αποτελέσματα. Στην εικόνα, οι περιοχές με μεταβολή παρουσιάστηκαν με άσπρες ή μαύρες αποχρώσεις, καθώς δεν χρησιμοποιήθηκε η απόλυτη τιμή της διαφοράς. Έτσι και αλλιώς σκοπός ήταν να μελετηθούν οι εικόνες και ως προς την διαφορά στις φασματικές υπογραφές τους. Στην περίπτωση της Αττικής Οδού λοιπόν, η αλλαγή παρουσιαστηκε με λευκές αποχρώσεις καθώς η φασματική υπογραφή της ασφάλτου ήταν μεγαλύτερη από αυτή του χωματουργικού υλικού. Με την συγκεκριμένη μέθοδο λοιπόν, ο χρήστης μπόρεσε εύκολα, γρήγορα και χωρίς να χρειαστεί υπερβολικούς υπολογιστικούς πόρους να προσδιορίσει τις μεταβολές από δύο εικόνες. Ο αλγόριθμος δεν ήταν πολύ πολύπλοκος, αλλά το αποτέλεσμά που προέκυψε, ήταν αρκετά καλό.

Σχήμα 7.5: Αποτέλεσμα αλγορίθμου κανονικοποιημένης διαφοράς στον αλγόριθμο διαφοράς έντασης

7.3.2 Αλγόριθμος Tasseled Cap

7.3.2.1 Θεωρία

Ο συγκεκριμένος αλγόριθμος που δημιουργήθηκε, αποτελείτο από δύο ξεχωριστά τμήματα. Το πρώτο τμήμα, υπολογίζει και εξάγει για κάθε εικόνα, δύο ξεχωριστές εικόνες. Η πρώτη περιείχε το πρώτο κανάλι όπως προέκυψε από τον μετασχηματισμό του Tasseled Cap και η δεύτερη το δεύτερο κανάλι. Στην συνέχεια, εφαρμόστηκε η κανονικοποιημένη διαφορά όπως υπολογίστηκε και στον παραπάνω αλγόριθμο.

Ο μετασχηματισμός Tasseled Cap συνίσταται στην εφαρμογή εξισώσεων γραμμικού μετασχηματισμού στα αρχικά δεδομένα, με βάση εμπειρικούς συντελεστές μετασχηματισμού που προσδιορίστηκαν για κάθε τύπο τηλεπισκοπικού αισθητήρα. Το διάνυσμα μετασχηματισμού για κάθε κανάλι δίνεται από τον τύπο [Horne, 2003]

$$u_i = R^T * x_i + r$$

όπου: u = το διάνυσμα των μετασχηματισμένων τιμών φωτεινότητας για το κανάλι i

R = ο πίνακας των συντελεστών μετασχηματισμού
x = το διάνυσμα των αρχικών τιμών φωτεινότητας για το κανάλι i
r = το διάνυσμα των σταθερών που επιλέγονται αυθαίρετα για την αποφυγή αρνητικών τιμών στο u

Οι νέες συνιστώσες την μετασχηματισμένης εικόνας ανταποκρίνονται σε φαινόμενα όπως η μεταβολή της φωτεινότητας του εδάφους και του πρασίνου. Αφού υπολογιστούν οι αντίστοιχες συνιστώσες σε κάθε εικόνα, στη συνέχεια εφαρμόζεται η σχετική διαφορά όπως και στον παραπάνω αλγόριθμο, για να παραχθεί έτσι η εικόνα μεταβολών μιας δεδομένης συνιστώσας, π.χ. θετική ή αρνητική μεταβολή στην εδαφική φωτεινότητα.

Για εικόνες Ikonos οπότε και με βάση τα αποτελέσματα του [Horne,2003], ο πίνακας των

συντελεστών είναι ο παρακάτω: (Σχήμα ;;)

$$\mathbf{R} = \begin{pmatrix} 0.326 & -0.311 & -0.612 & -0.650 \\ 0.509 & -0.356 & -0.312 & 0.719 \\ 0.560 & -0.325 & 0.722 & -0.243 \\ 0.567 & 0.819 & -0.081 & -0.031 \end{pmatrix}.$$

Σχήμα 7.6: Πίνακας συντελεστών για εικόνες Ikonos για τον αλγόριθμο Tasseled Cap

Έτσι λοιπόν, από ολόκληρη την εικόνα προκύπτει το πρώτο κανάλι του Tasseled Cap ως εξής:

$$u_1 = 0.326 * x_{blue} + 0.509 * x_{green} + 0.560 * x_{red} + 0.567 * x_{nir}$$

Αποτελεί δηλαδή ένα σταθμευμένο άθροισμα όλων των αρχικών καναλιών της εικόνας. Το αποτέλεσμα που προκύπτει έχει σχεδόν τα ίδια αποτελέσματα με αυτά που προκύπτουν από την παγχρωματική εικόνα. Λίγες είναι οι πληροφορίες που δεν υπάρχουν στο μετασχηματισμένο κανάλι.

Στη συνέχεια, εξάγεται και το δεύτερο κανάλι του Tasseled Cap μέσω στης σχέσης:

$$u_2 = -0.311 * x_{blue} - 0.356 * x_{green} - 0.325 * x_{red} + 0.819 * x_{nir}$$

Το συγκεκριμένο κανάλι, προκύπτει από την σταθμευμένη αφαίρεση όλων των ορατών καναλιών από αυτό του υπερύθρου. Το δεύτερο κανάλι του Tasseled Cap είναι πολύ χρήσιμο καθώς ξεχωρίζει τους διαφόρους τύπους εδάφους. Η βλάστηση τείνει να είναι μαύρη στο υπέρυθρο κανάλι, ενώ οι δρόμοι και τα κτίρια άσπρα. (Σχήμα 7.8, Σχήμα 7.9). Έτσι λοιπόν από τον συγκεκριμένο κανάλι προκύπτουν:

- Η βλάστηση παρουσιάζεται με φωτεινή ένταση.
- Τα αγροτεμάχια, οι βιομηχανικές περιοχές και γενικότερα όλες οι αντίστοιχες περιοχές παρουσιάζονται με μέτριας φωτεινότητας ένταση.
- Τα αστικά κέντρα, όπως δρόμοι, σπίτια, παρουσιάζονται με σκούρες αποχρώσεις.
- Το νερό παρουσιάζεται μαύρο.

Τέλος, τα άλλα δύο κανάλια παρουσιάζουν μικρό ποσοστό της πληροφορίας οπότε δεν θα χρησιμοποιηθούν στον συγκεκριμένο αλγόριθμο.

7.3.2.2 Προγραμματισμός

Πριν αναλυθεί σε λεπτομέρεια ο κώδικας που πραγματοποιεί ο αλγόριθμος, παρουσιάζεται σε ψευδογλώσσα παρακάτω: (Σχήμα 7.7)

```
Αλγόριθμος Tasseled Cap
    Έλεγχος σωστής εισαγωγής στοιχείων.
    Δήλωση των απαραίτητων τύπων μεταβλητών
    Δημιουργία του δείκτη reader
    reader → argv[1]
    Έλεγχος σωστής λειτουργίας της ροής
    Δημιουργία και αρχικοποίηση των δύο παραγόμενων εικόνων με μηδέν
    Δημιουργία και δήλωση των αντίστοιχων επαναληπτών για τις δύο εικόνες
    Όλοι οι επαναλήπτες πηγαίνουν στην αρχή της εικόνας
    Εξαγωγή των καναλιών της εικόνας μέσω της αντίστοιχης συνάρτησης (band)
    Όσο η αρχική εικόνα δεν είναι στο τέλος της επανέλαβε
        Δηλώνεται ο τύπος των εικονοστοιχείων για τις εικόνες που παράγονται
        Για i=0 μέχρι i=band με βήμα 1
            Άν (i=0) τότε
                pixelInput1 = 0.326*inputIt.Get()[i]
                pixelInput2 = -0.311*inputIt.Get()[i]
            Τέλος αν
            Άν (i=1) τότε
                 pixelInput1 += 0.509*inputIt.Get()[i]
                pixelInput2 += -0.356*inputIt.Get()[i]
            Τέλος αν
            Άν (i=2) τότε
                pixelInput1 += 0.560*inputIt.Get()[i]
                pixelInput2 += -0.325*inputIt.Get()[i]
            Τέλος αν
            Άν (i=3) τότε
                pixelInput1 += 0.567*inputIt.Get()[i]
                pixelInput2 += 0.819*inputIt.Get()[i]
            Τέλος αν
        Τέλος επανάληψης
        pixelOutput1 = static cast<PixelType>(pixelInput1)
        pixelOutput2 = static cast<PixelType>(pixelInput2)
        Οι επαναλήπτες πηγαίνουν στο επόμενο στοιχείο
        Αρχικοποίηση των απαραίτητων μεταβλητών
    Τέλος επανάληψης
    Δημιουργία δεικτών writer για τις δύο εικόνες
    writer → argv[2]
    writer → argv[3]
    Έλεγχος για την σωστή λειτουργίας της ροής των writer
Τέλος αλγορίθμου
```

Σχήμα 7.7: Διάγραμμα ροής για τον αλγόριθμο Tasseled Cap

Με βάση το Σχήμα 7.7, προέκυψε ότι έγινε ένας έλεγχος για το κανάλι και ανάλογα με το κανάλι η τιμή του εικονοστοιχείου πολλαπλασιάστηκε με τον ανάλογο συντελεστή. Στο τέλος πραγματοποιήθηκε το άθροισμα των τιμών αυτών και δημιουργήθηκαν τα δύο κανάλια του Tasseled Cap, τα οποία και περιείχαν την περισσότερη πληροφορία. Τα δύο αυτά κανάλια υπολογίστηκαν και για τις δύο εικόνες ξεχωριστά και στην συνέχεια εφαρμόστηκε η κανονικοποιημένη διαφορά τους σύμφωνα με τον αλγόριθμο που παρουσιάστηκε στην προηγούμενη παράγραφο.

Στο σημείο αυτό αναλύεται το εκτελέσιμο αρχείο που πραγματοποίησε την συγκεκριμένη εφαρμογή. Το αρχείο είχε πολλά κοινά με το προηγούμενο αρχείο της διαφοράς έντασης καθώς οι τύποι των εικόνων που επεξεργάστηκαν, ήταν οι ίδιοι.

- Αρχικά ορίστηκαν όλα τα αρχεία επικεφαλίδες που χρησιμοποιήθηκαν για την υλοποίηση των διαδικασιών στην συνέχεια. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 19-27)

- Ακολούθησε ο έλεγχος των ορισμάτων που έδινε ο χρήστης. Τα ορίσματα αποτελούνταν από τρεις εικόνες. Η πρώτη εικόνα ήταν αυτή που εισήχθηκε και περιείχε όλη την πληροφορία. Οι άλλες δύο εικόνες παράχθηκαν από το πρόγραμμα και αποτέλεσαν τα δύο πρώτα κανάλια του Tasseled Cap. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 36-41)

- Στην συνέχεια, ορίστηκαν όλοι οι τύποι των μεταβλητών που χρησιμοποιήθηκαν όπως συνήθως, καθώς επίσης και των επαναληπτών, ενώ έγινε έλεγχος για το αν ο αναγνώστης πήρε σωστά την τιμή του. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 44-73)

- Έπειτα, ορίστηκαν οι επαναλήπτες για τις δύο εικόνες που παράχθηκαν και δημιουργήθηκαν δύο εικόνες που είχαν ως αρχική τιμή, την τιμή 0.(παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 76-86)

- Ακολούθησε η δήλωση του επαναλήπτη για την εικόνα που εισήχθηκε, και η εξαγωγή των καναλιών της. Έπειτα όλοι οι επαναλήπτες αρχικοποιήθηκαν στην αρχή της περιοχής που ορίζουν προκειμένου να αρχίσει σωστά η επανάληψη. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 90-99)

- Για όλα τα εικονοστοιχεία λοιπόν της αρχικής εικόνας, ανάλογα με το κανάλι προέκυψε ένα γινόμενο της τιμής τους με έναν συντελεστή όπως προέκυψε από την έρευνα του Horne. Έτσι στις δύο εικόνες μπήκαν οι τιμές που προέκυψαν από την συγκεκριμένη διαδικασία. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 103-142)

- Τέλος, έγινε η μετάβαση των τιμών στους γραφείς και ο έλεγχος για την σωστή εκτέλεση της ροής. (παράρτημα, 7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap, 146-166)

Έτσι από κάθε εικόνα προέκυψαν δύο ξεχωριστές εικόνες. Οι εικόνες αυτές, αντίστοιχα με το κανάλι που αντιπροσώπευαν, επεξεργάστηκαν από τον NormalizedMeanDifferenceImageFilter. Για κάθε ένα λοιπόν προέκυψε το αντίστοιχο αποτέλεσμα ανίχνευσης μεταβολών.

7.3.2.3 Εκτέλεση - Αποτελέσματα

Αρχικά, από τις δύο εικόνες προέκυψαν τα παρακάτω αποτελέσματα από τον μετασχηματισμό. Στις εικόνες δεν πραγματοποιήθηκε καμία ενίσχυση για την εμφάνισή τους. (Σχήματα 7.8, 7.9)

(α′) Πρώτο κανάλι Tasseled Cap για εικόνα 2000 (β′) Δεύτερο κανάλι Tasseled Cap για εικόνα 2000

Σχήμα 7.8: Αποτελέσματα αλγορίθμου Tasseled Cap για το έτος 2000

(α′) Πρώτο κανάλι Tasseled Cap για εικόνα 2007 (β′) Δεύτερο κανάλι Tasseled Cap για εικόνα 2007

Σχήμα 7.9: Αποτελέσματα αλγορίθμου Tasseled Cap για το έτος 2007

Οι εικόνες που αναφέρονταν στο πρώτο κανάλι (Σχήματα 7.8α′, 7.9α′), παρουσίασαν μεγάλη διαφορά μεταξύ των δύο εποχών κατά μήκος της Αττικής Οδού. Παράλληλα στις εικόνες από το δεύτερο κανάλι, όπως προβλεπόταν και από την θεωρία, ξεχώρισαν οι διάφοροι τύποι εδάφους, όπου η βλάστηση εμφανίστηκε με φωτεινές αποχρώσεις, ενώ όλα τα αστικά με πιο σκούρες. Έτσι, στην περίπτωση της Αττικής Οδού, που στην μια εικόνα η περιοχή εμφανίστηκε με χώμα και στην άλλη με άσφαλτο, η διαφορά του εδάφους δεν είναι πολύ μεγάλη, οπότε και στις δύο περιπτώσεις παρατήθηκαν σκούροι τόνοι εικονοστοιχείων στην συγκεκριμένη περιοχή (Σχήματα 7.8β′, 7.9β′). Για το λόγο αυτό η διαφορά των δύο εικόνων από το δεύτερο κανάλι, δεν αναμένεται να δώσει καλά αποτελέσματα.

Έτσι η διαφορά των εικόνων έδωσε τα παρακάτω αποτελέσματα, για τα δύο κανάλια (Σχήμα 7.10). Για το πρώτο κανάλι, η μεταβολή στην Αττική Οδό εμφανίστηκε με πολύ καλό τρόπο (Σχήμα 7.10α′). Χωρίς να χρειαστεί να γίνει κάποια ενίσχυση στο αποτέλεσμα, οι περιοχές με μεταβολές παρουσιάστηκαν με λευκές και άσπρες αποχρώσεις, ενώ οι αμετάβλητες περιοχές παρουσιάστηκαν με μαύρες αποχρώσεις. Στην περίπτωση της Αττικής Οδού, οι μεταβολές γίνονται εμφανής με ανοιχτές αποχρώσεις.

Αντίθετα, το αποτέλεσμα που προέκυψε από το δεύτερο κανάλι δεν εμφάνισε κάποια σημαντικά συμπεράσματα (Σχήμα 7.10β′). Όπως αναφέρθηκε και παραπάνω, η διαφορά του χώματος και της ασφάλτου δεν ήταν πολύ μεγάλη και έτσι η διαφορά τους δεν έδωσε κάποιο σημαντικό αποτέλεσμα. Το συγκεκριμένο κανάλι θα έγινε πολύ καλά αποτελέσματα σε εικόνες που ο χρήστης ήθελε να ανιχνεύσει την επέκταση ενός αστικού περιβάλλοντος, ή μιας δασικής έκτασης.

(α′) Αποτέλεσμα αλγορίθμου διαφοράς Tasseled Cap για το πρώτο κανάλι

(β′) Αποτέλεσμα αλγορίθμου διαφοράς Tasseled Cap για το δεύτερο κανάλι

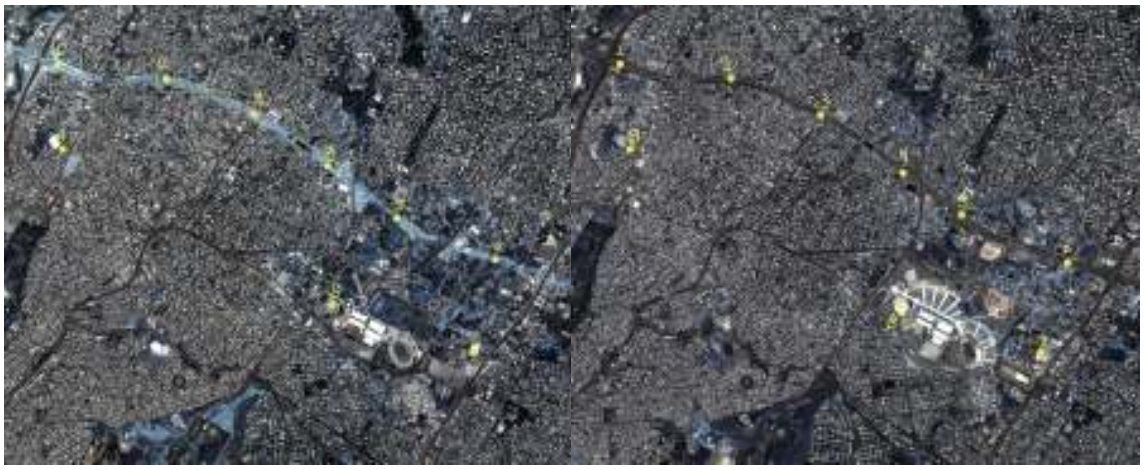
Σχήμα 7.10: Αποτελέσματα αλγορίθμου διαφοράς για την μέθοδο Tasseled Cap

7.3.3 Αλγόριθμος Διαχωρισμού Χρωμάτων (Color Difference)

7.3.3.1 Φασματικές υπογραφές

Στο σημείο αυτό, και πριν αναφερθεί οτιδήποτε για τους παρακάτω αλγορίθμους, κρίθηκε σκόπιμο να παρουσιαστούν και να συγκριθούν οι φασματικές υπογραφές από χαρακτηριστικά εικονοστοιχεία των δύο εικόνων. Οι παρακάτω δύο αλγόριθμοι, χρησιμοποιούν τις φασματικές υπογραφές των εικονοστοιχείων στις δύο εικόνες και συγκρίνουν τις τιμές αυτές. Για το λόγο αυτό, στο υποκεφάλαιο αυτό παρουσιάστηκαν κάποιες από τις τιμές αυτές έτσι ώστε ο χρήστης να πειστεί πως οι μέθοδοι αυτοί παράγουν καλά αποτελέσματα για την εφαρμογή του.

Στα παρακάτω Σχήμα 7.11, παρουσιάστηκαν οι δύο εικόνες και ένας πίνακας που περιείχε τις μεταβολές στις τιμές των αντίστοιχων φασματικών υπογραφών. Στις εικόνες έχει σημειωθεί η περιοχή στην οποία ανήκουν τα αντιπροσωπευτικά σημεία που χρησιμοποιήθηκαν για δειγματοληψία φασματικών υπογραφών. Τα σημεία αυτά πάρθηκαν σε περιοχές που παρουσίαζαν αλλαγές από την μια εικόνα στην άλλη, ενώ τα περισσότερα βρίσκονται, όπως φαίνεται και παρακάτω, κατά μήκος της Αττικής Οδού.



(α′) Σημεία στην εικόνα του 2000

(β′) Σημεία στην εικόνα 2007

Σχήμα 7.11: Περιοχές που ανήκουν τα σημεία που χρησιμοποιήθηκαν για δειγματοληψία

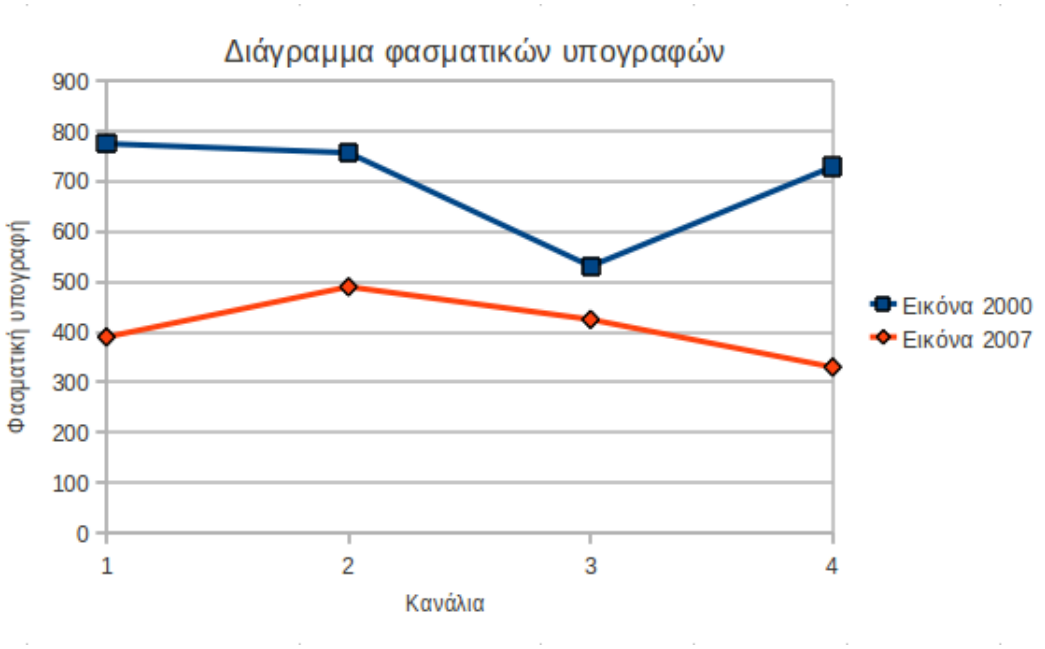
Μέσω λοιπόν το γραφικού περιβάλλοντος του orfeo, το Monteverdi, και του εργαλείου

του Spectral Viewer, προσδιορίστηκαν οι πίνακες που περιέχουν τις φασματικές υπογραφές των ομόλογων σημείων για όλα τα κανάλια των δύο εικόνων.

Σημεία	**Blue (1)**		**Green (2)**		**Red (3)**		**NIR (4)**	
Έτος	2000	2007	2000	2007	2000	2007	2000	2007
Σημείο 1	860	530	840	630	600	500	678	460
Σημείο 2	650	350	750	440	500	410	627	290
Σημείο 3	880	220	850	410	627	390	840	180
Σημείο 4	660	390	640	450	420	410	630	350
Σημείο 5	850	580	840	620	620	430	820	500
Σημείο 6	750	270	620	390	418	408	780	200
Μ.Ο.	775	390	757	490	531	425	729	330
Σημείο 7	1100	570	1110	600	840	420	1050	400
Σημείο 8	730	1500	750	1700	530	1220	610	1190
Σημείο 9	1040	1050	1050	1100	800	820	980	1180

Πίνακας 7.1: Φασματικές υπογραφές σημείων στην εικόνα του έτους 2000

Έτσι δημιουργήθηκε το Σχήμα 7.12, το οποίο αποτελεί μια γραφική αναπαράσταση των μέσων όρων των 6 πρώτων σημείων που αντιστοιχούσαν σε περιοχές κατά μήκους της Αττικής Οδού. Το γράφημα αποτελείτο από τον οριζόντιο άξονα που περιέχονται τα κανάλια της εικόνας και το οριζόντιο άξονα που αποτελείτο από την τιμή του μέσου όρου των φασματικών υπογραφών για τις δύο εικόνες.



Σχήμα 7.12: Διάγραμμα φασματικών υπογραφών για τις δυο εικόνες

Έτσι από τα παραπάνω Σχήμα 7.12, εξάγονται τα εξής συμπεράσματα:

- Αρχικά οι φασματικές υπογραφές, ειδικά αυτές που βρίσκονταν κατά μήκος της Αττικής

Οδού παρουσίασαν αρκετή διαφορά ανά κανάλι στις δύο εικόνες. Μάλιστα, το κανάλι που παρουσίασε την μεγαλύτερη μεταβολή είναι το μπλε και το υπέρυθρο.

- Ακόμα και μεταξύ των διαφόρων καναλιών της ίδιας εικόνας παρουσιάστηκαν αλλαγές. Μάλιστα η μεγαλύτερη διαφορά μεταξύ διαδοχικών καναλιών παρουσιάστηκε μεταξύ του κόκκινου και του πράσινου καναλιού.
- Επίσης, η διαφορά μεταξύ των καναλιών άλλαζε ανάλογα με την μεταβολή που παρατηρείται. Έτσι οι μεταβολές που οφείλονταν σε ύπαρξη καινούργιων κατασκευών, ανιχνεύονταν με βάση την φασματική τους υπογραφή, καλύτερα από ότι για παράδειγμα η μεταβολή στο οδόστρωμα της Αττικής Οδού.

7.3.3.2 Θεωρία

Όπως και στις προηγούμενες δύο μέθοδοι, ο αλγόριθμος χωρίστηκε σε δύο ξεχωριστά μέρη. Το πρώτο μέρος περιείχε τον αλγόριθμο του Διαχωρισμού καναλιών, ενώ το δεύτερο μέρος, εφάρμοζε στο αποτέλεσμα του μετασχηματισμού τον κανονικοποιημένο λόγο. Για κάθε μια εικόνα οπότε, δημιουργήθηκε η αντίστοιχη μετασχηματισμένη με βάση τους αντίστοιχους τύπους, και στην συνέχεια, το αποτέλεσμά της, εισήχθηκε στο αλγόριθμο του κανονικοποιημένου λόγου.

Ο αλγόριθμος έχει εφαρμογή μόνο σε πολυφασματικά δεδομένα. Καταρχάς, κατωφλιώνει τα εικονοστοιχεία των εικόνων, ανάλογα με το πόσο κόκκινο, μπλε, πράσινο ή υπέρυθρο εμφανίζεται ένα αντικείμενο και στη συνέχεια διαχωρίζονται όσα εικονοστοιχεία θεωρήθηκαν ότι έχουν ένα συγκεκριμένο χρώμα. Η τιμή του χρώματος των εικονοστοιχείων καθορίζεται με τον μετασχηματισμό φασματικής γωνίας. Αυτό γίνεται αντιληπτό αν θεωρηθεί το χρώμα ως διάνυσμα του τρισδιάστατου χώρου στο οποίο ο αριθμός των καναλιών τοποθετούνται σε σειρά σύμφωνα με το μήκος κύματος. Το φάσμα του εικονοστοιχείου παριστάνεται από το διάνυσμα:

$$\overrightarrow{P} = (P_1, P_2, P_3, P_4)$$

Το φάσμα αναφοράς που αντιπροσωπεύει τα κανάλια, είναι το παρακάτω

$$NIR = (0, 0, 0, 1)$$
$$RED = (0, 0, 1, 0)$$
$$GREEN = (0, 1, 0, 0)$$
$$BLUE = (1, 0, 0, 0)$$

Έτσι λοιπόν, ανάλογα με το κανάλι που προσδιορίζεται η φασματική γωνία ο τύπος μετασχηματισμού είναι:

$$cos(a_i) = \frac{P_i}{(\sum(P_k^2))^{1/2}}$$

Ο αλγόριθμος Διαχωρισμού Χρωμάτων εφαρμόζει την παραπάνω σχέση σε κάθε εικόνα και στη συνέχεια εφαρμόζει τον αλγόριθμο λόγου, όπως θα αναλυθεί παρακάτω, για να παράγει το τελικό αποτέλεσμα της ανίχνευσης μεταβολών.

Με τη χρήση εικόνων υψηλής χωρικής ανάλυσης όπως του Ikonos, ο αλγόριθμος παράγει πολύ καλά αποτελέσματα σε περιοχές που εκτιμώνται μεταβολές σε κτίρια ή άλλες τεχνητές περιοχές. Όμως όσο μειώνεται η ανάλυση των δεδομένων, τόσο θα μειώνεται και η αποδοτικότητα της συγκεκριμένης τεχνικής. [Καραλής, 2008]

7.3.3.3 Προγραμματισμός

Όπως και οι προηγούμενες δύο μέθοδοι, το πρόγραμμα χωρίστηκε σε δύο ξεχωριστά μέρη. Το πρώτο μέρος περιείχε το εκτελέσιμο αρχείο που παρήγαγε τόσες εικόνες όσα ήταν και τα ξεχωριστά κανάλια, εφαρμόζοντας τον ανάλογο τύπο μετασχηματισμού, ενώ το δεύτερο μέρος περιείχε τον αλγόριθμο του σταθμισμένου λόγου που πραγματοποίησε την ανίχνευση των μεταβολών. Τα επιμέρους προγράμματα, παρόλο το μέγεθος των εικόνων, δεν χρειάστηκαν πολύ χρόνο και υπολογιστικούς πόρους να να παράξουν το τελικό αποτέλεσμα.

Στο σημείο αυτό παρουσιάζεται σε ψευδογλώσσα ο αλγόριθμος Διαχωρισμού Χρωμάτων. (Σχημα 7.13)

```
Αλγόριθμος Διαχωρισμού Χρωμάτων
    Δήλωση των απαραίτητων τύπων μεταβλητών
    Δημιουργία του δείκτη reader
    reader → argv[1]
    Έλεγχος σωστής λειτουργίας της ροής
    Εξαγωγή των καναλιών της εικόνας μέσω της αντίστοιχης συνάρτησης (band)
    Για j=0 μέχρι j=band με βήμα 1
        Δημιουργία και αρχικοποίηση της παραγόμενης εικόνας με μηδέν
        Όλοι οι επαναλήπτες πηγαίνουν στην αρχή της εικόνας
        Όσο η αρχική εικόνα δεν είναι στο τέλος της επανέλαβε
            Για i=0 μέχρι i=band με βήμα 1
                pixelInput += inputIt.Get()[i]*inputIt.Get()[i]
                temp=inputIt.Get()[j]
            Τέλος επανάληψης
            pixelOutput = static cast<PixelType>(temp/sqrt(pixelInput))
            outputIt.Set(pixelOutput)
            Οι επαναλήπτες πηγαίνουν στο επόμενο στοιχείο
        Τέλος επανάληψης
        Δημιουργία δείκτη writer
        writer → argv[2+j]
    Τέλος επανάληψης
Τέλος αλγορίθμου
```

Σχήμα 7.13: Διάγραμμα ροής για τον αλγόριθμο Διαχωρισμού Χρωμάτων

Αναλυτικότερα τώρα, παρουσιάζεται το εκτελέσιμο αρχείο που δημιουργήθηκε:

- Αρχικά ορίστηκαν τα αρχεία επικεφαλίδες όπως κάθε φορά, ορίστηκαν επίσης οι διαστάσεις της εικόνας, οι τύποι των μεταβλητών που χρησιμοποιήθηκαν, ενώ ο δείκτης του αναγνώστη έλαβε την τιμή του. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 18-65)

- Ακολούθησε η δήλωση του επαναλήπτη για την εικόνα που εισήχθηκε στο πρόγραμμα, η εξαγωγή του αριθμού των καναλιών της εικόνας και η αρχικοποίηση των απαραίτητων στοιχείων. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 69-72)

- Ο αλγόριθμος, δεν γνώριζε από την αρχή πόσες εικόνες παράγονταν, καθώς γνώριζε πόσα ήταν τα κανάλια της εικόνας που εισήχθηκαν σε αυτόν. Για το λόγο αυτό, δημιουργήθηκε μια επανάληψη, στην οποία, ξεχωριστά κάθε φορά παράχθηκε η ανάλογη εικόνα εξόδου. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 76)

- Ακολούθησε, όπως κάθε φορά, η δήλωση των μεταβλητών της εικόνας που παράχθηκε και η αρχικοποίησή της, ενώ όλοι οι επαναληπτές αρχικοποιήθηκαν στην αρχή της περιοχής που προσδιόριζαν. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 80-89)

- Στην συνέχεια, έγινε η επανάληψη που διέτρεχε όλα τα εικονοστοιχεία της εικόνας και ορίστηκε ο τύπος των εικονοστοιχείων της εικόνας που παράχθηκε. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 93-97)

- Έπειτα, για κάθε κανάλι, ο αλγόριθμος πραγματοποίησε το άθροισμα του γινομένου για κάθε εικονοστοιχείο, ενώ παράλληλα, αποθήκευσε την τιμή του εικονοστοιχείου που ορίστηκε από το συγκεκριμένο κανάλι, ανάλογα με την εξωτερική επανάληψη. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 101-112)

- Ακολούθησε η μετάβαση της τιμής του εικονοστοιχείου που προέκυψε στην τελική εικόνα και η αρχικοποίηση ή αύξηση των τιμών όλων των επαναλήπτων. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί πως δεν πραγματοποιήθηκε κάποιος έλεγχος στον παρανομαστή της σχέσης, καθώς πρόκειται για άθροισμα και οι τιμές των εικονοστοιχείων είναι διπλής ακρίβειας. Έτσι η περίπτωση ο παρανομαστής να είναι μηδέν ήταν απίθανη. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 106-112)

- Τέλος, και πριν τελειώσει και η εξωτερική επανάληψη, δημιουργήθηκε ο αντίστοιχος γραφέας, λαμβάνοντας το αντίστοιχο όρισμα και ελέγχοντας αν η ροή εκτελείτο με το σωστό τρόπο. (παράρτημα, 7.6 Αλγόριθμος Διαχωρισμού Χρωμάτων, 115-130)

Στην περίπτωση των εικόνων που χρησιμοποιήθηκαν στην συγκεκριμένη διπλωματική, οι εικόνες που παράχθηκαν ήταν τέσσερις, καθώς τέσσερα κανάλια διέθεται ο δορυφόρος Ikonos. Στις αντίστοιχες εικόνες που παράχθηκαν, πραγματοποιήθηκε κανονικοποιημένος λόγος, όπως δημιουργήθηκε στα πλαίσια της συγκεκριμένης διπλωματικής.

Μια μέθοδος ανίχνευσης μεταβολών αποτελεί ο λόγος των εικόνων $\frac{I_2}{I_1}$. Το περιβάλλον του OTB διαθέτει αλγόριθμο λόγου καναλιών που προέκυψε από την σχέση $1 - \frac{meanB}{meanA}$ όπου γινόταν έλεγχος ποιο στοιχείο είναι μεγαλύτερο για να μπει στον αριθμητή. Στην περίπτωση του αλγορίθμου που δημιουργήθηκε, υπολογίστηκε ο κανονικοποιημένος λόγος όπως προκύπτει από τον τύπο $\frac{(meanB-meanA)}{(meanB+meanA)}$.

Ο συγκεκριμένος αλγόριθμος ακολουθεί την δομή του NomarlizedMeanDifferenceImageFilter που παρουσιάστηκε στο κεφάλαιο 6, και για το λόγο αυτό δεν κρίνεται σκόπιμο να αναλυθεί σε λεπτομέρεια. Πάλι κατασκευάστηκαν τρία ξεχωριστά αρχεία τα οποία επισυνάπτονται στο παράρτημα. Το πρώτο αρχείο επικεφαλίδα otbNomarlizedMeanRatioImageFilter.h (παράρτημα 7.7 Αρχείο επικεφαλίδα για λόγο εικόνων) περιείχε όλους τους ορισμούς της κλάσης που δημιουργήθηκε. Το επόμενο αρχείο που δημιουργήθηκε ήταν το otbNomarlizedRatio.h (παράρτημα 7.8 Αρχείο επικεφαλίδα για λόγο εικόνων) και περιείχε την δημιουργία της σχέσης που εφαρμόστηκε, δηλαδή του κανονικοποιημένου λόγου. Τέλος, τα δύο αρχεία συνοδεύονταν από το αντίστοιχο εκτελέσιμο (παράρτημα, 7.9 Εκτελέσιμο αρχείο για λόγο εικόνων), το οποίο καλούσε τα αρχεία επικεφαλίδες και λάμβανε τις εικόνες που εισάγονταν για επεξεργασία.

Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί πως στο εκτελέσιμο αρχείο των λόγων, χρησιμοποιήθηκε το πρότυπο της απόλυτης τιμής, ώστε η τιμή του αποτελέσματος να είναι πάντα θετική, αλλά και το πρότυπο ενίσχυσης. Το αποτέλεσμα, προκειμένου να γίνει ορατό στο χρήστη υπέστη μια ενίσχυση μέσω το φίλτρου itkShiftScaleImageFilter.

7.3.3.4 Εκτέλεση- Αποτελέσματα

Όπως έγινε κατανοητό και παραπάνω, τα αποτελέσματα του παραπάνω αλγορίθμου δίνονται σε δύο στάδια. Αρχικά το πρώτο εκτελέσιμο αρχείο που περιείχε τον μετασχηματισμό, εφαρμόστηκε και στις δύο εικόνες. Από την εφαρμογή του παράχθηκαν τέσσερις εικόνες για κάθε μια εικόνα. Έτσι για την εικόνα του 2000 παράχθηκαν τα παρακάτω αποτελέσματα: (Σχήμα 7.14)

(α′) Μετασχηματισμός στο μπλε κανάλι (β′) Μετασχηματισμός στο πράσινο κανάλι

(γ′) Μετασχηματισμός στο κόκκινο κανάλι (δ′) Μετασχηματισμός στο υπέρυθρο κανάλι

Σχήμα 7.14: Αποτελέσματα αλγορίθμου Διαχωρισμού Χρωμάτων για την εικόνα του έτους 2000

Τα παραπάνω αποτελέσματα (Σχήμα 7.14) προέκυψε πως το υπέρυθρο κανάλι εμφανίσε καλύτερα την βλάστηση, ενώ ο μετασχηματισμός στο κόκκινο και πράσινο κανάλι δεν παράγει τόσο καλά αποτελέσματα, καθώς η εικόνα παρουσιάστηκε σχεδόν ξεθωριασμένη.

Αντίστοιχα αποτελέσματα (Σχήμα 7.15) παρουσίασε και η εικόνα του 2007. Αυτό που έγινε πολύ έντονα αντιληπτό ήταν το γεγονός ότι στο μπλε και υπέρυθρο κανάλι η Αττική Οδός παρουσίαζε μεγάλη διαφορά από το ένα έτος στο άλλο. Η διαφορά αυτή παρουσιαζόταν και στα άλλα δύο κανάλια, αλλά δεν ήταν τόσο έντονη, οπότε αναμένεται να μην παράγει πολύ καλά αποτελέσματα ο αλγόριθμος του λόγου.

(α′) Μετασχηματισμός στο μπλε κανάλι (β′) Μετασχηματισμός στο πράσινο κανάλι

(γ′) Μετασχηματισμός στο κόκκινο κανάλι (δ′) Μετασχηματισμός στο υπέρυθρο κανάλι

Σχήμα 7.15: Τα αποτελέσματα αλγορίθμου Διαχωρισμού Χρωμάτων για την εικόνα του έτους 2007

Αφού συγκεντρώθηκαν όλες οι εικόνες από τον μετασχηματισμό, δόθηκαν σαν είσοδο στο εκτελέσιμο αρχείο του κανονικοποιημένου λόγου. Τα αποτελέσματα που παράχθηκαν εμφανίζονται στο Σχήμα 7.16

(α′) Αλγόριθμος λόγου στο μπλε κανάλι (β′) Αλγόριθμος λόγου στο πράσινο κανάλι

Σχήμα 7.16: Αποτελέσματα αλγορίθμου κανονικοποιημένου λόγου για αλγόριθμο Διαχωρισμού Χρωμάτων

(γ΄) Αλγόριθμος λόγου στο κόκκινο κανάλι (δ΄) Αλγόριθμος λόγου στο υπέρυθρο κανάλι

Σχήμα 7.16: Αποτελέσματα αλγορίθμου κανονικοποιημένου λόγου για αλγόριθμο Διαχωρισμού Χρωμάτων

Όπως ήταν αναμενόμενο λοιπόν, ο μετασχηματισμός στο μπλε και υπέρυθρο κανάλι έδωσε τα καλύτερα αποτελέσματα, ενώ οι άλλες δύο εικόνες δεν εμφάνισαν τόσο καλά αποτελέσματα για την Αττική Οδό. Το συγκεκριμένο αποτέλεσμα ήταν αναμενόμενο μετά από την ανάλυση των φασματικών υπογραφών των χαρακτηριστικών στην εικόνα σημείων. Σε κάθε περίπτωση όμως, ο αλγόριθμος παρήγαγε πολύ καλά αποτελέσματα και μπορεί να χρησιμοποιηθεί σε περιπτώσεις ανίχνευσης μεταβολών. Ανάλογα με την αλλαγή που θέλει να προσδιορίσει ο χρήστης μπορεί να επιλέξει το ανάλογο κανάλι και να πραγματοποιήσει τον λόγο σε αυτό. Οι αλλαγές σε κάθε περίπτωση παρουσιάστηκαν με λευκές και μαύρες αποχρώσεις, ενώ οι αμετάβλητες περιοχές με γκρι αποχρώσεις.

7.3.4 Αλγόριθμος Διαφορών κλίσεων (Band-Slope Difference)

7.3.4.1 Θεωρία

Ο συγκεκριμένος αλγόριθμος αποτελείτο από δυο τμήματα. Το πρώτο στηρίζεται στην διαφορά διαδοχικών καναλιών και ανίχνευση των μεταβολών μέσω του δείκτη αυτού. Ο συγκεκριμένος δείκτης αναδεικνύει την μεταβολή στην φασματική υπογραφή του εικονοστοιχείου (αποτελεί δηλαδή την πρώτη παράγωγο της φασματικής υπογραφής). Αφού λοιπόν υπολογιστεί η καινούργια εικόνα με χρήση του συγκεκριμένου δείκτη, εφαρμόζεται ο αλγόριθμος του κανονικοποιημένου λόγου [Καραλής, 2008]. Η διαφορά της κλίσης μεταξύ διαδοχικών καναλιών περιγράφεται από τους τύπους:

$$D_1 = BV_{k+1}(1) - BV_k(1)$$
$$D_2 = BV_{k+1}(2) - BV_k(2)$$

όπου D_1 = η μεταβολή στην τιμή φωτεινότητας στην εικόνα 1
D_2 = η μεταβολή στην τιμή φωτεινότητας στην εικόνα 2
$BV_{k+1}(1)$ = η τιμή φωτεινότητας στο κανάλι k+1 της εικόνας 1
$BV_k(1)$ = η τιμή φωτεινότητας στο κανάλι k της εικόνας 1
$BV_{k+1}(2)$ = η τιμή φωτεινότητας στο κανάλι k+1 της εικόνας 2
$BV_k(2)$ = η τιμή φωτεινότητας στο κανάλι k της εικόνας 2

7.3.4.2 Προγραμματισμός

Στο σημείο αυτό παρουσιάζεται ο τρόπος με τον οποίο δομήθηκε ο αλγόριθμος της διαφοράς κλίσεων και ο τρόπος με τον οποίο αυτός λειτουργεί. Για να γίνει αυτό πιο κατανοητό έγινε μια προσπάθεια να παρασταθεί σε ψευδογλώσσα η ροή των εντολών. (Σχήμα 7.17) Έτσι λοιπόν ο αλγόριθμος σε ψευδογλώσσα θα μπορούσε να είναι ως εξής:

```
Αλγόριθμος Διαφορών Κλίσεων
    Δήλωση των απαραίτητων τύπων μεταβλητών
    Δημιουργία του δείκτη αναγνώστη
    reader → argv[1]
    Έλεγχος σωστής λειτουργίας της ροής
    Εξαγωγή των καναλιών της εικόνας μέσω της αντίστοιχης συνάρτησης (band)
    Για i=0 μέχρι i=band-1 με βήμα 1
        Δημιουργία και αρχικοποίηση της παραγόμενης εικόνας με μηδέν
        Όλοι οι επαναλήπτες πηγαίνουν στην αρχή της εικόνας
        Όσο η αρχική εικόνα δεν είναι στο τέλος της επανέλαβε
            pixelInput = inputIt.Get()[i+1]-inputIt.Get()[i]
            pixelOutput = pixelInput
            outputIt.Set(pixelOutput)
            Οι επαναλήπτες πηγαίνουν στο επόμενο στοιχείο
        Τέλος επανάληψης
        Δημιουργία δείκτη γραφέα
        writer → argv[2+i]
    Τέλος επανάληψης
Τέλος αλγορίθμου
```

Σχήμα 7.17: Αλγόριθμος Διαφορών κλίσεων σε ψευδογλώσσα

Με βάση το παραπάνω προσχέδιο, (Σχήμα 7.17), δημιουργήθηκε το εκτελέσιμο αρχείο που παρήγαγε το τελικό αποτέλεσμα και υπολόγιζε τις διαδοχικές αλλαγές στην φασματική υπογραφή των εικονοστοιχείων. Παρακάτω, παρουσιάζεται με κάθε λεπτομέρεια η διαδικασία που ακολουθήθηκε.

- Αρχικά, ορίστηκαν όλα τα αρχεία επικεφαλίδες που ήταν απαραίτητα για την λειτουργία του αλγορίθμου. Τα αρχεία αυτά όρισαν τις μεθόδους του αναγνώστη, και του γραφέα, τους τύπους των εικόνων που παράχθηκαν και τους τύπους των επαναληπτών. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 19-24)

- Στην συνέχεια, ορίστηκαν οι τύποι όλων των μεταβλητών, και ο δείκτης του αναγνώστη πήρε την τιμή εισόδου του. Έπειτα ελέγχθηκε αν η μετάβαση αυτή της τιμής έγινε με το σωστό τρόπο. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 34-63)

- Όπως και στον προηγούμενο αλγόριθμο, τα αποτελέσματα που παράχθηκαν από τον αλγόριθμο εξαρτώνται από τον αριθμό των καναλιών της εικόνας, καθώς ο αλγόριθμος υπολόγιζε τις διαδοχικές διαφορές καναλιών. Για το λόγο αυτό, η εικόνα που παράχθηκε, ορίστηκε μέσα στην επανάληψη, έτσι ώστε να εκτελείτο για κάθε διαφορά. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 72-80)

- Οι επαναλήπτες αρχικοποιήθηκαν στην αρχή της εικόνας που προσδιόριζαν και δηλώθηκε και ο τύπος των εικονοστοιχείων της εικόνας που παράχθηκε. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 84-88)

- Ακολούθησε η επανάληψη μέχρι η αρχική εικόνα να φτάσει στο τέλος της. Στην επανάληψη πραγματοποιήθηκε η διαφορά μεταξύ των διαδοχικών καναλιών και η τιμή της διαφοράς πέρασε στην μεταβλητή για την εικόνα που δημιουργήθηκε. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 91-99)

- Τέλος, ακολούθησε η δημιουργία του γραφέα και το πέρασμα της τιμής σε αυτό ώστε να παραχθεί το τελικό αποτέλεσμα. (παράρτημα, 7.10 Εκτελέσιμο αρχείο για Διαφορά Κλίσεων, 103-109)

Στην συνέχεια, τα αποτελέσματα του παραπάνω αλγορίθμου εισήχθησαν στον αλγόριθμο του λόγου καναλιών και παρήγαγαν τις τελικές εικόνες ανίχνευσης μεταβολών.

7.3.4.3 Εκτέλεση- Αποτελέσματα

Χρησιμοποιώντας τις εικόνες Ikonos για την εφαρμογή, παράγονται τρεις ξεχωριστές εικόνες που περιέχουν τις διαδοχικές διαφορές των καναλιών. Τα αποτελέσματα παρουσιάζονται παρακάτω στο Σχήμα 7.18.

(α′) Αποτέλεσμα διαφοράς καναλιού πράσινου με μπλε

(β′) Αποτέλεσμα διαφοράς κόκκινου με πράσινου

(γ′) Αποτέλεσμα διαφοράς υπερύθρου με κόκκινο

Σχήμα 7.18: Τα αποτελέσματα αλγορίθμου διαφορών κλίσεων για εικόνα έτους 2000

Παρατηρείται πως τα αποτελέσματα (Σχήμα 7.18) ήταν αρκετά καλά καθώς όπως προέκυψε από τις φασματικές υπογραφές, οι διαφορές των φασματικών υπογραφών στα διάφορα κανάλια ήταν μεγάλες. Η μεγαλύτερη διαφορά παρατηρήθηκε στην διαφορά του πράσινου καναλιού από το μπλε. Το συγκεκριμένο συμπέρασμα ήταν αναμενόμενο, καθώς και από την ανάλυση των φασματικών υπογραφών, η διαφορά αυτή παρουσίαζε την μεγαλύτερη απόκλιση. Η ίδια διαδικασία έγινε και για την εικόνα του έτους 2007.

(α′) Αποτέλεσμα διαφοράς καναλιού πράσινου με μπλε

(β′) Αποτέλεσμα διαφοράς κόκκινου με πράσινου

Σχήμα 7.19: Αποτελέσματα αλγορίθμου διαφορών κλίσεων για εικόνα έτους 2007

(γ′) Αποτέλεσμα διαφοράς υπερύθρου με κόκκινο

Σχήμα 7.19: Αποτελέσματα αλγορίθμου διαφορών κλίσεων για εικόνα έτους 2007

Παρόμοια αποτελέσματα λοιπόν, (Σχήμα 7.19), παράχθηκαν και για την εικόνα του έτους 2007. Η Αττική Οδός παρουσίαζε μεγαλύτερη διαφορά μεταξύ των δύο εικόνων στην πρώτη διαφορά, δηλαδή σε αυτή των καναλιών πρασίνου και κόκκινου. Έτσι στο σημείο αυτό, εφαρμόστηκε ο αλγόριθμος του λόγου για τις αντίστοιχες εικόνες. Τα αποτελέσματα είναι τα παρακάτω:

(α′) Αποτέλεσμα λόγου στην διαφορά πρασίνου-μπλε

(β′) Αποτέλεσμα λόγου στην διαφορά κόκκινου-πράσινου

(γ′) Αποτέλεσμα λόγου διαφοράς υπερύθρου με κόκκινο

Σχήμα 7.20: Αποτελέσματα αλγορίθμου κανονικοποιημένου λόγου για αλγόριθμο Διαφορών Κλίσεων

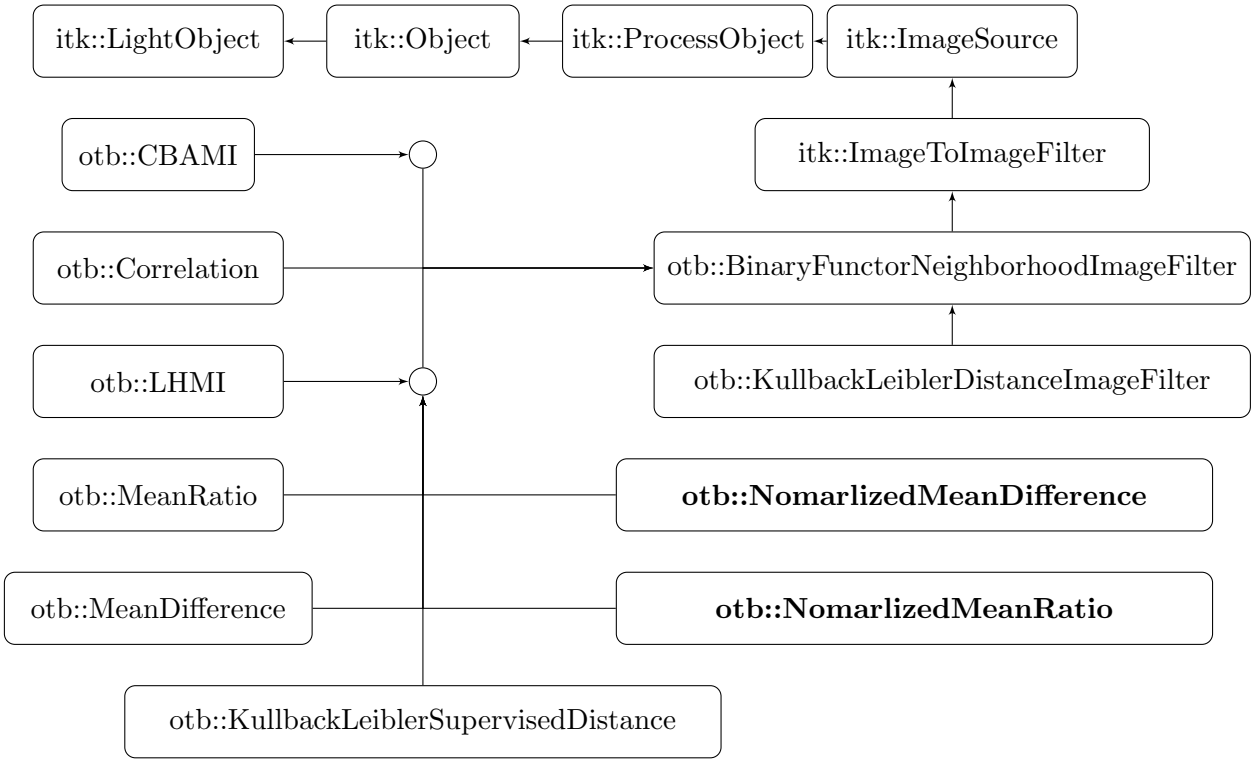
Τα παραπάνω αποτελέσματα (Σχήμα 7.20) προσέφεραν μια πολύ καλή προσέγγιση της μεταβολής της Αττικής Οδού. Βέβαια, η καλύτερη ανίχνευση της συγκεκριμένης μεταβολής έγινε από την διαφορά καναλιών του κόκκινου και του πράσινου.

7.3.5 Γενικά στοιχεία αλγορίθμων

Στο σημείο αυτό κρίνεται σκόπιμο να παρουσιαστούν σε διαγράμματα UML των αλγορίθμων που δημιουργήθηκαν και προστέθηκαν τους αλγορίθμους ανίχνευσης μεταβολών. Οι αλγόριθμοι αυτοί δεν ήταν άλλοι από τον αλγόριθμο κανονικοποιημένης διαφοράς και κανονικοποιημένου λόγου. Στους συγκεκριμένους αλγορίθμους η σχεδίασή τους έγινε σε επίπεδο

κλάσης, χρησιμοποιώντας άλλες βασικές κλάσεις για την εξάρτησή τους. Έτσι η βασική κλάση που χρησιμοποιήθηκε ήταν αυτή της BinaryFunctorNeighborhoodImageFilter η οποία χρησιμοποιήθηκε για όλους τους αλγορίθμους ανίχνευσης μεταβολών που υπάρχουν στο Orfeo Toolbox.

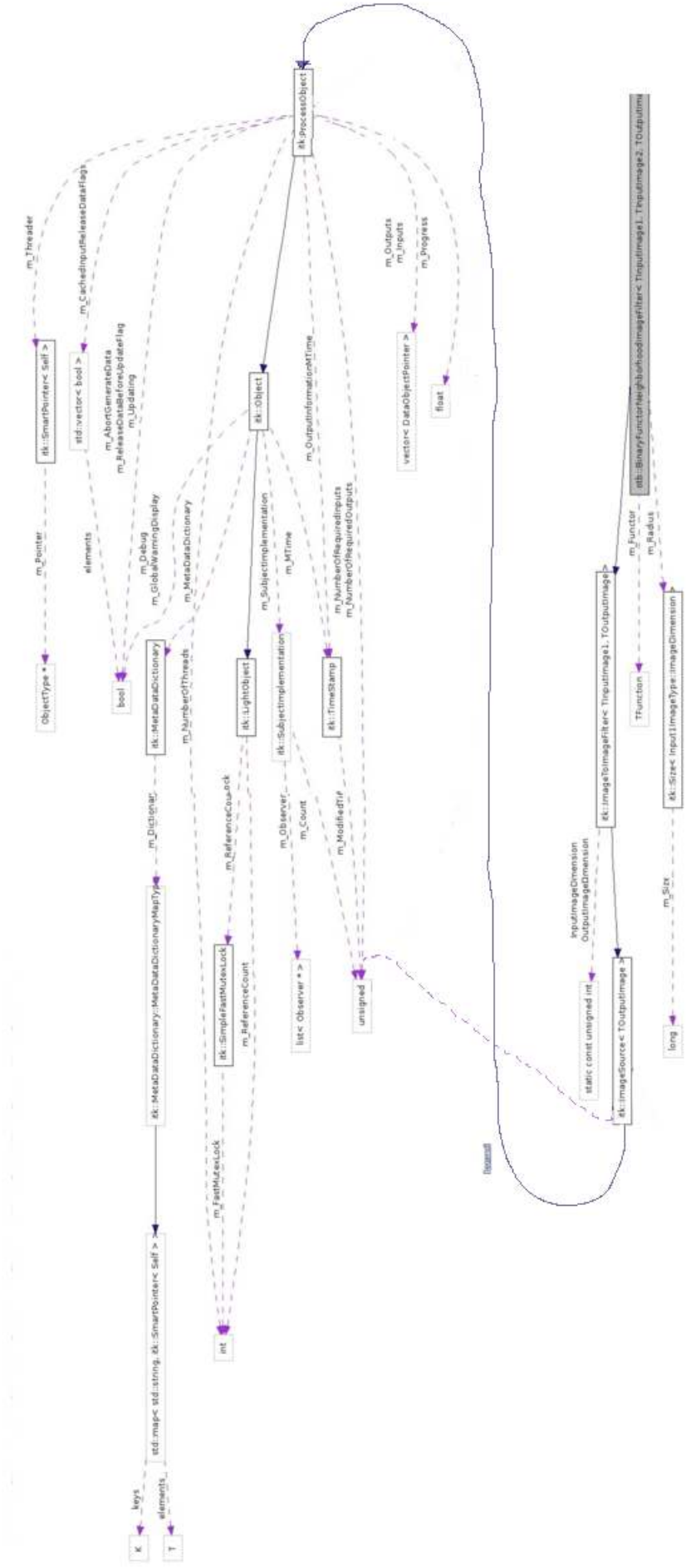
Έτσι, το διάγραμμα κληρονομικότητας για την συγκεκριμένη κλάση θα είναι ως εξής (Σχήμα 7.21):



Σχήμα 7.21: Διάγραμμα Κληρονομικότητας

Από το παραπάνω διάγραμμα, (Σχήμα 7.21), προκύπτει πως όλες οι κλάσεις που εφαρμόζουν φίλτρα ανίχνευσης μεταβολών στις εικόνες κληρονομούν ιδιότητες από την BinaryFunctorNeighborhoodImageFilter, όπου με την σειρά της κληρονομεί ιδιότητες από βασικές κλάσεις της ITK. Με τον συγκεκριμένο τρόπο δομούνται όλες οι κλάσεις για την ανίχνευση μεταβολών. Με έντονη γραμματοσειρά, παρουσιάζονται οι κλάσεις που δημιουργήθηκαν στα πλαίσια της συγκεκριμένης εργασίας.

Τέλος, κρίνεται σκόπιμο να παρουσιαστεί το διάγραμμα UML της συνεργασίας (Collaboration diagram) της κλάσης BinaryFunctorNeighborhoodImageFilter με όλα τα άλλα αντικείμενα που χρησιμοποιεί και δημιουργεί. Παρακάτω λοιπόν παρουσιάζεται το συγκεκριμένο διάγραμμα όπως παρουσιάζεται στο doxygen της πλατφόρμας.



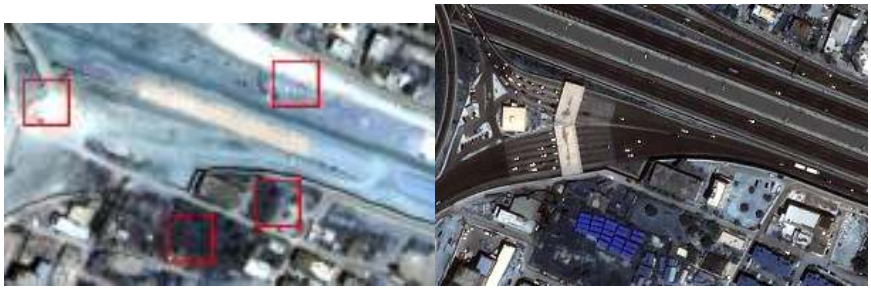
Σχήμα 7.22: Διάγραμμα συνεργασίας (Collaboration diagram). Πηγή OTB Doxygen

7.3.6 Αξιολόγηση αλγορίθμων

7.3.6.1 Εισαγωγή

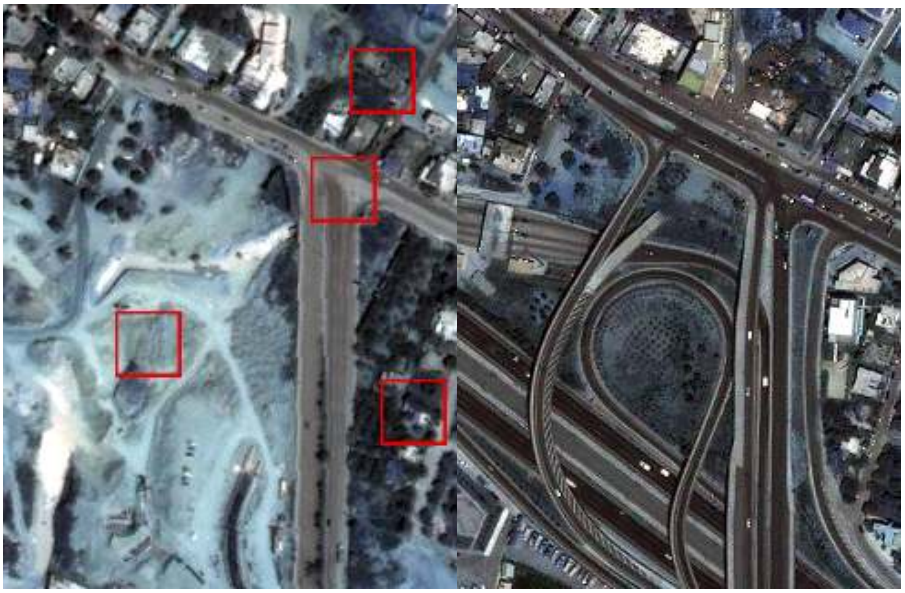
Στο σημείο αυτό, κρίθηκε σκόπιμο, να γίνει μια αξιολόγηση των αποτελεσμάτων που προέκυψαν από την χρήση των αλγορίθμων ανίχνευσης μεταβολών που παρουσιάστηκαν. Για το λόγο αυτό, επιλέχτηκε η φωτοερμηνεία ως ένα μέσω αξιολόγησης των παρεχόμενων αποτελεσμάτων. Έτσι και αλλιώς η φωτοερμηνεία, αποτελεί έναν από τους βασικότερους τρόπους αξιολόγησης αποτελεσμάτων που προέρχονται από αλγορίθμους όλων των εφαρμογών.

Από τις δύο εικόνες λοιπόν, που περιείχαν την περιοχή μελέτης πριν και μετά την κατασκευή της Αττικής Οδού, απομονώθηκαν τέσσερα κομμάτια, στα οποία οι αλλαγές ήταν ορατές. Οι αλλαγές αυτές αριθμήθηκαν σε κάθε εικόνα, ενώ αφορούσαν τόσο τις μεταβολές κατά μήκος της Αττικής Οδού, όσο και μεταβολές στον αστικό ιστό που τον περιβάλλουν.



(α′) Αποτέλεσμα πρώτης περιοχής για το έτος 2000

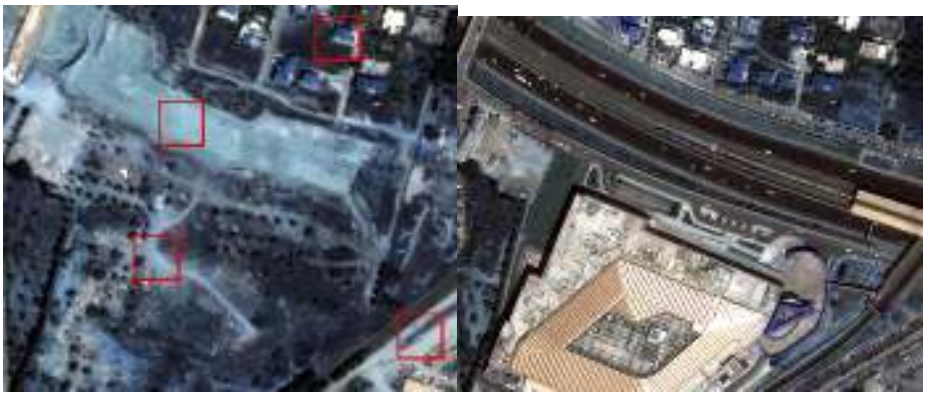
(β′) Αποτέλεσμα πρώτης περιοχής για το έτος 2007



(γ′) Αποτέλεσμα δεύτερης περιοχής για το έτος 2000

(δ′) Αποτέλεσμα δεύτερης περιοχής για το έτος 2007

Σχήμα 7.23: Περιοχές με μεταβολές



(ε΄) Αποτέλεσμα τρίτης περιοχής για το έτος 2000 (ϛ΄) Αποτέλεσμα τρίτης περιοχής για το έτος 2007

(ζ΄) Αποτέλεσμα τέταρτης περιοχής για το έτος 2000 (η΄) Αποτέλεσμα τέταρτης περιοχής για το έτος 2007

Σχήμα 7.23: Περιοχές με μεταβολές

Με τον τρόπο αυτό απομονώθηκαν οι τέσσερις αυτές χαρακτηριστικές περιοχές και αριθμήθηκαν όλες οι αλλαγές που διακρίνονταν σε αυτές. Έτσι στο σημείο αυτό, για κάθε αλγόριθμο ανίχνευσης μεταβολών που δημιουργήθηκε, συγκρίνονται οι αντίστοιχες περιοχές και δημιουργείται ένας πίνακας που προσδιορίζει το ποσοστό των αλλαγών που ανιχνεύτηκαν ή όχι από αυτόν.

7.3.6.2 Θεωρία

Προκειμένου να συγκριθούν τα αποτελέσματα από τους αλγορίθμους που δημιουργήθηκαν, κατασκευάστηκε ένας πίνακας σφαλμάτων, ή αλλιώς πίνακας μεταβλητότητας (confusion matrix) ή αλλιώς, contingency table, ο οποίος περιέχει, τα σφάλματα συμπερίληψης (comission) και παράληψης (omission). Το σφάλμα συμπερίληψης, αποτελεί το σφάλμα, υπολογισμού ενός αντικειμένου σε μια κλάση στην οποία δεν ανήκει. Στην περίπτωση της συγκεκριμένης εφαρμογής, το σφάλμα συμπερίληψης, αναφέρεται στον χαρακτηρισμό ενός εικονοστοιχείου (για την φωτοερμηνεία συστάδα σημείων), ως ανίχνευση χωρίς όμως αυτό να είναι ή και το αντίθετο. Το σφάλμα παράληψης, αναφέρεται στην παράληψη ενός αντικειμένου από την σωστή κατηγορία που ανήκει. [George Joseph, 2011]

Στην περίπτωση της συγκεκριμένης εφαρμογής, τα σφάλματα αυτά προήλθαν από φωτοερμηνεία τις αντίστοιχες εικόνες. Έτσι δημιουργήθηκε ο πίνακας σφαλμάτων όπου περιείχε τέσσερα διαφορετικά κελιά. Στο πάνω αριστερά κελί, αναφέρεται ο αριθμός των περιοχών που αποτελούσαν μεταβολή και ανιχνεύτηκαν από τον αλγόριθμο. Στο κάτω αριστερά

κελί, αναφέρονται ο αριθμός των περιοχών που δεν αποτελούν μεταβολή, αλλά ανιχνεύονται ως μια. Αντίθετα, πάνω δεξιά, βρίσκονται οι περιοχές που αποτελούν μεταβολή αλλά δεν ανιχνεύονται από τον αλγόριθμο. Το κάτω δεξιά κελί μένει πάντα χωρίς κάποια τιμή. [http://en.wikipedia.org/wiki/Groundtruth, 2011]

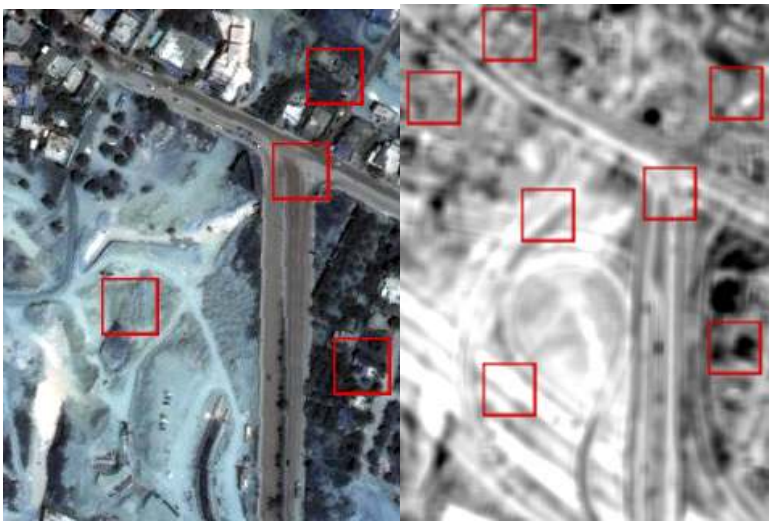
Κανονικά, η συγκεκριμένη αξιολόγηση, θα έπρεπε να πραγματοποιηθεί σε δεδομένα αληθών εικόνων. (ground truth), και οι εικόνες αυτές να συγκριθούν στην συνέχεια με τα αποτελέσματα των αλγορίθμων. Οι αληθής εικόνες, αποτελούν εικόνες που παρουσιάζουν την σημερινή κατάσταση στην αντίστοιχη περιοχή μελέτης. Οι εικόνες αυτές, μέσα από την διαδικασία που πραγματοποιήθηκε, και την δημιουργία των αντίστοιχων πινάκων, συγκρίνονται με αυτές που προκύπτουν από τους αλγορίθμους. Στην συγκεκριμένη περίπτωση, όμως, δεν υπάρχει κάποια αληθής εικόνα για να συγκριθούν τα αποτελέσματα των αλγορίθμων, οπότε προσδιορίστηκαν αλλαγές μεταξύ των δύο εικόνων και αυτές συγκρίθηκαν με τα αποτελέσματα των αλγορίθμων.

7.3.6.3 Αλγόριθμος Διαφοράς Έντασης

Το αποτέλεσμα του αλγορίθμου Διαφοράς Έντασης παρουσιάζεται στα παρακάτω σχήμα (Σχήμα 7.24)

(α′) Αποτέλεσμα πρώτης περιοχής για το έτος 2000

(β′) Αποτέλεσμα αλγορίθμου Διαφοράς Έντασης

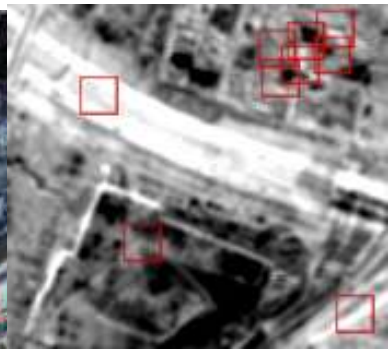


(γ′) Αποτέλεσμα δεύτερης περιοχής για το έτος 2000

(δ′) Αποτέλεσμα αλγορίθμου Διαφοράς Έντασης

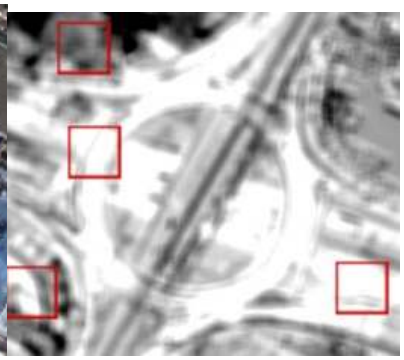
Σχήμα 7.24: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαφοράς Έντασης

(ε′) Αποτέλεσμα τρίτης περιοχής για το έτος 2000



(ϝ′) Αποτέλεσμα αλγορίθμου Διαφοράς Έντασης

(ζ′) Αποτέλεσμα τεταρτης περιοχής για το έτος 2000



(η′) Αποτέλεσμα αλγορίθμου Διαφοράς Έντασης

Σχήμα 7.24: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαφοράς Έντασης

Από το Σχήμα 7.24, προκύπτει ο παρακάτω πίνακας αξιολόγησης των αποτελεσμάτων. Ο πίνακας, για στήλες έχει το αν οι μεταβολές ανιχνεύτηκαν ή όχι από τον αλγόριθμο, ενώ για γραμμές έχει αν με την φωτοερμηνεία, ανιχνεύονται οι μεταβολές αυτές ή όχι. Έτσι λοιπόν προκύπτει ο παρακάτω πίνακας.

Αλγόριθμος / Φωτοερμηνεία	**Ανίχνευση μεταβολών**	**Μη ανίχνευση μεταβολών**
Μεταβολή	19/26 (73%)	5/26 (19%)
Μη μεταβολή	2/26 (8%)	–

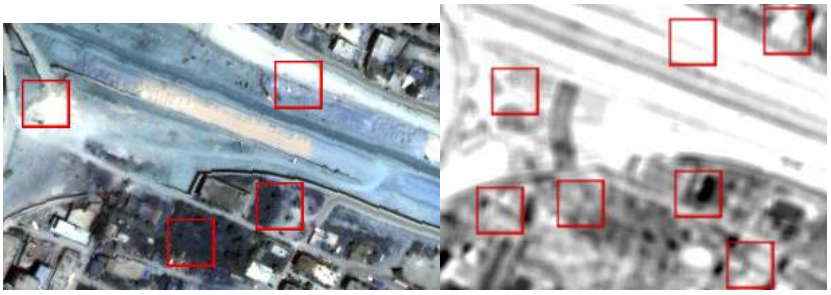
Πίνακας 7.2: Πίνακας αξιολόγησης για αλγόριθμο Διαφοράς Έντασης

Έτσι λοιπόν, από τον παραπάνω πίνακα προκύπτει πως ο αλγόριθμος δίνει αποδεκτά αποτελέσματα. Την μεταβολή κατά μήκος της Αττικής Οδούς, την ανιχνεύει με πάρα πολύ καλό τρόπο. Στο μόνο σημείο που υστερεί, και για αυτό και τα συγκεντρωτικά του αποτελέσματα δεν είναι τόσο καλά,ήταν η ανίχνευση του αστικού ιστού. Παρόλα αυτά, για το λόγο που δημιουργήθηκε ο συγκεκριμένος αλγόριθμος, την ανίχνευση δηλαδή της Αττικής Οδού, τα αποτελέσματά του είναι πολύ ικανοποιητικά.

7.3.6.4 Αλγόριθμος Tasseled Cap

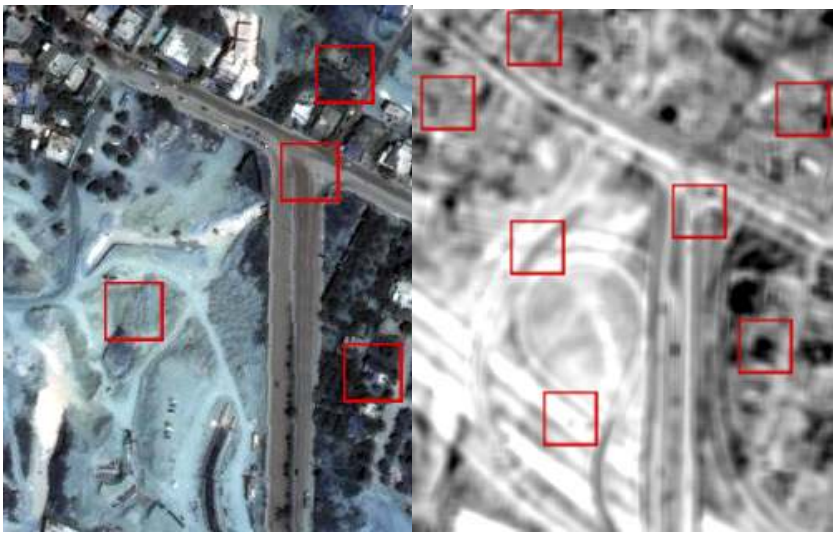
Το αποτέλεσμα του αλγορίθμου Tasseled Cap παρουσιάστηκαν στα παρακάτω σχήμα. (Σχήμα 7.25) Οι εικόνες που χρησιμοποιήθηκαν, ήταν αυτές που προέκυψαν από το πρώτο κανάλι του Tas-

seled Cap, καθώς το συγκεκριμένο κανάλι έδωσε τα καλύτερα αποτελέσματα.



(α΄) Αποτέλεσμα πρώτης περιοχής για το έτος 2000

(β΄) Αποτέλεσμα αλγορίθμου Tasseled Cap

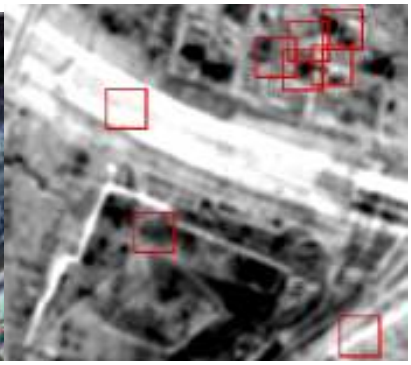


(γ΄) Αποτέλεσμα δεύτερης περιοχής για το έτος 2000

(δ΄) Αποτέλεσμα αλγορίθμου Tasseled Cap

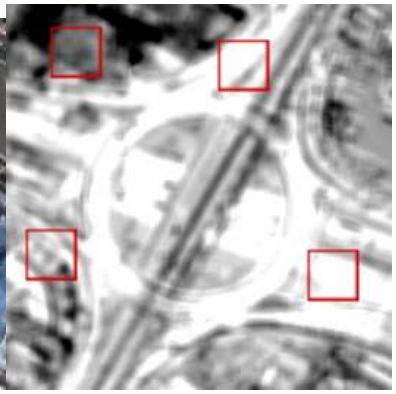
Σχήμα 7.25: Μεταβολές που προέκυψαν από τον αλγόριθμο Tasseled Cap

(ε′) Αποτέλεσμα τρίτης περιοχής για το έτος 2000



(στ′) Αποτέλεσμα αλγορίθμου Tasseled Cap

(ζ′) Αποτέλεσμα τέταρτης περιοχής για το έτος 2000



(η′) Αποτέλεσμα αλγορίθμου Tasseled Cap

Σχήμα 7.25: Μεταβολές που προέκυψαν από τον αλγόριθμο Tasseled Cap

Από το Σχήμα 7.25, προέκυψε ο παρακάτω πίνακας αξιολόγησης των αποτελεσμάτων που περιείχε τα σφάλματα συμπερίληψης και παράληψης.

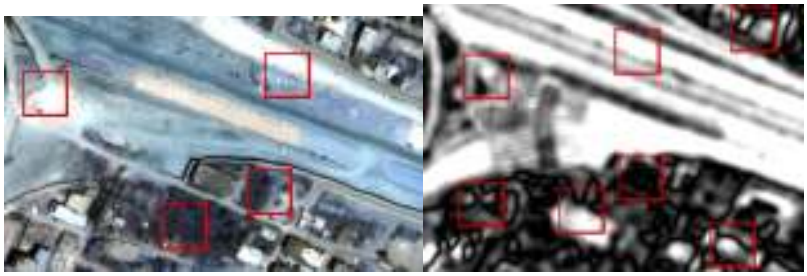
Αλγόριθμος / Φωτοερμηνεία	**Ανίχνευση μεταβολών**	**Μη ανίχνευση μεταβολών**
Μεταβολή	18/26 (69%)	6/26 (23%)
Μη μεταβολή	2/26 (8%)	–

Πίνακας 7.3: Πίνακας αξιολόγησης για αλγόριθμο Tasseled Cap

Έτσι λοιπόν, από τον παραπάνω πίνακα προκύπτει πως ο αλγόριθμος δίνει αποδεκτά αποτελέσματα. Το πρόβλημα του συγκεκριμένου αλγορίθμου είναι το ίδιο με το παραπάνω. Παρόλο που οι μεταβολές κατά μήκος της Αττικής Οδού ανιχνεύονται με πολύ καλό τρόπο, ο αστικός ιστός δεν παρουσιάζει πολύ καλά αποτελέσματα όσων αφορά την ανίχνευση των μεταβολών του. Το γεγονός αυτό έχει ως αποτέλεσμα να πέφτουν λίγο τα ποσοστά του αλγορίθμου.

7.3.6.5 Αλγόριθμος Διαχωρισμού Χρωμάτων

Το αποτέλεσμα του αλγορίθμου Διαχωρισμού Χρωμάτων παρουσιάζονται στα παρακάτω σχήμα. (Σχήμα 7.26) Όλη η διαδικασία της αξιολόγησης, έγινε στα αποτελέσματα που προέκυψαν από το μπλε κανάλι του μετασχηματισμού.



(α΄) Αποτέλεσμα πρώτης περιοχής για το έτος 2000

(β΄) Αποτέλεσμα αλγορίθμου Διαχωρισμού Χρωμάτων

(γ΄) Αποτέλεσμα δεύτερης περιοχής για το έτος 2000

(δ΄) Αποτέλεσμα αλγορίθμου Διαχωρισμού Χρωμάτων

(ε΄) Αποτέλεσμα τρίτης περιοχής για το έτος 2000

(ϛ΄) Αποτέλεσμα αλγορίθμου Διαχωρισμού Χρωμάτων

Σχήμα 7.26: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαχωρισμού Χρωμάτων

(ζ΄) Αποτέλεσμα τεταρτης περιοχής για το έτος 2000 (η΄) Αποτέλεσμα αλγορίθμου Διαχωρισμού Χρωμάτων

Σχήμα 7.26: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαχωρισμού Χρωμάτων

Από το Σχήμα 7.26, προέκυψε ο παρακάτω πίνακας αξιολόγησης των αποτελεσμάτων που περιείχε τα σφάλματα συμπερίληψης και παράληψης.

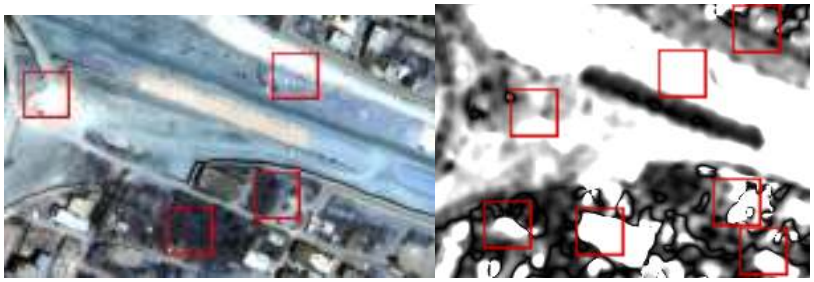
Αλγόριθμος / Φωτοερμηνεία	**Ανίχνευση μεταβολών**	**Μη ανίχνευση μεταβολών**
Μεταβολή	20/26 (77%)	4/26 (15%)
Μη μεταβολή	2/26 (8%)	–

Πίνακας 7.4: Πίνακας αξιολόγησης για αλγόριθμο Διαχωρισμού Χρωμάτων

Έτσι λοιπόν, από τον παραπάνω πίνακα προέκυψε πως ο αλγόριθμος έδωσε πολύ πάρα καλά αποτελέσματα. Επίσης, με την χρήση του συγκεκριμένου αλγορίθμου, οι μεταβολές στον αστικό ιστό ανιχνεύτηκαν πολύ καλύτερα.

7.3.6.6 Αλγόριθμος Διαφορών Κλίσεων

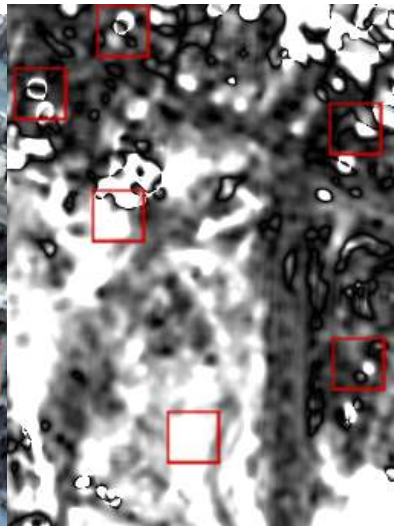
Το αποτέλεσμα του αλγορίθμου Διαφορών Κλίσεων παρουσιάστηκαν στα παρακάτω σχήμα. (Σχήμα 7.27) Η αξιολόγηση πραγματοποιήθηκε στα αποτελέσματα της διαφοράς του κόκκινου με το πράσινο κανάλι, καθώς παρουσίαζαν την καλύτερη ανίχνευση των μεταβολών στην εικόνα.



(α΄) Αποτέλεσμα πρώτης περιοχής για το έτος 2000 (β΄) Αποτέλεσμα αλγορίθμου Διαφορών Κλίσεων

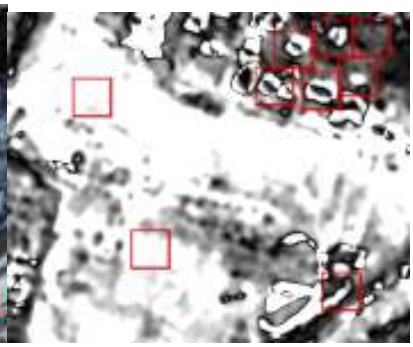
Σχήμα 7.27: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαφορών Κλίσεων

(γ′) Αποτέλεσμα δεύτερης περιοχής για το έτος 2000



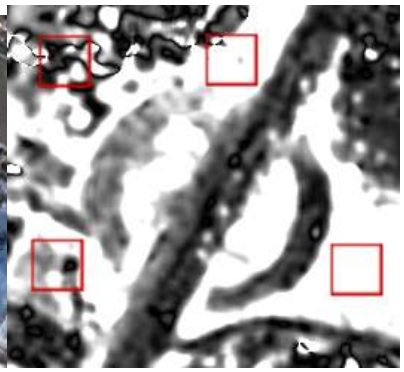
(δ′) Αποτέλεσμα αλγορίθμου Διαφορών Κλίσεων

(ε′) Αποτέλεσμα τρίτης περιοχής για το έτος 2000



(στ′) Αποτέλεσμα αλγορίθμου Διαφορών Κλίσεων

(ζ′) Αποτέλεσμα τεταρτης περιοχής για το έτος 2000



(η′) Αποτέλεσμα αλγορίθμου Διαφορών Κλίσεων

Σχήμα 7.27: Μεταβολές που προέκυψαν από τον αλγόριθμο Διαφορών Κλίσεων

Από το Σχήμα 7.27, προέκυψε ο παρακάτω πίνακας αξιολόγησης των αποτελεσμάτων που περιείχε τα σφάλματα συμπερίληψης και παράληψης.

Αλγόριθμος / Φωτοερμηνεία	**Ανίχνευση μεταβολών**	**Μη ανίχνευση μεταβολών**
Μεταβολή	15/26 (58%)	2/26 (8%)
Μη μεταβολή	9/26 (34%)	–

Πίνακας 7.5: Πίνακας αξιολόγησης για αλγόριθμο Διαφορών Κλίσεων

Έτσι λοιπόν, από τον παραπάνω πίνακα προκύπτει πως ο αλγόριθμος παρουσιάζει το μεγαλύτερο ποσοστό σφαλμάτων που ανιχνεύτηκαν σαν αλλαγή ενώ στην πραγματικότητα δεν ήταν. Αυτό συμβαίνει καθώς, ο αλγόριθμος ήταν πολύ ευαίσθητος στην ανίχνευση οποιαδήποτε μεταβολής, έστω και αν αυτή αναφερόταν σε σφάλμα γεωμετρικής ή ραδιομετρικής διόρθωσης.

7.3.6.7 Γενικά συμπεράσματα αξιολόγησης

Στο σημείο αυτό κρίθηκε σκόπιμο να γίνει μια ανακεφαλαίωση των αποτελεσμάτων που προέκυψαν από τους αλγορίθμους που δημιουργήθηκαν. Έτσι λοιπόν, αξιολογώντας τα αποτελέσματα που προέκυψαν από τους παραπάνω πίνακες και με βάση την φωτοερμηνεία που έγινε στην εικόνα προέκυψαν τα εξής:

- Η καλύτερη μέθοδος ανίχνευσης μεταβολών που δημιουργήθηκε ήταν αυτή του Διαχωρισμού Χρωμάτων, καθώς παρουσίασε τα καλύτερα ποσοστά ανίχνευσης σωστών μεταβολών, ενώ παράλληλα, τα σφάλματα παράληψης και συμπερίληψης, ήταν στο μικρότερο ποσοστό τους.

- Η χειρότερη μέθοδος που δημιουργήθηκε ήταν αυτή των Διαφορών Κλίσεων. Η μέθοδος ήταν πολύ ευαίσθητη στις φασματικές μεταβολές με αποτέλεσμα, το σφάλμα συμπερίληψης να είναι αρκετά μεγάλο.

- Όλες οι μέθοδοι δίνουν ικανοποιητικά αποτελέσματα τόσο σε ανίχνευση μεταβολών σε αστικό ιστό όσο και σε κόμβους.

- Οι μέθοδοι του Διαχωρισμού Χρωμάτων και Διαφορών Κλίσεων, παρουσιάζουν μεγάλο κορεσμό στις ψηφιακές τιμές καθώς ο αλγόριθμος του λόγου είναι πιο ευαίσθητος από ότι της διαφοράς.

- Το σφάλμα συμπερίληψης είναι πολύ χαμηλά σε ποσοστά, κάτι που υποδηλώνει αρκετά καλή ραδιομετρική και γεωμετρική διόρθωση.

- Η μέθοδος αξιολόγησης έγινε σε επίπεδο αντικειμένων και όχι σε επίπεδο εικονοστοιχείων, καθώς σε εικόνες πολύ υψηλής ανάλυσης με μεγάλες αλλαγές, όπως είναι αυτές που χρησιμοποιούνται στην συγκεκριμένη περίπτωση, αλλοιώνονται αριθμητικά οι ακρίβειες εξαιτίας του μεγάλου αριθμού φατνίων που καταλαμβάνουν.

- Τα σφάλματα παράληψης, είναι σχετικά υψηλά σε κάποιες μεθόδους, εξαιτίας της ομαλοποίησης που πραγματοποιείται με την χρήση του μέσου όρου σε υποπεριοχές της εικόνας.

Κεφάλαιο 8

Συμπεράσματα

8.1 Συμπεράσματα

Μετά το πέρας της συγκεκριμένης διπλωματικής εργασίας, θεωρείται ότι υλοποιήθηκε ο κύριος στόχος της, δηλαδή η παρουσίαση στους χρήστες ενός καινούργιου εργαλείου τηλεπισκόπησης, ελεύθερου κώδικα και η συνεισφορά σε αυτό κώδικα έτσι ώστε να συνεχιστεί η βελτίωσή του. Όπως προέκυψε από τις επιμέρους εφαρμογές, τόσο στο γραφικό περιβάλλον της πλατφόρμας (monteverdi), όσο και με εκτέλεση του ίδιου του κώδικά του, το συγκεκριμένο εργαλείο περιέχει πληθώρα αλγορίθμων και εφαρμογών που παρέχονται ελεύθερα στον οποιοδήποτε χρήστη. Τα αποτελέσματα των εφαρμογών, ήταν αντάξια των αποτελεσμάτων άλλων κλειστών λογισμικών τηλεπισκόπησης.

Μέσω της εργασίας αυτής, ο χρήστης, μπόρεσε πολύ εύκολα να καταλάβει τον τρόπο με τον οποίο λειτουργεί και δομείται η πλατφόρμα και να εφαρμόσει τους δικούς του αλγορίθμους σε αυτή. Τα παραδείγματα που παρουσιάστηκαν προκειμένου να εισάγουν τον χρήστη στο περιβάλλον του orfeo ήταν βασικές εφαρμογές τηλεπισκόπησης, τις οποίες οποιοσδήποτε αρχάριος χρήστης θα μπορούσε να χρειαστεί προκειμένου να εξάγει πληροφορίες από διαθέσιμες πολυφασματικές εικόνες. Έγινε μια προσπάθεια λοιπόν, δημιουργίας ενός εύχρηστου και πρακτικού εγχειριδίου, στο οποίο να απευθύνεται ο χρήστης για οποιαδήποτε απορία και ανάγκη του. Έτσι και αλλιώς η φιλοσοφία δόμησης του κώδικα στην συγκεκριμένη πλατφόρμα είναι παρόμοια, απλά προσαρμόζεται ανάλογα με την εφαρμογή.

Στην συνέχεια, έγινε μια προσπάθεια ανάλυσης του τρόπου με τον οποίο το εργαλείο πραγματοποιεί ανίχνευση μεταβολών σε ζεύγος εικόνων. Οι αλγόριθμοι που αντιστοιχούσαν στην συγκεκριμένη εφαρμογή, παρουσιάστηκαν με λεπτομέρεια, ενώ τα αποτελέσματά τους ήταν ικανοποιητικά, αν και σε κάποιες περιπτώσεις δεν ενδεικνύονταν για την συγκεκριμένη εφαρμογή, δηλαδή την ανίχνευση της μεταβολής στην περιοχή της Αττικής Οδού. Σε κάθε περίπτωση όμως, η χρήση τους παρείχε στον χρήστη μια τελική εικόνα που παρουσίαζε επιτυχώς μεταβολές, είτε μεγαλύτερες, είτε μικρότερες των δύο εικόνων. Έτσι και αλλιώς, κάθε αλγόριθμος ανίχνευσης μεταβολών, παρέχει τα δικά του αποτελέσματα καθώς είναι ευαίσθητος σε ξεχωριστά κομμάτια πληροφορίας πάνω στην εικόνα.

Έπειτα, έγινε μια προσπάθεια, εμπλουτισμού του κώδικα που ήδη υπήρχε μέσα στο εργαλείο, για ανίχνευση μεταβολών. Η συγκεκριμένη διαδικασία πραγματοποιήθηκε σε δύο ξεχωριστά στάδια. Το πρώτο στάδιο, αποτελείτο από την δημιουργία εκτελέσιμων αρχείων που μετασχημάτιζαν την αρχική πολυκάναλη εικόνα, σε μια σειρά εικόνων ή σε μια μονοκάναλη εικόνα, παρέχοντας βελτιστοποίηση πληροφοριών για αυτή. Με άλλα λόγια, δημιουργήθηκαν μετασχηματισμοί που χρησιμοποιούσαν όλα ή συνδυασμό καναλιών από την αρχική εικόνα, και παρείχαν ένα ή περισσότερα αποτελέσματα τα οποία τόνιζαν ανά εφαρμογή ξεχωριστά χα-

ρακτηριστικά της. Στην συνέχεια, η εικόνες αυτές ανά ζεύγη εποχών εισάγονταν σε δύο ξεχωριστές κλάσεις που πραγματοποιούσαν κανονικοποιημένο λόγο ή διαφορά και παρήγαγαν το τελικό αποτέλεσμα. Οι κλάσεις αυτές δημιουργήθηκαν με βάση τις υπάρχουσες προδιαγραφές της πλατφόρμας, έτσι ώστε να μπορέσουν να ενσωματωθούν σε αυτή. Τα αποτελέσματα των κλάσεων αυτών είναι πολύ ικανοποιητικά και κατάλληλα για χρήση.

Αξιολογώντας λοιπόν, τα παρεχόμενα αποτελέσματα, η ανάπτυξη των συγκεκριμένων μεθόδων ανίχνευσης μεταβολών, θεωρείται επιτυχής. Στις περισσότερες περιπτώσεις, τα αποτελέσματα ήταν ισάξια των μεθόδων που ήδη υπήρχαν μέσα στο εργαλείο. Το γεγονός μάλιστα πως οι εικόνες που εισάγονταν σε αυτά ήταν πολύ μεγάλων διαστάσεων και το κόστος εκτέλεσης μικρό, δίνει ένα επιπλέον βάρος στην θετική αξιολόγησή τους. Οι υπολογιστικοί πόροι μάλιστα που χρειάζονταν για να παράγουν το τελικό αποτέλεσμα ήταν ελάχιστοι, ενώ το αποτέλεσμα ολοκληρωνόταν σύντομα. Μάλιστα, για τις εικόνες που χρησιμοποιήθηκαν στην συγκεκριμένη διπλωματική εργασία, οι διαστάσεις ήταν 5407x4379 και διέθεταν 4 κανάλια, ενώ χειρίστηκαν ως πραγματικοί αριθμοί διπλής ακρίβειας, οπότε η μνήμη που χρειάστηκαν ήταν 723 MB. Παράλληλα, η εικόνα που παράχθηκε από το πρόγραμμα ήταν των ίδιων διαστάσεων, αλλά μονοκάναλη, οπότε η μνήμη που χρειάστηκε ήταν 181 MB. Μάλιστα, οι αλγόριθμοι χρειάστηκαν λιγότερο από 1 λεπτά για να υλοποιηθούν σε τετραπύρηνο υπολογιστή. Το μέγεθος των εικόνων, και ο χρόνος που χρειάστηκε για την επεξεργασία τους, υποδηλώνει την αποδοτικότητα των αλγορίθμων.

8.2 Προοπτικές

Το γεγονός ότι για όλη την διαδικασία, χρησιμοποιήθηκε ελεύθερο λογισμικό, αφήνει ανοιχτό το ενδεχόμενο οποιασδήποτε βελτίωσης των παρεχόμενων αλγορίθμων από οποιοδήποτε χρήστη. Έτσι και αλλιώς, εκτελέσιμα προγράμματα που ήδη υπήρχαν στο εργαλείο, βελτιώθηκαν ή τροποποιήθηκαν, προκειμένου να εξυπηρετήσουν τους ανάλογους σκοπούς. Γενικότερα, κάποιες προοπτικές για περαιτέρω ενασχόληση με το συγκεκριμένο αντικείμενο στο μέλλον θα μπορούσαν να είναι οι εξής:

- Αρχικά, τα δύο ξεχωριστά τμήματα, στα οποία έχουν δημιουργηθεί παραπάνω αλγόριθμοι, θα μπορούσαν να ομαδοποιηθούν σε ένα, δημιουργώντας την αντίστοιχη κλάση, ή και πρότυπο, έτσι ώστε να μην χρειάζεται ο χρήστης να εκτελεί χειροκίνητα το κάθε κομμάτι. Στην συγκεκριμένη εργασία, θεωρήθηκε πως ο τελικός αλγόριθμος ανίχνευσης μεταβολών, δεν χρειαζόταν να εφαρμοστεί σε ολόκληρη την πολυκάναλη εικόνα, κάνοντας πιο περίπλοκη την όλη διαδικασία, αλλά η εισαγωγή σε αυτόν μιας επεξεργασμένης μονοκάναλης εικόνας θα έδινε ικανοποιητικά αποτελέσματα. Σε κάθε περίπτωση όμως, η ακόμα **μεγαλύτερη αυτοματοποίηση** των αλγορίθμων, θεωρείται ένας μελλοντικός στόχος.

- Ο εμπλουτισμός της πλατφόρμας με ακόμα περισσότερους αλγορίθμους, που θα αφορούν όχι μόνο την ανίχνευση μεταβολών, αλλά και οποιαδήποτε άλλη εφαρμογή. Από την στιγμή που έγινε κατανοητή η φιλοσοφία δόμησης κλάσεων και προτύπων στο περιβάλλον του orfeo, οποιαδήποτε εφαρμογή θα μπορούσε με ανάλογη προσέγγιση να φέρει τα ίδια αποτελέσματα. Έτσι και αλλιώς αυτός είναι και ο απώτερος στόχος ολόκληρης της πλατφόρμας. Ο συνεχής δηλαδή, **εμπλουτισμός** της και **βελτίωσή** της από τους επιμέρους χρήστες της με νέους αλγορίθμους.

- **Εισαγωγή στο γραφικό περιβάλλον του εργαλείου**, Monteverdi των αλγορίθμων που δημιουργήθηκαν, έτσι ώστε η χρήση τους να είναι εύκολη για όλους τους χρήστες της βιβλιοθήκης.

- Τέλος, η δημιουργία του συγκεκριμένου εγχειριδίου αναφέρεται στην έκδοση που βρίσκεται σήμερα η πλατφόρμα. Το γεγονός ότι ανά τακτά χρονικά διαστήματα υπάρχει και καινούργια έκδοση βελτιωμένη και με περισσότερες εφαρμογές, καθιστά απαραίτητη την συνεχής **βελτίωση του εγχειριδίου** που παρέχεται στα πλαίσια της συγκεκριμένης εργασίας.

Βιβλιογραφία

[1] Addison-Wesley *Effective C++ Third Edition*,Addison - Wesley Professional Computing Series, 2005

[2] Argialas Demetre and Derzekos Panos *Mapping Urban Green from IKONOS Data by an Object-Oriented Knowledge-base and Fuzzy Logic*

[3] Argialas, D.P. and Harlow D. A. *Computational Image Interpretation ModelsQ An Overview and a Perspective*, American Society for Photogrammetry and Remote Sensing, 1990, p.p. 871-886

[4] Argialas D.P. and Krishanamurthy S. *Detection of lines and circles in maps and engineering drawings*, p.p 392-399

[5] Argialas D.P. and Mavrantza O.D. *Comparison of edge detection and Hough transform techniques for the extraction of geologic features*, Laboratory of Remote Sensing, School of Rural and Surveying Engineering, National Technical University of Athens

[6] Argialas D.P. and Mavrantza O.D. *Edge Detection Techniques For Extracting Linear Information In An Urban/Peri-Urban Environment*, Spatial Information Management toward Legalizing Informal Urban Development, Athens, Greece, March 28-31, 2007

[7] Christophe Emmanuel, Member, IEEM, Julien Michel, and Jordi Inglada, Member, IEEE *Remote Sensing Processing: From Multicore to GPU*, IEEE Journal of selected topics in applied earth observations and remote sensing, August 2011

[8] Christophe Emmanuel, Inglada Jordi *Open Source Remote Sensing: Increasing the Usability of Cutting- Edge Algoriths*, IEEE Geoscience and Remote Sensing Society Newsletter, March 2009

[9] Christophe Emmanuel, Inglada Jordi, Giros Alain *Orfeo ToolBox A complete solution for mapping from High Resolution Satellite images*, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B4. Beijing 2008, p.p 1263-1268

[10] Christophe Emmanuel, Inglada Jordi, Julien Michel *Remote Sensing ProcessingQ From Multicore to GPU*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2011

[11] Deitel H.M., Deitel P.J., *C++programming*, Εκδότης Μ. Γκιούρδας,, p.1319 2003

[12] Grizonnet Manuel and Inglada Jordi *Monteverdi - Remote sensing software from educational to operational context*, 30th EARSeL Symposium : Remote Sensing for Science, Education and Culture, Paris, France, May 2010

[13] Haverkamp Donna and Poulsen Rick *Change Detection Using Ikonos IMagery*, Departement of Research and Product Development

[14] Horne, J. *A Tasseled Cap Transformation for IKONOS Images* ASPRS 2003 Annual Conference Proceedings, May 2003, Anchorage, Alaska, pp. 34-43

[15] Hyvarinen Aapo *Survey on Independent Component Analysis*, Neural Computing Surveys, Vol2, pp. 94-128, 1999

[16] Hyvarinen Aapo *Estimation of Non-Normalized Statistical Models by Score Matching*, Journal of Machine Learning Research, 2005

[17] Hyvarinen Aapo and Erkki Oja *Independent Component Analysis: Algorithms and Applications*, 2000

[18] *Imagine Delta Cue Users Guide* Leica Geosystems Geospatial Imaging LLC 2007

[19] Joseph George *Fyndamentals of Remote Sensing*

[20] Mercier Gregoire and Inglada Jordi *Change Detection with Misregistration Errors and Heterogeneous Data Through the Orfeo ToolBox*, Collection des rapports de recherche de TELECOM Bretagne, 2008

[21] Stroustrup Bjarne*C++*, Εκδόσεις Κλειδάριθμος, , p.1189, 1997

[22] Soulie Juan *C++ Language Tutorial*, June 2007

[23] OTB Development Team *The ORFEO ToolBox Software Guide Update for OTB-3.8* 2010.

[24] Αργιαλάς Δ.Π., Μαυραντζά Ο.Δ., Στεφούλη Μ. *Αυτόματη χαρτογράφηση τεκτονικών φωτογραμμώσεων (ρηγμάτων) με χρήση μεθόδων και τεχνικών Φωτοερμηνείας - Ψηφιακής Τηλεπισκόπισης και Έμπειρων Συστημάτων*, Σχολή Αγρονόμων και Τοπογράφων Μηχανικών - Ε.Μ.Π. και Ινστιτούτο Γεωλογικών και Μεταλλευτικών Ερευνών (Ι.Γ.Μ.Ε.)

[25] Αργιαλάς Δημήτρης Π. *Φωτοερμηνεία-Τηλεπισκόπηση* 1999

[26] Αργιαλάς Δημήτρης Π. *Ψηφιακή Τηλεπισκόπηση* 1998

[27] Καραλής Αθ. Αντώνιος *Διπλωματική Εργασία Διερεύνηση μεθόδων ανίχνευσης μεταβολών μέσω τηλεπισκοπικών δεδομένων Ikonos για την αναθεώρηση του οδικού δικτύου* 2008

[28] Ρόκος Δημήτρης Κλ. *Φωτοερμηνεία Τηλεπισκόπηση* 2005

Ηλεκτρονικές πηγές

[29] "`http://hg.orfeo-toolbox.org/OTB/summary`"

[30] "`http://blog.orfeo-toolbox.org/monteverdi`"

[31] "`http://www.orfeo-toolbox.org/otb/`"

[32] "`http://www.orfeo-toolbox.org/otb/monteverdi.html`"

[33] "`http://www.orfeo-toolbox.org/doxygen/`"

[34] "`http://www.itk.org/`"

[35] "`http://www.gdal.org/`"

[36] "http://www.fltk.org/"

[37] "http://www.cmake.org/"

[38] "http://en.wikipedia.org/wiki/K-means_clustering"

[39] "http://www.mywebsite.force9.co.uk/png/"

[40] "http://el.wikipedia.org/wiki/LIDAR"

[41] "http://www.nasa.gov/"

[42] "http://www.cnrhome.uidaho.edu/default.aspx?pid=68480"

[43] "http://www.ccrs.nrcan.gc.ca/glossary/index_e.php?id=68"

[44] "http://www.ltrs.uri.edu/nps_lulc/Wang-Article.pdf"

[45] "http://en.wikipedia.org/wiki/Entropy"

[46] "http://www.nationalacademies.org/rise/backg4a.htm"

[47] "http://www.ccis2k.org/iajit/PDF/vol.8,no.2/12-1093.pdf"

[48] "http://www.uwf.edu/gis/manuals/DeltaCue.pdf"

[49] "http://www.bme.columbia.edu/eekweb/journals/2002-eek_ieee_localized_harmonic_motion_imaging.pdf"

[50] "http://www.erdas.com/products/erdasimagine/imaginedeltacue/news/08-06-11/Geoinformation_SA_Providing_ERDAS_Solutions_to_Map_Greece_s_National_Forests.aspx"

[51] "http://en.wikipedia.org/wiki/Ground_truth"

Παράρτημα

Τα αρχεία που περιέχονται στο παράρτημα ακολουθούν την αρίθμηση των κεφαλαίων στα οποία αναφέρονται. Έτσι, σε περίπτωση που αναφέρονται στο πέμπτο κεφάλαιο θα ξεκινάνε με τον αριθμό 5 , έτσι ώστε ο χρήστης να μπορεί πολύ εύκολα να παρακολουθεί την ροή του κώδικα.

5.1 Παρουσίαση εικόνας

5.1.1 Αρχείο CMake

```
1  PROJECT (Tutorials)
2
3  cmake_minimum_required(VERSION 2.6)
4
5  FIND_PACKAGE(OTB)
6  IF(OTB_FOUND)
7    INCLUDE (${OTB_USE_FILE})
8  ELSE (OTB_FOUND)
9    MESSAGE (FATAL_ERROR
10      "Cannot build OTB project without OTB. Please set OTB_DIR.")
11 ENDIF (OTB_FOUND)
12
13 ADD_EXECUTABLE (pipeline pipeline.cxx)
14 TARGET_LINK_LIBRARIES (pipeline OTBCommon OTBIO)
```

5.1.2 Αρχείο Κώδικα

```
1  /*=========================================================================
2
3    Program:   ORFEO Toolbox
4    Language:  C++
5    Date:      $Date$
6    Version:   $Revision$
7
8
9    Copyright (c) Centre National d'Etudes Spatiales. All rights
       reserved.
10   See OTBCopyright.txt for details.
11
12
13       This software is distributed WITHOUT ANY WARRANTY; without
           even
14       the implied warranty of MERCHANTABILITY or FITNESS FOR A
           PARTICULAR
15       PURPOSE.  See the above copyright notices for more
           information.
16
17 =========================================================================
     */
18
19 //  Software Guide : BeginLatex
20 //
21 //  The following code is an implementation of a small OTB
22 //  program. It tests including header files and linking with
     OTB
23 //  libraries.
24 //
25 //  Software Guide : EndLatex
26
27 // Software Guide : BeginCodeSnippet
28
29 #include "otbImage.h"
30 #include "otbImageFileReader.h"
31 #include "otbStreamingImageFileWriter.h"
32 #include <iostream>
33
34 int main(int argc, char * argv[] )
35 {
36   typedef otb::Image<unsigned short, 2> ImageType;
37
38   typedef otb::ImageFileReader<ImageType> ReaderType;
39   ReaderType::Pointer reader = ReaderType::New();
40
41   typedef otb::StreamingImageFileWriter<ImageType> WriterType;
42   WriterType::Pointer writer= WriterType::New();
43
44   reader->SetFileName(argv[1]);
```

```
45    writer->SetFileName(argv[2]);
46
47    writer->SetInput(reader->GetOutput());
48    writer->Update();
49
50    return EXIT_SUCCESS;
51  }
```

5.2 Εξαγωγή ενός καναλιού από πολυκάναλη εικόνα

```
1  /*=========================================================================
2
3     Program:   ORFEO Toolbox
4     Language:  C++
5     Date:      $Date$
6     Version:   $Revision$
7
8
9     Copyright (c) Centre National d'Etudes Spatiales. All rights
         reserved.
10    See OTBCopyright.txt for details.
11
12
13       This software is distributed WITHOUT ANY WARRANTY; without
            even
14       the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
15       PURPOSE.  See the above copyright notices for more
            information.
16
17 =========================================================================*/
18
19 //  Software Guide : BeginCommandLineArgs
20 //    INPUTS: {axlada.tif}
21 //    OUTPUTS: {blue.tif}
22 //  Software Guide : EndCommandLineArgs
23
24
25 // Software Guide : BeginCodeSnippet
26 #include "otbImage.h"
27 #include "otbImageFileReader.h"
28 #include "otbVectorImage.h"
29 #include "otbStreamingImageFileWriter.h"
30 #include "otbMultiToMonoChannelExtractROI.h"
31
32  int main(int argc, char *argv[])
33  {
34
35    if (argc < 2)
36 {
37
38     std::cerr << "Usage: " << std::endl;
39     std::cerr << argv[0] << "  inputImageFile   outputImageFile"
          << std::endl;
40
41    return EXIT_FAILURE;
42
43  }
44
45    typedef unsigned short int              PixelType;
```

```
46      typedef otb::VectorImage<PixelType, 2> VectorImageType;
47
48      typedef otb::ImageFileReader<VectorImageType> ReaderType;
49      ReaderType::Pointer reader = ReaderType::New();
50
51      reader->SetFileName(argv[1]);
52
53      typedef otb::MultiToMonoChannelExtractROI<PixelType,
            PixelType>
54      ExtractChannelType;
55      ExtractChannelType::Pointer extractChannel =
            ExtractChannelType::New();
56
57      reader->UpdateOutputInformation();
58      extractChannel->SetExtractionRegion(
59      reader->GetOutput()->GetLargestPossibleRegion());
60
61      extractChannel->SetChannel(3);
62      extractChannel->SetInput(reader->GetOutput());
63
64
65      typedef otb::Image<PixelType, 2>                    ImageType;
66      typedef otb::StreamingImageFileWriter<ImageType> WriterType;
67      WriterType::Pointer writer = WriterType::New();
68
69      writer->SetFileName(argv[2]);
70      writer->SetInput(extractChannel->GetOutput());
71
72      writer->Update();
73
74      return EXIT_SUCCESS;
75  }
```

5.3 Δημιουργία RGB εικόνας από πολυκαναλη

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All
            rights reserved.
9       See OTBCopyright.txt for details.
10
11      Some parts of this code are derived from ITK. See
            ITKCopyright.txt
12      for details.
13
14      This software is distributed WITHOUT ANY WARRANTY; without
            even
15      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
16      PURPOSE.  See the above copyright notices for more
            information.
17      Modified by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
18  =========================================================================*/
19  //  Software Guide : BeginCommandLineArgs
20  //    INPUTS: {IMAGERY_SSECH.tif}
21  //    OUTPUTS: {ROI_IMAGERY_RGB.tif}
22  //  Software Guide : EndCommandLineArgs
23
24  #include "otbImageFileReader.h"
25  #include "otbImageFileWriter.h"
26  #include "otbStreamingImageFileWriter.h"
27  #include "otbMultiChannelExtractROI.h"
28  #include "otbVectorImage.h"
29
30  int main(int argc, char *argv[])
31  {
32
33     if (argc < 3)
34      {
35
36      std::cerr << "Usage: " << std::endl;
37      std::cerr << argv[0] <<
38       "  inputImageFile outputImageFileRGB"
39       << std::endl;
40
41      return EXIT_FAILURE;
42      }
43
44      const char *inputFilename  = argv[1];
```

```
45      const char *outputFilenameRGB = argv[2];
46
47    typedef unsigned short int                PixelType;
48    typedef otb::VectorImage<PixelType, 2> VectorImageType;
49
50    typedef otb::ImageFileReader<VectorImageType> ReaderType;
51    ReaderType::Pointer reader = ReaderType::New();
52
53
54    typedef otb::MultiChannelExtractROI<PixelType,
55            PixelType>  ExtractROIFilterType;
56   ExtractROIFilterType::Pointer extractROIFilter =
       ExtractROIFilterType::New();
57
58    typedef otb::StreamingImageFileWriter<VectorImageType>
        WriterType;
59    WriterType::Pointer writer = WriterType::New();
60
61    reader->SetFileName(inputFilename);
62    reader->UpdateOutputInformation();
63    writer->SetFileName(outputFilenameRGB);
64
65
66     extractROIFilter->SetExtractionRegion(
67     reader->GetOutput()->GetLargestPossibleRegion());
68
69
70     extractROIFilter->SetFirstChannel(1);
71     extractROIFilter->SetLastChannel(3);
72
73
74     extractROIFilter->SetInput(reader->GetOutput());
75     writer->SetInput(extractROIFilter->GetOutput());
76
77     writer->Update();
78
79     return EXIT_SUCCESS;
80 }
```

5.4 Δημιουργία κυρίων συνιστωσών

```
1  /*=========================================================================
2
3     Program:   ORFEO Toolbox
4     Language:  C++
5     Date:      $Date$
6     Version:   $Revision$
7
8
9     Copyright (c) Centre National d'Etudes Spatiales. All rights
          reserved.
10    See OTBCopyright.txt for details.
11
12
13    This software is distributed WITHOUT ANY WARRANTY; without
          even
14    the implied warranty of MERCHANTABILITY or FITNESS FOR A
          PARTICULAR
15    PURPOSE.  See the above copyright notices for more
          information.
16
17  =========================================================================*/
18  #include "otbImage.h"
19  #include "otbVectorImage.h"
20  #include "otbImageFileReader.h"
21  #include "otbImageFileWriter.h"
22  #include "otbMultiToMonoChannelExtractROI.h"
23  #include "itkRescaleIntensityImageFilter.h"
24  #include "otbInnerProductPCAImageFilter.h"
25
26  //  Software Guide : BeginCommandLineArgs
27  //    INPUTS: {ROI_QB_MUL_1.png}
28  //    OUTPUTS: {InnerProductPCAOutput.tif}, {
        PrettyInnerProductPCAOutput1.png}, {
        PrettyInnerProductPCAOutput2.png}, {
        PrettyInnerProductPCAOutput3.png}
29  //    3
30
31
32  int main(int argc, char *argv[])
33  {
34    typedef double PixelType;
35    const unsigned int Dimension = 2;
36    const char  *inputFileName = argv[1];
37    const char  *outputFilename = argv[2];
38    const unsigned int numberOfPrincipalComponentsRequired(atoi(
          argv[6]));
39
40    typedef otb::VectorImage<PixelType, Dimension> ImageType;
41    typedef otb::ImageFileReader<ImageType>          ReaderType;
42    typedef otb::ImageFileWriter<ImageType>          WriterType;
```

```
43
44      ReaderType::Pointer reader      = ReaderType::New();
45      reader->SetFileName(inputFileName);
46
47      typedef otb::InnerProductPCAImageFilter<ImageType, ImageType>
            PCAFilterType;
48      PCAFilterType::Pointer pcafilter      = PCAFilterType::New();
49
50      pcafilter->SetNumberOfPrincipalComponentsRequired(
51        numberOfPrincipalComponentsRequired);
52
53
54      WriterType::Pointer writer      = WriterType::New();
55      writer->SetFileName(outputFilename);
56
57      pcafilter->SetInput(reader->GetOutput());
58      writer->SetInput(pcafilter->GetOutput());
59
60      writer->Update();
61
62  //Extraction of the PCA and rescale the final images
63
64      typedef otb::Image<PixelType, Dimension> MonoImageType;
65
66      typedef otb::MultiToMonoChannelExtractROI<PixelType,
            PixelType>
67      ExtractROIFilterType;
68
69      typedef otb::Image<unsigned char,2>
70                              OutputImageType;
71      typedef otb::ImageFileWriter<OutputImageType>
72      WriterType2;
73      typedef itk::RescaleIntensityImageFilter<MonoImageType,
74           OutputImageType> RescalerType;
75
76      for (unsigned int cpt = 0; cpt <
            numberOfPrincipalComponentsRequired; cpt++)
77        {
78        ExtractROIFilterType::Pointer extractROIFilter =
             ExtractROIFilterType::New();
79        RescalerType::Pointer          rescaler = RescalerType::New
             ();
80
81        WriterType2::Pointer writer2      = WriterType2::New();
82        extractROIFilter->SetInput(pcafilter->GetOutput());
83        extractROIFilter->SetChannel(cpt + 1);
84
85        rescaler->SetInput(extractROIFilter->GetOutput());
86        rescaler->SetOutputMinimum(0);
87        rescaler->SetOutputMaximum(255);
88
```

```
89          writer2->SetInput(rescaler->GetOutput());
90          writer2->SetFileName(argv[cpt + 3]);
91          writer2->Update();
92          }
93
94      return EXIT_SUCCESS;
95  }
```

5.5 Αρχείο Κώδικα για λόγους

```
1  /*=========================================================================
2  Program:   ORFEO Toolbox
3  Language:  C++
4  Date:      $Date$
5  Version:   $Revision$
6
7  Copyright (c) Centre National d'Etudes Spatiales. All rights
       reserved.
8  See OTBCopyright.txt for details.
9
10 This software is distributed WITHOUT ANY WARRANTY; without even
11 the implied warranty of MERCHANTABILITY or FITNESS FOR A
       PARTICULAR
12 PURPOSE.  See the above copyright notices for more information.
13 =========================================================================*/
14
15 //  Software Guide : BeginCommandLineArgs
16 //    INPUTS: {qb_RoadExtract.tif}
17 //    OUTPUTS: {RoadExtractBandMath.tif}, {qb_BandMath-pretty.
       jpg}
18 //  Software Guide : EndCommandLineArgs
19
20 #include "itkExceptionObject.h"
21 #include <iostream>
22
23 #include "otbImage.h"
24 #include "otbVectorImage.h"
25 #include "otbImageFileReader.h"
26 #include "otbImageFileWriter.h"
27 #include "itkCastImageFilter.h"
28 #include "otbVectorImageToImageListFilter.h"
29 #include "otbBandMathImageFilter.h"
30
31 int main( int argc, char *argv[])
32 {
33
34    if (argc != 4)
35    {
36       std::cerr << "Usage: " << argv[0] << " inputImageFile ";
37       std::cerr << " outputImageFile ";
38       std::cerr << " outputPrettyImageFile" << std::endl;
39       return EXIT_FAILURE;
40     }
41
42    typedef double                                    PixelType;
43    typedef otb::Image<PixelType, 2>                  OutputImageType;
44    typedef otb::ImageList<OutputImageType>           ImageListType;
45    typedef OutputImageType::PixelType                VPixelType;
46    typedef otb::VectorImageToImageListFilter<InputImageType,
          ImageListType>
```

```
47                                          VectorImageToImageListType;
48    typedef otb::ImageFileReader<InputImageType>     ReaderType;
49    typedef otb::ImageFileWriter<OutputImageType>    WriterType;
50
51    typedef otb::BandMathImageFilter<OutputImageType>
          FilterType;
52
53    ReaderType::Pointer reader = ReaderType::New();
54    WriterType::Pointer writer = WriterType::New();
55
56    FilterType::Pointer filter = FilterType::New();
57    writer->SetInput(filter->GetOutput());
58    reader->SetFileName(argv[1]);
59    writer->SetFileName(argv[2]);
60
61
62    reader->UpdateOutputInformation();
63
64    VectorImageToImageListType::Pointer imageList =
          VectorImageToImageListType::New();
65    imageList->SetInput(reader->GetOutput());
66
67    imageList->UpdateOutputInformation();
68
69    const unsigned int nbBands = reader->GetOutput()->
          GetNumberOfComponentsPerPixel();
70    for(unsigned int j = 0; j < nbBands; ++j)
71    {
72
73    filter->SetNthInput(j, imageList->GetOutput()->GetNthElement(
          j));
74
75    }
76
77    filter->SetExpression("if(ndvi(b3,␣b4)>0.4,␣255,␣0)");
78
79    writer->Update();
80
81   typedef otb::Image<unsigned char, 2>
         OutputPrettyImageType;
82   typedef otb::ImageFileWriter<OutputPrettyImageType>
         PrettyImageFileWriterType;
83   typedef itk::CastImageFilter<OutputImageType,
         OutputPrettyImageType> CastImageFilterType;
84
85    PrettyImageFileWriterType::Pointer prettyWriter =
          PrettyImageFileWriterType::New();
86    CastImageFilterType::Pointer caster = CastImageFilterType::
          New();
87    caster->SetInput(filter->GetOutput());
88
```

```
89      prettyWriter->SetInput(caster->GetOutput());
90      prettyWriter->SetFileName(argv[3]);
91      prettyWriter->Update();
92      return EXIT_SUCCESS;
93  }
```

Αρχείο CMake για λόγους με viewer

```
PROJECT (Tutorials)

cmake_minimum_required(VERSION 2.6)

FIND_PACKAGE(OTB)
IF(OTB_FOUND)
  INCLUDE (${OTB_USE_FILE})
ELSE (OTB_FOUND)
  MESSAGE (FATAL_ERROR
     "Cannot build OTB project without OTB. Please set OTB_DIR.")
ENDIF (OTB_FOUND)

ADD_EXECUTABLE (VisuExample1 VisuExample1.cxx)
 TARGET_LINK_LIBRARIES(VisuExample1 OTBVisualization OTBGui OTBIO
OTBCommon ${OTB_VISU_GUI_LIBRARIES})
```

5.6 Αρχείο Κώδικα για λόγους με viewer

```
/*=========================================================================
Program:   ORFEO Toolbox
Language:  C++
Date:      $Date$
Version:   $Revision$

Copyright (c) Centre National d'Etudes Spatiales. All rights reserved.
See OTBCopyright.txt for details.

This software is distributed WITHOUT ANY WARRANTY; without even
the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.  See the above copyright notices for more information.

 //  Modified by Maria Vakalopoulou Sep 2011 mariavakalopoulou@gmail.com
=========================================================================*/
//  Software Guide : BeginCommandLineArgs
//    INPUTS: {qb_RoadExtract.tif}
//    OUTPUTS: {RoadExtractBandMath.tif}, {qb_BandMath-pretty.jpg}
//  Software Guide : EndCommandLineArgs
//  Software Guide : BeginLatex

#include "itkExceptionObject.h"
#include <iostream>
 #include <stdlib.h>

#include "otbImage.h"
#include "otbVectorImage.h"
#include "otbImageFileReader.h"
#include "otbImageFileWriter.h"
#include "otbVectorImageToImageListFilter.h"
#include "otbStandardImageViewer.h"
#include "otbBandMathImageFilter.h"

int main( int argc, char *argv[])
{

   if (argc != 3)
   {
      std::cerr << "Usage: " << argv[0] << " inputImageFile ";
      std::cerr << " outputImageFile " << std::endl;
      return EXIT_FAILURE;
    }

   typedef double                                PixelType;
   typedef otb::VectorImage<PixelType, 2>
      InputImageType;
```

```
46      typedef otb::Image<PixelType, 2>
            OutputImageType;
47      typedef otb::ImageList<OutputImageType>
            ImageListType;
48      typedef OutputImageType::PixelType                  VPixelType;
49      typedef otb::VectorImageToImageListFilter<InputImageType,
            ImageListType>
50                      VectorImageToImageListType;
51      typedef otb::ImageFileReader<InputImageType>        ReaderType
            ;
52      typedef otb::ImageFileWriter<OutputImageType>       WriterType
            ;
53
54      typedef otb::BandMathImageFilter<OutputImageType>
            FilterType;
55
56      ReaderType::Pointer reader = ReaderType::New();
57      WriterType::Pointer writer = WriterType::New();
58
59      FilterType::Pointer filter = FilterType::New();
60      writer->SetInput(filter->GetOutput());
61      reader->SetFileName(argv[1]);
62      writer->SetFileName(argv[2]);
63
64      reader->UpdateOutputInformation();
65
66      VectorImageToImageListType::Pointer imageList =
            VectorImageToImageListType::New();
67      imageList->SetInput(reader->GetOutput());
68
69      imageList->UpdateOutputInformation();
70
71      const unsigned int nbBands = reader->GetOutput()->
            GetNumberOfComponentsPerPixel();
72      for(unsigned int j = 0; j < nbBands; ++j)
73      {
74
75      filter->SetNthInput(j, imageList->GetOutput()->GetNthElement(
            j));
76
77      }
78
79
80      filter->SetExpression("if(ndvi(b3,␣b4)>0.4,␣255,␣0)");
81
82      writer->Update();
83
84    const char *inputFilename = argv[2];
85
86       typedef otb::VectorImage<PixelType, 2>        ImageType;
87       typedef otb::StandardImageViewer<ImageType> ViewerType;
```

```
88
89      ViewerType::Pointer lViewer = ViewerType::New();
90      ReaderType::Pointer lReader = ReaderType::New();
91      lReader->SetFileName(inputFilename);
92      lReader->UpdateOutputInformation();
93
94      lViewer->SetLabel("My Image");
95
96      lViewer->SetImage(lReader->GetOutput());
97      lViewer->Update();
98
99    Fl::run();
100
101   return EXIT_SUCCESS;
102 }
```

5.7 Αρχείο Κώδικα για φίλτρο μέσου όρου

```
1  /*=========================================================================
2
3     Program:   ORFEO Toolbox
4     Language:  C++
5     Date:      $Date$
6     Version:   $Revision$
7
8
9     Copyright (c) Centre National d'Etudes Spatiales. All rights
         reserved.
10    See OTBCopyright.txt for details.
11
12
13       This software is distributed WITHOUT ANY WARRANTY; without
            even
14       the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
15       PURPOSE.  See the above copyright notices for more
            information.
16    Modified by Maria Vakalopoulou Sep 2011
         mariavakalopoulou@gmail.com
17 =========================================================================*/
18
19 //  Software Guide : BeginCommandLineArgs
20 //    INPUTS: {qb_RoadExtract.tif}
21 //    OUTPUTS: {mean_filter.tif}
22 //  Software Guide : EndCommandLineArgs
23
24 #include "otbImage.h"
25 #include "otbImageFileReader.h"
26 #include "otbVectorImage.h"
27 #include "otbStreamingImageFileWriter.h"
28 #include "itkMeanImageFilter.h"
29 #include "otbPerBandVectorImageFilter.h"
30
31  int main(int argc, char *argv[])
32  {
33
34    if (argc < 3)
35 {
36
37     std::cerr << "Usage: " << std::endl;
38     std::cerr << argv[0] << "  inputImageFile  meanfilter " <<
          std::endl;
39
40    return EXIT_FAILURE;
41
42  }
43
44
```

```
45      typedef unsigned short int                PixelType;
46      typedef otb::VectorImage<PixelType, 2> VectorImageType;
47
48      typedef otb::ImageFileReader<VectorImageType> ReaderType;
49      ReaderType::Pointer reader = ReaderType::New();
50
51      typedef otb::Image<PixelType, 2>          ImageType;
52
53      reader->SetFileName(argv[1]);
54
55      typedef itk::MeanImageFilter<ImageType, ImageType>
            FilterType;
56      FilterType::Pointer filter = FilterType::New();
57
58      ImageType::SizeType indexRadius;
59
60      indexRadius[0] = 2; // radius along x
61      indexRadius[1] = 2; // radius along y
62      filter->SetRadius(indexRadius);
63
64
65      typedef otb::PerBandVectorImageFilter
66      <VectorImageType, VectorImageType, FilterType>
            VectorFilterType;
67      VectorFilterType::Pointer vectorFilter = VectorFilterType::
            New();
68      vectorFilter->SetFilter(filter);
69
70      vectorFilter->SetInput(reader->GetOutput());
71
72
73      typedef otb::StreamingImageFileWriter<VectorImageType>
            VectorWriterType;
74      VectorWriterType::Pointer writerVector = VectorWriterType::
            New();
75
76      writerVector->SetFileName(argv[2]);
77      writerVector->SetInput(vectorFilter->GetOutput());
78
79      writerVector->Update();
80
81      return EXIT_SUCCESS;
82  }
```

5.8 Αρχείο Κώδικα για φίλτρο sobel

```
1  /*=========================================================================
2
3    Program:   ORFEO Toolbox
4    Language:  C++
5    Date:      $Date$
6    Version:   $Revision$
7
8
9    Copyright (c) Centre National d'Etudes Spatiales. All rights
       reserved.
10   See OTBCopyright.txt for details.
11
12
13       This software is distributed WITHOUT ANY WARRANTY; without
           even
14       the implied warranty of MERCHANTABILITY or FITNESS FOR A
           PARTICULAR
15       PURPOSE.  See the above copyright notices for more
           information.
16     Edited by Maria Vakalopoulou Sep 2011 mariavakalopoulou@gmail
         .com
17 =========================================================================*/
18
19 //  Software Guide : BeginLatex
20 //
21 //  The following code is an implementation of a small OTB
22 //  program.
23 //  Writen by Maria Vakalopoulou
24 //
25 //  Software Guide : EndLatex
26
27 // Software Guide : BeginCodeSnippet
28 #include <stdlib.h>
29 #include "itkExceptionObject.h"
30 #include "otbImage.h"
31 #include "otbImageFileReader.h"
32 #include "otbStreamingImageFileWriter.h"
33 #include "itkSobelEdgeDetectionImageFilter.h"
34 #include "otbVectorImage.h"
35 #include "otbStandardImageViewer.h"
36
37 int main(int argc, char *argv[] )
38 {
39   typedef otb::Image<double, 2> ImageType;
40
41   typedef otb::ImageFileReader<ImageType> ReaderType;
42   ReaderType::Pointer reader = ReaderType::New();
43
44   typedef itk::SobelEdgeDetectionImageFilter
45   <ImageType, ImageType> FilterType;
```

```
46    FilterType::Pointer filter = FilterType::New();
47
48    typedef otb::StreamingImageFileWriter<ImageType> WriterType;
49    WriterType::Pointer writer= WriterType::New();
50
51    reader->SetFileName(argv[1]);
52    writer->SetFileName(argv[2]);
53
54    filter->SetInput(reader->GetOutput());
55    writer->SetInput(filter->GetOutput());
56
57    writer->Update();
58
59   //----------------viewer-----------------------------------------
60
61     const char * inputFilename  = argv[2];
62
63
64   typedef otb::ImageFileReader<ImageType>      ReaderType;
65   typedef otb::StandardImageViewer<ImageType> ViewerType;
66
67   ViewerType::Pointer lViewer = ViewerType::New();
68   ReaderType::Pointer lReader = ReaderType::New();
69   lReader->SetFileName(inputFilename);
70   lReader->UpdateOutputInformation();
71
72   lViewer->SetLabel("My Image");
73   lViewer->SetImage(lReader->GetOutput());
74   lViewer->Update();
75
76   Fl::run();
77
78
79    return EXIT_SUCCESS;
80  }
```

5.9 Αρχείο Κώδικα για μορφολογίες

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8      Copyright (c) Centre National d'Etudes Spatiales. All rights
           reserved.
9      See OTBCopyright.txt for details.
10
11     This software is distributed WITHOUT ANY WARRANTY; without
           even
12     the implied warranty of MERCHANTABILITY or FITNESS FOR A
           PARTICULAR
13     PURPOSE.  See the above copyright notices for more
           information.
14
15 =========================================================================*/
16
17 //  Software Guide : BeginCommandLineArgs
18 //    INPUTS: {QB_Suburb.png}
19 //    OUTPUTS: {MathematicalMorphologyGrayscaleErosionOutput.png
       }
20 //    OUTPUTS: {MathematicalMorphologyGrayscaleDilationOutput.
       png}
21 //    150 180
22
23 #include "otbImage.h"
24 #include "otbImageFileReader.h"
25 #include "otbImageFileWriter.h"
26 #include "itkGrayscaleErodeImageFilter.h"
27 #include "itkGrayscaleDilateImageFilter.h"
28 #include "itkBinaryBallStructuringElement.h"
29
30 int main(int argc, char *argv[])
31 {
32
33     if (argc < 4)
34   {
35     std::cerr << "Usage: " << std::endl;
36     std::cerr << argv[0] << "  inputImageFile  ";
37     std::cerr << " outputImageFileErosion
           outputImageFileDilation" <<
38     std::endl;
39     return EXIT_FAILURE;
40 }
41
42     const unsigned int Dimension = 2;
43
```

```
44      typedef unsigned char InputPixelType;
45      typedef unsigned char OutputPixelType;
46      typedef otb::Image<InputPixelType, Dimension>
           InputImageType;
47      typedef otb::Image<OutputPixelType, Dimension>
           OutputImageType;
48
49      typedef otb::ImageFileReader<InputImageType> ReaderType;
50      typedef otb::ImageFileWriter<OutputImageType> WriterType;
51
52    typedef itk::BinaryBallStructuringElement<
53          InputPixelType, Dimension>       StructuringElementType
              ;
54
55    typedef itk::GrayscaleErodeImageFilter<
56           InputImageType, OutputImageType,
57                  StructuringElementType> ErodeFilterType;
58
59    typedef itk::GrayscaleDilateImageFilter<
60          InputImageType, OutputImageType,
61                  StructuringElementType> DilateFilterType;
62
63    ReaderType::Pointer reader = ReaderType::New();
64    WriterType::Pointer writerDilation = WriterType::New();
65    WriterType::Pointer writerErosion = WriterType::New();
66    ErodeFilterType::Pointer grayscaleErode = ErodeFilterType::
         New();
67    DilateFilterType::Pointer grayscaleDilate = DilateFilterType
         ::New();
68
69    StructuringElementType structuringElement;
70
71    structuringElement.SetRadius(1);     // 3x3 structuring
         element
72    structuringElement.CreateStructuringElement();
73    grayscaleErode->SetKernel(structuringElement);
74    grayscaleDilate->SetKernel(structuringElement);
75
76    reader->SetFileName(argv[1]);
77    writerErosion->SetFileName(argv[2]);
78    writerDilation->SetFileName(argv[3]);
79
80    grayscaleErode->SetInput(reader->GetOutput());
81    grayscaleDilate->SetInput(reader->GetOutput());
82
83    writerDilation->SetInput(grayscaleDilate->GetOutput());
84    writerDilation->Update();
85
86    writerErosion->SetInput(grayscaleErode->GetOutput());
87    writerErosion->Update();
88
```

```
89
90      return EXIT_SUCCESS;
91  }
```

6.1 Αρχείο Κώδικα για αλγόριθμο διαφοράς εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14     Modified by Maria Vakalopoulou Sep 2011
           mariavakalopoulou@gmail.com
15  =========================================================================*/
16
17  #if defined(_MSC_VER)
18  #pragma warning ( disable : 4786 )
19  #endif
20
21  //  Software Guide : BeginCommandLineArgs
22  //    INPUTS: {SpotBefore.png}, {SpotAfter.png}
23  //    OUTPUTS: {DiffChDet.tif}
24  //    3
25
26  #include "otbMeanDifferenceImageFilter.h"
27  #include "otbImageFileReader.h"
28  #include "otbImageFileWriter.h"
29  #include "otbImage.h"
30  #include "itkAbsImageFilter.h"
31  #include "itkRescaleIntensityImageFilter.h"
32  #include "otbCommandProgressUpdate.h"
33
34  int main(int argc, char* argv[])
35  {
36
37      if (argc < 5)
38      {
39          std::cerr << "Usage: " << std::endl;
40          std::cerr << argv[0] <<
41          " inputImageFile1 inputImageFile2  outputImageFile
                radius" << std::endl;
42          return -1;
43      }
44
```

```
45      const unsigned int Dimension = 2;
46
47      typedef float              InternalPixelType;
48      typedef unsigned char      OutputPixelType;
49      typedef otb::Image<InternalPixelType, Dimension>
            InputImageType1;
50      typedef otb::Image<InternalPixelType, Dimension>
            InputImageType2;
51      typedef otb::Image<InternalPixelType, Dimension>
            ChangeImageType;
52      typedef otb::Image<OutputPixelType, Dimension>
            OutputImageType;
53
54    typedef otb::ImageFileReader<InputImageType1> ReaderType1;
55    typedef otb::ImageFileReader<InputImageType2> ReaderType2;
56    typedef otb::ImageFileWriter<OutputImageType> WriterType;
57
58    typedef itk::AbsImageFilter<ChangeImageType, ChangeImageType>
          AbsType;
59    typedef itk::RescaleIntensityImageFilter<ChangeImageType,
          OutputImageType> RescalerType;
60
61    typedef otb::MeanDifferenceImageFilter<InputImageType1,
          InputImageType2, ChangeImageType>        FilterType;
62
63    ReaderType1::Pointer  reader1 = ReaderType1::New();
64    ReaderType2::Pointer  reader2 = ReaderType2::New();
65    WriterType::Pointer   writer = WriterType::New();
66    FilterType::Pointer   filter = FilterType::New();
67    AbsType::Pointer      absFilter = AbsType::New();
68    RescalerType::Pointer rescaler = RescalerType::New();
69
70    const char *inputFilename1  = argv[1];
71    const char *inputFilename2  = argv[2];
72    const char *outputFilename = argv[3];
73
74    reader1->SetFileName(inputFilename1);
75    reader2->SetFileName(inputFilename2);
76    writer->SetFileName(outputFilename);
77    rescaler->SetOutputMinimum(itk::NumericTraits<OutputPixelType
          >::min());
78    rescaler->SetOutputMaximum(itk::NumericTraits<OutputPixelType
          >::max());
79
80    filter->SetRadius(atoi(argv[4]));
81
82    filter->SetInput1(reader1->GetOutput());
83    filter->SetInput2(reader2->GetOutput());
84    absFilter->SetInput(filter->GetOutput());
85    rescaler->SetInput(absFilter->GetOutput());
86    writer->SetInput(rescaler->GetOutput());
```

```
87
88      typedef otb::CommandProgressUpdate<FilterType> CommandType;
89      CommandType::Pointer observer = CommandType::New();
90      filter->AddObserver(itk::ProgressEvent(), observer);
91
92       try
93       {
94            writer->Update();
95       }
96       catch (itk::ExceptionObject& err)
97       {
98            std::cout << "ExceptionObject caught !" << std::endl;
99            std::cout << err << std::endl;
100          return -1;
101      }
102
103
104     return EXIT_SUCCESS;
105 }
```

6.2 Αρχείο Κώδικα για αλγόριθμο λόγου εικόνων

```
1  /*=========================================================
2
3           Program:   ORFEO Toolbox
4           Language:  C++
5           Date:      $Date$
6           Version:   $Revision$
7
8          Copyright (c) Centre National d'Etudes Spatiales. All
               rights reserved.
9          See OTBCopyright.txt for details.
10
11         This software is distributed WITHOUT ANY WARRANTY;
               without even
12         the implied warranty of MERCHANTABILITY or FITNESS FOR A
               PARTICULAR
13         PURPOSE.  See the above copyright notices for more
               information.
14     Modified by Maria Vakalopoulou Sep 2011
           mariavakalopoulou@gmail.com
15 =========================================================*/
16
17 //  Software Guide : BeginCommandLineArgs
18 //    INPUTS: {GomaAvant.png}, {GomaApres.png}
19 //    OUTPUTS: {RatioChDet.tif}
20 //    3
21
22  #include "otbMeanRatioImageFilter.h"
23 #include "otbImageFileReader.h"
24 #include "otbStreamingImageFileWriter.h"
25 #include "otbImage.h"
26 #include "itkShiftScaleImageFilter.h"
27 #include "otbCommandProgressUpdate.h"
28
29     int main(int argc, char* argv[])
30       {
31
32         if (argc < 5)
33           {
34           std::cerr << "Usage: " << std::endl;
35           std::cerr << argv[0] <<
36           " inputImageFile1 inputImageFile2  outputImageFile
                radius" << std::endl;
37           return -1;
38           }
39
40         const unsigned int Dimension = 2;
41         typedef float                InternalPixelType;
42         typedef unsigned char        OutputPixelType;
43
```

```
44          typedef otb::Image<InternalPixelType, Dimension>
                InputImageType1;
45          typedef otb::Image<InternalPixelType, Dimension>
                InputImageType2;
46          typedef otb::Image<InternalPixelType, Dimension>
                ChangeImageType;
47          typedef otb::Image<OutputPixelType, Dimension>
                OutputImageType;
48
49        typedef otb::ImageFileReader<InputImageType1>
              ReaderType1;
50        typedef otb::ImageFileReader<InputImageType2>
              ReaderType2;
51        typedef otb::StreamingImageFileWriter<OutputImageType>
              WriterType;
52
53        typedef itk::ShiftScaleImageFilter<ChangeImageType,
54            OutputImageType> RescalerType;
55
56      typedef otb::MeanRatioImageFilter< InputImageType1,
            InputImageType2,
57              ChangeImageType>          FilterType;
58
59        ReaderType1::Pointer  reader1 = ReaderType1::New();
60        ReaderType2::Pointer  reader2 = ReaderType2::New();
61        WriterType::Pointer   writer = WriterType::New();
62        FilterType::Pointer   filter = FilterType::New();
63        RescalerType::Pointer rescaler = RescalerType::New();
64
65
66        const char *inputFilename1  = argv[1];
67        const char *inputFilename2  = argv[2];
68        const char *outputFilename = argv[3];
69
70        reader1->SetFileName(inputFilename1);
71        reader2->SetFileName(inputFilename2);
72        writer->SetFileName(outputFilename);
73        float scale = itk::NumericTraits<OutputPixelType>::max();
74        rescaler->SetScale(scale);
75
76        filter->SetRadius(atoi(argv[4]));
77
78        filter->SetInput1(reader1->GetOutput());
79        filter->SetInput2(reader2->GetOutput());
80        rescaler->SetInput(filter->GetOutput());
81        writer->SetInput(rescaler->GetOutput());
82
83        typedef otb::CommandProgressUpdate<FilterType> CommandType
              ;
84        CommandType::Pointer observer = CommandType::New();
85        filter->AddObserver(itk::ProgressEvent(), observer);
```

```
86
87          try
88            {
89            writer->Update();
90            }
91          catch (itk::ExceptionObject& err)
92          {
93            std::cout << "ExceptionObject caught !" << std::endl;
94            std::cout << err << std::endl;
95            return -1;
96           }
97
98      return EXIT_SUCCESS;
99  }
```

6.3 Αρχείο Κώδικα για αλγόριθμο διασποράς εικόνων

```
1  /*=========================================================================
2
3         Program:   ORFEO Toolbox
4         Language:  C++
5         Date:      $Date$
6         Version:   $Revision$
7
8        Copyright (c) Centre National d'Etudes Spatiales. All
             rights reserved.
9        See OTBCopyright.txt for details.
10
11          This software is distributed WITHOUT ANY WARRANTY;
               without even
12          the implied warranty of MERCHANTABILITY or FITNESS FOR
               A PARTICULAR
13          PURPOSE.  See the above copyright notices for more
               information.
14     Modified by Maria Vakalopoulou Sep 2011
          mariavakalopoulou@gmail.com
15   =========================================================================*/
16
17 #include "otbImageFileReader.h"
18 #include "otbStreamingImageFileWriter.h"
19 #include "otbImage.h"
20 #include "itkShiftScaleImageFilter.h"
21 #include "otbCommandProgressUpdate.h"
22
23 //  Software Guide : BeginCommandLineArgs
24 //    INPUTS: {ERSBefore.png}, {ERSAfter.png}
25 //    OUTPUTS: {CorrChDet.tif}
26 //    15
27
28
29 #include "otbCorrelationChangeDetector.h"
30
31
32
33
34     int main(int argc, char* argv[])
35      {
36
37        if (argc < 5)
38         {
39          std::cerr << "Usage: " << std::endl;
40          std::cerr << argv[0] << " inputImageFile1
               inputImageFile2 "
41                    << "outputImageFile radius" << std::endl;
42          return -1;
43          }
44
```

```
45        const unsigned int Dimension = 2;
46
47        typedef float
              InternalPixelType;
48        typedef unsigned char
              OutputPixelType;
49        typedef otb::Image<InternalPixelType, Dimension>
              InputImageType1;
50        typedef otb::Image<InternalPixelType, Dimension>
              InputImageType2;
51        typedef otb::Image<InternalPixelType, Dimension>
              ChangeImageType;
52        typedef otb::Image<OutputPixelType, Dimension>
              OutputImageType;
53
54        typedef otb::ImageFileReader<InputImageType1>
              ReaderType1;
55        typedef otb::ImageFileReader<InputImageType2>
              ReaderType2;
56        typedef otb::StreamingImageFileWriter<OutputImageType>
              WriterType;
57
58       typedef itk::ShiftScaleImageFilter<ChangeImageType,
59            OutputImageType> RescalerType;
60
61    typedef otb::CorrelationChangeDetector<InputImageType1,
        InputImageType2,
62            ChangeImageType>          FilterType;
63
64       ReaderType1::Pointer  reader1 = ReaderType1::New();
65       ReaderType2::Pointer  reader2 = ReaderType2::New();
66       WriterType::Pointer   writer = WriterType::New();
67       FilterType::Pointer   filter = FilterType::New();
68       RescalerType::Pointer rescaler = RescalerType::New();
69
70       const char *inputFilename1  = argv[1];
71       const char *inputFilename2  = argv[2];
72       const char *outputFilename = argv[3];
73
74       reader1->SetFileName(inputFilename1);
75       reader2->SetFileName(inputFilename2);
76       writer->SetFileName(outputFilename);
77       float scale = itk::NumericTraits<OutputPixelType>::max();
78       rescaler->SetScale(scale);
79
80       filter->SetRadius(atoi(argv[4]));
81
82       filter->SetInput1(reader1->GetOutput());
83       filter->SetInput2(reader2->GetOutput());
84       rescaler->SetInput(filter->GetOutput());
85       writer->SetInput(rescaler->GetOutput());
```

```
86
87
88          typedef otb::CommandProgressUpdate<FilterType> CommandType
                ;
89          CommandType::Pointer observer = CommandType::New();
90          filter->AddObserver(itk::ProgressEvent(), observer);
91
92          try
93            {
94            writer->Update();
95            }
96          catch (itk::ExceptionObject& err)
97            {
98            std::cout << "ExceptionObject caught !" << std::endl;
99            std::cout << err << std::endl;
100           return -1;
101           }
102
103         return EXIT_SUCCESS;
104 }
```

6.4 Αρχείο Κώδικα για αλγόριθμο LHMI εικόνων

```
/*=========================================================================
      Language:  C++
      Date:      $Date$
      Version:   $Revision$

      Copyright (c) Centre National d'Etudes Spatiales. All rights reserved.
      See OTBCopyright.txt for details.

       This software is distributed WITHOUT ANY WARRANTY; without even
       the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
       PURPOSE.  See the above copyright notices for more information.
   Modified by Maria Vakalopoulou Sep 2011 mariavakalopoulou@gmail.com
=========================================================================*/

#include "otbImageFileReader.h"
#include "otbStreamingImageFileWriter.h"
#include "otbImage.h"
#include "itkShiftScaleImageFilter.h"
#include "otbLHMIChangeDetector.h"
#include "otbCommandProgressUpdate.h"

    int main(int argc, char *argv[])
     {

       if (argc < 5)
        {
         std::cerr << "Usage: " << std::endl;
         std::cerr << argv[0] <<
         " inputImageFile1 inputImageFile2  radius  outputImageFile " << std::endl;
         return -1;
         }

// Define the dimension of the images

       const unsigned int Dimension = 2;

// Declare the types of the images

       typedef float                 InternalPixelType;
       typedef unsigned char         OutputPixelType;
       typedef otb::Image<InternalPixelType, Dimension> InputImageType1;
       typedef otb::Image<InternalPixelType, Dimension> InputImageType2;
```

```
43          typedef otb::Image<InternalPixelType, Dimension>
               ChangeImageType;
44          typedef otb::Image<OutputPixelType, Dimension>
               OutputImageType;
45          typedef otb::ImageFileReader<InputImageType1>
               ReaderType1;
46          typedef otb::ImageFileReader<InputImageType2>
               ReaderType2;
47          typedef otb::StreamingImageFileWriter<OutputImageType>
               WriterType;
48          typedef itk::ShiftScaleImageFilter<ChangeImageType,
49               OutputImageType> RescalerType;
50
51  // Declare the type for the filter
52
53          typedef otb::LHMIChangeDetector<InputImageType1,
               InputImageType2,
54               ChangeImageType>          FilterType;
55
56          ReaderType1::Pointer   reader1 = ReaderType1::New();
57          ReaderType2::Pointer   reader2 = ReaderType2::New();
58          WriterType::Pointer    writer = WriterType::New();
59          FilterType::Pointer    filter = FilterType::New();
60          RescalerType::Pointer rescaler = RescalerType::New();
61
62          const char * inputFilename1  = argv[1];
63          const char * inputFilename2  = argv[2];
64          const char * outputFilename = argv[4];
65
66          reader1->SetFileName(inputFilename1);
67          reader2->SetFileName(inputFilename2);
68          writer->SetFileName(outputFilename);
69
70
71          float scale = itk::NumericTraits<OutputPixelType>::max();
72          rescaler->SetScale(scale);
73          filter->SetInput1(reader1->GetOutput());
74          filter->SetInput2(reader2->GetOutput());
75          filter->SetRadius(atoi(argv[3]));
76          rescaler->SetInput(filter->GetOutput());
77          writer->SetInput(rescaler->GetOutput());
78
79          typedef otb::CommandProgressUpdate<FilterType>
               CommandType;
80          CommandType::Pointer observer = CommandType::New();
81          filter->AddObserver(itk::ProgressEvent(), observer);
82
83          try
84            {
85            writer->Update();
86            }
```

```
87          catch (itk::ExceptionObject& err)
88            {
89            std::cout << "ExceptionObject caught !" << std::endl;
90            std::cout << err << std::endl;
91            return -1;
92            }
93
94        return EXIT_SUCCESS;
95  }
```

6.5 Αρχείο Κώδικα για αλγόριθμο CBAMI εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14     Edited by Maria Vakalopoulou Sep 2011 mariavakalopoulou@gmail
           .com
15  =========================================================================*/
16
17  #include "otbImageFileReader.h"
18  #include "otbStreamingImageFileWriter.h"
19  #include "otbImage.h"
20  #include "otbCBAMIChangeDetector.h"
21  #include "otbCommandProgressUpdate.h"
22
23     int main(int argc, char* argv[])
24      {
25
26        if (argc < 5)
27         {
28           std::cerr << "Usage: " << std::endl;
29          std::cerr << argv[0] <<
30           " inputImageFile1 inputImageFile2  radius
                outputImageFile " << std::endl;
31           return -1;
32          }
33
34  // Define the dimension of the images
35
36       const unsigned int Dimension = 2;
37
38  // Declare the types of the images
39
40       typedef float
             InternalPixelType;
41       typedef float                                OutputPixelType;
42       typedef otb::Image<InternalPixelType, Dimension>
             InputImageType1;
```

```
43      typedef otb::Image<InternalPixelType, Dimension>
            InputImageType2;
44      typedef otb::Image<InternalPixelType, Dimension>
            ChangeImageType;
45      typedef otb::Image<OutputPixelType, Dimension>
            OutputImageType;
46      typedef otb::ImageFileReader<InputImageType1>
            ReaderType1;
47      typedef otb::ImageFileReader<InputImageType2>
            ReaderType2;
48      typedef otb::StreamingImageFileWriter<OutputImageType>
            WriterType;
49
50
51  // Declare the type for the filter
52
53      typedef otb::CBAMIChangeDetector<
54        InputImageType1,
55        InputImageType2,
56        ChangeImageType>        FilterType;
57
58      ReaderType1::Pointer  reader1 = ReaderType1::New();
59      ReaderType2::Pointer  reader2 = ReaderType2::New();
60      WriterType::Pointer   writer = WriterType::New();
61      FilterType::Pointer   filter = FilterType::New();
62
63
64      const char * inputFilename1  = argv[1];
65      const char * inputFilename2  = argv[2];
66      const char * outputFilename = argv[4];
67
68      reader1->SetFileName(inputFilename1);
69      reader2->SetFileName(inputFilename2);
70      writer->SetFileName(outputFilename);
71
72
73      filter->SetInput1(reader1->GetOutput());
74      filter->SetInput2(reader2->GetOutput());
75      filter->SetRadius(atoi(argv[3]));
76
77      writer->SetInput(filter->GetOutput());
78
79      typedef otb::CommandProgressUpdate<FilterType> CommandType;
80      CommandType::Pointer observer = CommandType::New();
81      filter->AddObserver(itk::ProgressEvent(), observer);
82
83       try
84         {
85          writer->Update();
86         }
87       catch (itk::ExceptionObject& err)
```

```
88          {
89           std::cout << "ExceptionObject caught !" << std::endl;
90           std::cout << err << std::endl;
91           return -1;
92          }
93
94         return EXIT_SUCCESS;
95 }
```

7.1 Αρχείο Κώδικα για αλγόριθμο Διαφοράς Έντασης .cxx

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
15
16  =========================================================================*/
17
18  #include "otbVectorImage.h"
19  #include "otbImage.h"
20  #include "otbImageFileReader.h"
21  #include "otbImageFileWriter.h"
22  #include "itkImageRegionIterator.h"
23  #include "itkImageRegionConstIterator.h"
24
25  #include <math.h>
26
27  //  Software Guide : BeginCommandLineArgs
28  //    INPUTS: {InputImage.tif}
29  //    OUTPUTS: {OutputImage.tif}
30
31  int main(int argc, char * argv[] )
32  {
33
34      if (argc < 2)
35      {
36          std::cerr << "Usage: " << std::endl;
37          std::cerr << argv[0] << " inputImageFile outputImageFile
                " << std::endl;
38          return -1;
39      }
40
41  // Define the dimension of the images
42    const unsigned int Dimension = 2;
43
```

```
44 // We start by declaring the types for the input image, wich is
       vector type
45 // and the output image, which is image type
46
47  typedef double                  PixelType;
48  typedef otb::VectorImage<PixelType, Dimension> VectorImageType;
49  typedef otb::Image<PixelType, Dimension> ImageType;
50  typedef otb::ImageFileReader<VectorImageType> ReaderType;
51
52 // We declare the types for the iterators. For the vector type
       image,
53 //the iterator is const
54
55   typedef itk::ImageRegionConstIterator<VectorImageType>
         IteratorType2;
56   typedef itk::ImageRegionIterator<ImageType> IteratorType;
57
58 // We can now declare the type for the reader.
59   VectorImageType::ConstPointer inputImage;
60   ReaderType::Pointer  reader = ReaderType::New();
61   reader -> SetFileName(argv[1]);
62   try
63   {
64           reader -> Update();
65           inputImage = reader -> GetOutput();
66   }
67   catch (itk::ExceptionObject& err)
68   {
69           std::cout << "ExceptionObject␣caught␣!" << std::endl;
70           std::cout << err << std::endl;
71           return -1;
72   }
73
74 //We create the output image and its iterator by fill it 0
75   ImageType::Pointer outputImage = ImageType::New();
76   outputImage -> SetRegions(inputImage -> GetRequestedRegion());
77   outputImage ->Allocate();
78   IteratorType outputIt (outputImage, outputImage ->
         GetRequestedRegion());
79   outputImage->FillBuffer(0);
80
81 //We create the iterator for the input images and extract the
       number
82 //of the bands
83   IteratorType2 inputIt (inputImage, inputImage ->
         GetRequestedRegion());
84   unsigned int band = reader->GetOutput()->
         GetNumberOfComponentsPerPixel();
85   double pixelInput=0;
86
```

```
87 //The two iterators go to the beginning of the image that they
       declare
88
89   inputIt.GoToBegin();
90   outputIt.GoToBegin();
91
92 //The prossess will continue until the iterator of the input
       image go to end
93
94   while (!inputIt.IsAtEnd())
95   {
96 //We declare the type of the output pixel
97           PixelType pixelOutput;
98
99 //For all the bands of the image
100          for(unsigned int i=0; i<band; i++)
101           {
102                   pixelInput += inputIt.Get()[i]*inputIt.Get()[i
                        ];
103           }
104           pixelOutput = static_cast<PixelType>(sqrt(pixelInput)
                  );
105           outputIt.Set(pixelOutput);
106           ++inputIt;
107           ++outputIt;
108           pixelInput=0;
109  }
110
111
112 //We create the writer
113   typedef otb::ImageFileWriter<ImageType> WriterType;
114   WriterType::Pointer writer= WriterType::New();
115
116   writer->SetFileName(argv[2]);
117   writer->SetInput(outputImage);
118
119   try
120     {
121          writer->Update();
122     }
123     catch (itk::ExceptionObject& err)
124     {
125          std::cout << "ExceptionObject caught !" << std::endl;
126          std::cout << err << std::endl;
127         return -1;
128     }
129
130
131   return EXIT_SUCCESS;
132 }
```

7.2 Εκτελέσιμο αρχείο για διαφορά εικόνων και ανίχνευση μεταβολών

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Modified by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18
19  #if defined(_MSC_VER)
20  #pragma warning ( disable : 4786 )
21  #endif
22
23  //  Software Guide : BeginCommandLineArgs
24  //    INPUTS: {InputImage1.tif}, {InputImage2.tif}
25  //    OUTPUTS: {DiffChDet.tif}
26  //    3
27  //  Software Guide : EndCommandLineArgs
28
29  #include "otbNomarlizedMeanDifferenceImageFilter.h"
30  #include "otbImageFileReader.h"
31  #include "otbImageFileWriter.h"
32  #include "otbImage.h"
33  #include "otbCommandProgressUpdate.h"
34
35  int main(int argc, char* argv[])
36  {
37
38      if (argc < 5)
39      {
40          std::cerr << "Usage: " << std::endl;
41          std::cerr << argv[0] <<
42          " inputImageFile1 inputImageFile2  outputImageFile
                radius" << std::endl;
```

```
43          return -1;
44      }
45
46      const unsigned int Dimension = 2;
47      typedef float
           InternalPixelType;
48      typedef unsigned char
           OutputPixelType;
49      typedef otb::Image<InternalPixelType, Dimension>
           InputImageType1;
50      typedef otb::Image<InternalPixelType, Dimension>
           InputImageType2;
51      typedef otb::Image<InternalPixelType, Dimension>
           ChangeImageType;
52      typedef otb::ImageFileReader<InputImageType1> ReaderType1;
53      typedef otb::ImageFileReader<InputImageType2> ReaderType2;
54      typedef otb::ImageFileWriter<ChangeImageType> WriterType;
55
56     typedef otb::NomarlizedMeanDifferenceImageFilter<
          InputImageType1, InputImageType2, ChangeImageType>
          FilterType;
57
58     ReaderType1::Pointer  reader1 = ReaderType1::New();
59     ReaderType2::Pointer  reader2 = ReaderType2::New();
60     WriterType::Pointer   writer = WriterType::New();
61     FilterType::Pointer   filter = FilterType::New();
62
63     const char *inputFilename1  = argv[1];
64     const char *inputFilename2  = argv[2];
65     const char *outputFilename = argv[3];
66
67     reader1->SetFileName(inputFilename1);
68     reader2->SetFileName(inputFilename2);
69     writer->SetFileName(outputFilename);
70
71     filter->SetRadius(atoi(argv[4]));
72
73     filter->SetInput1(reader1->GetOutput());
74     filter->SetInput2(reader2->GetOutput());
75     writer->SetInput(filter->GetOutput());
76
77     typedef otb::CommandProgressUpdate<FilterType> CommandType;
78     CommandType::Pointer observer = CommandType::New();
79     filter->AddObserver(itk::ProgressEvent(), observer);
80
81      try
82      {
83          writer->Update();
84      }
85      catch (itk::ExceptionObject& err)
86      {
```

```
87            std::cout << "Exception Object caught !" << std::endl;
88            std::cout << err << std::endl;
89           return -1;
90        }
91
92      return EXIT_SUCCESS;
93  }
```

7.3 Αρχείο επικεφαλίδας για διαφορά εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18
19
20  #ifndef __otbNomarlizedMeanDifferenceImageFilter_h
21  #define __otbNomarlizedMeanDifferenceImageFilter2_h
22
23
24      #include "otbBinaryFunctorNeighborhoodImageFilter.h"
25      #include "otbNomarlizedMeanDifference.h"
26
27      namespace otb
28      {
29      template <class TInputImage1, class TInputImage2, class
            TOutputImage>
30      class ITK_EXPORT NomarlizedMeanDifferenceImageFilter :
31          public BinaryFunctorNeighborhoodImageFilter<
32              TInputImage1, TInputImage2, TOutputImage,
33              Functor::NomarlizedMeanDifference<
34                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage1>,
35                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage2>,
36                  ITK_TYPENAME TOutputImage::PixelType> >
37      {
38      public:
39  /** Standard class typedefs. */
40
41          typedef NomarlizedMeanDifferenceImageFilter Self;
42          typedef BinaryFunctorNeighborhoodImageFilter<
```

```
43              TInputImage1, TInputImage2, TOutputImage,
44              Functor::NomarlizedMeanDifference<
45                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage1>,
46                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage2>,
47                  ITK_TYPENAME TOutputImage::PixelType> >
                        Superclass;
48      typedef itk::SmartPointer<Self>        Pointer;
49      typedef itk::SmartPointer<const Self> ConstPointer;
50
51  /** Method for creation through the object factory. */
52
53      itkNewMacro(Self);
54
55  /** Macro defining the type*/
56
57      itkTypeMacro(NomarlizedMeanDifferenceImageFilter, SuperClass
            );
58
59      protected:
60      NomarlizedMeanDifferenceImageFilter() {}
61      virtual ~NomarlizedMeanDifferenceImageFilter() {}
62
63      private:
64      NomarlizedMeanDifferenceImageFilter(const Self &); //
            purposely not implemented
65      void operator =(const Self&); //purposely not implemented
66
67  };
68  } // end namespace otb
69  #endif
```

7.4 Αρχείο επικεφαλίδας για διαφορά εικόνων Functor

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18  #ifndef __otbNomarlizedMeanDifference_h
19  #define __otbNomarlizedMeanDifference_h
20
21  namespace otb
22  {
23
24  namespace Functor
25  {
26  template<class TInput1, class TInput2, class TOutput>
27      class NomarlizedMeanDifference
28
29      {
30      public:
31      NomarlizedMeanDifference() {}
32      virtual ~NomarlizedMeanDifference() {}
33      inline TOutput operator ()(const TInput1& itA,
34                                 const TInput2& itB)
35      {
36
37      TOutput meanA = 0.0;
38      TOutput meanB = 0.0;
39
40      for (unsigned long pos = 0; pos < itA.Size(); ++pos)
41      {
42            meanA += static_cast<TOutput>(itA.GetPixel(pos));
43            meanB += static_cast<TOutput>(itB.GetPixel(pos));
44       }
45
```

```
46      return static_cast<TOutput>(((meanB - meanA) / meanA)+((
            meanB - meanA) / meanB));
47  }
48  };
49  }
50  }
51
52  #endif
```

7.5 Εκτελέσιμο αρχείο αλγορίθμου Tasseled Cap

```
1  /*=========================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================*/
18
19  #include "otbVectorImage.h"
20  #include "otbImage.h"
21  #include "otbImageFileReader.h"
22  #include "otbImageFileWriter.h"
23  #include "itkImageRegionIterator.h"
24  #include "itkImageRegionConstIterator.h"
25
26
27  #include <math.h>
28
29  //  Software Guide : BeginCommandLineArgs
30  //    INPUTS: {InputImage.tif}
31  //    OUTPUTS: {OutputImage1.tif}, {OutputImage2.tif},
32
33
34  int main(int argc, char * argv[] )
35  {
36            if (argc < 3)
37      {
38          std::cerr << "Usage: " << std::endl;
39          std::cerr << argv[0] << " inputImageFile
                outputImageFile1 outputImageFile2" << std::endl;
40          return -1;
41      }
42
43  // Define the dimension of the images
44    const unsigned int Dimension = 2;
```

```
45
46 // We start by declaring the types for the input image, wich is
       vector type
47 // and the output image, which is image type
48   typedef double                PixelType;
49   typedef otb::VectorImage<PixelType, Dimension> VectorImageType
       ;
50   typedef otb::Image<PixelType, Dimension> ImageType;
51   typedef otb::ImageFileReader<VectorImageType> ReaderType;
52
53 // We declare the types for the iterators. For the vector type
       image,
54 //the iterator is const
55
56   typedef itk::ImageRegionConstIterator<VectorImageType>
       IteratorType2;
57   typedef itk::ImageRegionIterator<ImageType> IteratorType;
58
59 // We can now declare the type for the reader.
60   VectorImageType::ConstPointer inputImage;
61   ReaderType::Pointer  reader = ReaderType::New();
62   reader -> SetFileName(argv[1]);
63   try
64   {
65           reader -> Update();
66           inputImage = reader -> GetOutput();
67   }
68   catch (itk::ExceptionObject& err)
69   {
70           std::cout << "ExceptionObject caught !" << std::endl;
71           std::cout << err << std::endl;
72           return -1;
73   }
74
75 //We create the two output images and its iterator by fill it 0
76   ImageType::Pointer outputImage1 = ImageType::New();
77   ImageType::Pointer outputImage2 = ImageType::New();
78   outputImage1 -> SetRegions(inputImage -> GetRequestedRegion())
       ;
79   outputImage1 ->Allocate();
80   outputImage2 -> SetRegions(inputImage -> GetRequestedRegion())
       ;
81   outputImage2 ->Allocate();
82   IteratorType outputIt1 (outputImage1, outputImage1 ->
       GetRequestedRegion());
83   IteratorType outputIt2 (outputImage2, outputImage2 ->
       GetRequestedRegion());
84
85   outputImage1->FillBuffer(0);
86   outputImage2->FillBuffer(0);
87
```

```
88 //We create the iterator for the input images and extract the
       number
89 //of the bands
90   IteratorType2 inputIt (inputImage, inputImage ->
         GetRequestedRegion());
91   unsigned int band = reader->GetOutput()->
         GetNumberOfComponentsPerPixel();
92   double pixelInput1=0;
93   double pixelInput2=0;
94
95 //The three iterators go to the beginning of the image that they
       declare
96
97   inputIt.GoToBegin();
98   outputIt1.GoToBegin();
99   outputIt2.GoToBegin();
100
101 //The prossess will continue until the iterator of the input
       image go to end
102
103   while (!inputIt.IsAtEnd())
104   {
105 //We declare the type of the output pixel
106          PixelType pixelOutput1;
107          PixelType pixelOutput2;
108
109 //For all the bands of the image
110          for(unsigned int i=0; i<band; i++)
111           {
112                   if (i = 0)
113                   {
114                           pixelInput1 = 0.326*inputIt.Get()[i];
115                           pixelInput2 = -0.311*inputIt.Get()[i];
116                   }
117                   if (i =1)
118                   {
119                           pixelInput1 += 0.509*inputIt.Get()[i];
120                           pixelInput2 += -0.356*inputIt.Get()[i
                                ];
121                   }
122                   if (i =2)
123                   {
124                           pixelInput1 += 0.560*inputIt.Get()[i];
125                           pixelInput2 += -0.325*inputIt.Get()[i
                                ];
126                   }
127                  if (i=3)
128                   {
129                           pixelInput1 += 0.567*inputIt.Get()[i];
130                           pixelInput2 += 0.819*inputIt.Get()[i];
131                   }
```

```
132             }
133             pixelOutput1 = static_cast<PixelType>(pixelInput1);
134             pixelOutput2 = static_cast<PixelType>(pixelInput2);
135             outputIt1.Set(pixelOutput1);
136             outputIt2.Set(pixelOutput2);
137             ++inputIt;
138             ++outputIt1;
139             ++outputIt2;
140             pixelInput1=0;
141             pixelInput2=0;
142   }
143
144
145 //We create the writer
146   typedef otb::ImageFileWriter<ImageType> WriterType;
147   WriterType::Pointer writer1= WriterType::New();
148   WriterType::Pointer writer2= WriterType::New();
149
150
151   writer1->SetFileName(argv[2]);
152   writer1->SetInput(outputImage1);
153   writer2->SetFileName(argv[3]);
154   writer2->SetInput(outputImage2);
155
156   try
157     {
158         writer1->Update();
159         writer2->Update();
160     }
161     catch (itk::ExceptionObject& err)
162     {
163         std::cout << "ExceptionObject caught !" << std::endl;
164         std::cout << err << std::endl;
165        return -1;
166     }
167
168   return EXIT_SUCCESS;
169 }
```

7.6 Εκτελέσιμο αρχείο αλγορίθμου Διαχωρισμού Χρωμάτων

```
1  /*=========================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================*/
18  #include "otbVectorImage.h"
19  #include "otbImage.h"
20  #include "otbImageFileReader.h"
21  #include "otbImageFileWriter.h"
22  #include "itkImageRegionIterator.h"
23  #include "itkImageRegionConstIterator.h"
24
25  #include <math.h>
26
27  //  Software Guide : BeginCommandLineArgs
28  //    INPUTS: {InputImage.tif}
29  //    OUTPUTS: {OutputImage1.tif}, {OutputImage2.tif}, {
        OutputImage_n.tif}
30
31
32  int main(int argc, char * argv[] )
33  {
34
35  // Define the dimension of the images
36    const unsigned int Dimension = 2;
37
38  // We start by declaring the types for the input image, wich is
        vector type
39  // and the output image, which is image type
40    typedef double                 PixelType;
41    typedef otb::VectorImage<PixelType, Dimension> VectorImageType
          ;
42    typedef otb::Image<PixelType, Dimension> ImageType;
```

```
43   typedef otb::ImageFileReader<VectorImageType> ReaderType;
44
45 // We declare the types for the iterators. For the vector type
       image,
46 //the iterator is const
47
48   typedef itk::ImageRegionConstIterator<VectorImageType>
       IteratorType2;
49   typedef itk::ImageRegionIterator<ImageType> IteratorType;
50
51 // We can now declare the type for the reader.
52   VectorImageType::ConstPointer inputImage;
53   ReaderType::Pointer  reader = ReaderType::New();
54   reader -> SetFileName(argv[1]);
55   try
56   {
57           reader -> Update();
58           inputImage = reader -> GetOutput();
59   }
60   catch (itk::ExceptionObject& err)
61   {
62           std::cout << "ExceptionObject caught !" << std::endl;
63           std::cout << err << std::endl;
64           return -1;
65   }
66
67 //We create the iterator for the input images and extract the
       number
68 //of the bands
69   IteratorType2 inputIt (inputImage, inputImage ->
       GetRequestedRegion());
70   unsigned int band = reader->GetOutput()->
       GetNumberOfComponentsPerPixel();
71   double pixelInput=0;
72   double temp=0;
73
74 //For every band, the algorithm produce a new output image
75
76    for (unsigned int j=0; j<band; j++)
77   {
78 //We create the output image and its iterator by fill it 0
79
80        ImageType::Pointer outputImage = ImageType::New();
81        outputImage -> SetRegions(inputImage -> GetRequestedRegion
              ());
82        outputImage ->Allocate();
83        IteratorType outputIt (outputImage, outputImage ->
              GetRequestedRegion());
84        outputImage->FillBuffer(0);
85
```

```
86 //The iterators go to the beginning of the image that they
       declare
87
88         inputIt.GoToBegin();
89         outputIt.GoToBegin();
90
91 //The prossess will continue until the iterator of the input
       image go to end
92
93         while (!inputIt.IsAtEnd())
94         {
95 //We declare the type of the output pixel
96
97                         PixelType pixelOutput;
98
99 //For all the bands of the image
100
101                  for(unsigned int i=0; i<band; i++)
102                 {
103                             pixelInput += inputIt.Get()[i]*inputIt
                                .Get()[i];
104                         temp=inputIt.Get()[j];
105                 }
106                 pixelOutput = static_cast<PixelType>(temp/sqrt(
                       pixelInput) );
107                 outputIt.Set(pixelOutput);
108                 ++inputIt;
109                 ++outputIt;
110                 pixelInput=0;
111                 temp=0;
112         }
113
114 //We create the writer
115         typedef otb::ImageFileWriter<ImageType> WriterType;
116         WriterType::Pointer writer= WriterType::New();
117
118         writer->SetFileName(argv[2+j]);
119         writer->SetInput(outputImage);
120
121          try
122       {
123            writer->Update();
124       }
125       catch (itk::ExceptionObject& err)
126       {
127            std::cout << "ExceptionObject caught !" << std::endl;
128            std::cout << err << std::endl;
129           return -1;
130       }
131
132   }
```

```
133
134     return EXIT_SUCCESS;
135 }
```

7.7 Αρχείο επικεφαλίδα για λόγο εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18
19  #ifndef __otbNomarlizedMeanRatioImageFilter_h
20  #define __otbNomarlizedMeanRatioImageFilter_h
21
22
23      #include "otbBinaryFunctorNeighborhoodImageFilter.h"
24      #include "otbNomarlizedMeanRatio.h"
25
26      namespace otb
27      {
28      template <class TInputImage1, class TInputImage2, class
            TOutputImage>
29      class ITK_EXPORT NomarlizedMeanRatioImageFilter :
30          public BinaryFunctorNeighborhoodImageFilter<
31              TInputImage1, TInputImage2, TOutputImage,
32              Functor::NomarlizedMeanRatio<
33                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage1>,
34                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage2>,
35                  ITK_TYPENAME TOutputImage::PixelType> >
36      {
37      public:
38  /** Standard class typedefs. */
39
40          typedef NomarlizedMeanRatioImageFilter Self;
41          typedef BinaryFunctorNeighborhoodImageFilter<
42              TInputImage1, TInputImage2, TOutputImage,
```

```
43              Functor::NomarlizedMeanRatio<
44                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage1>,
45                  ITK_TYPENAME itk::ConstNeighborhoodIterator<
                        TInputImage2>,
46                  ITK_TYPENAME TOutputImage::PixelType>  >
                        Superclass;
47      typedef itk::SmartPointer<Self>        Pointer;
48      typedef itk::SmartPointer<const Self> ConstPointer;
49
50  /** Method for creation through the object factory. */
51
52      itkNewMacro(Self);
53
54  /** Macro defining the type*/
55
56      itkTypeMacro(NomarlizedMeanRatioImageFilter, SuperClass);
57
58      protected:
59      NomarlizedMeanRatioImageFilter() {}
60      virtual ~NomarlizedMeanRatioImageFilter() {}
61
62      private:
63      NomarlizedMeanRatioImageFilter(const Self &); //purposely
            not implemented
64      void operator =(const Self&); //purposely not implemented
65
66  };
67  } // end namespace otb
68  #endif
```

7.8 Αρχείο επικεφαλίδα για λόγο εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18
19  #ifndef __otbNomarlizedMeanRatio_h
20  #define __otbNomarlizedMeanRatio_h
21
22  namespace otb
23  {
24
25  namespace Functor
26  {
27  template<class TInput1, class TInput2, class TOutput>
28      class NomarlizedMeanRatio
29
30      {
31      public:
32      NomarlizedMeanRatio() {}
33      virtual ~NomarlizedMeanRatio() {}
34      inline TOutput operator ()(const TInput1& itA,
35                                 const TInput2& itB)
36      {
37
38      TOutput meanA = 0.0;
39      TOutput meanB = 0.0;
40
41      for (unsigned long pos = 0; pos < itA.Size(); ++pos)
42      {
43            meanA += static_cast<TOutput>(itA.GetPixel(pos));
44            meanB += static_cast<TOutput>(itB.GetPixel(pos));
45       }
```

```
46
47      return static_cast<TOutput>((meanB - meanA) / (meanA + meanB
            ));
48  }
49  };
50  }
51  }
52
53  #endif
```

7.9 Εκτελέσιμο αρχείο για λόγο εικόνων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15     Modified by Maria Vakalopoulou Sep 2011
           mariavakalopoulou@gmail.com
16
17  =========================================================================
       */
18
19  #if defined(_MSC_VER)
20  #pragma warning ( disable : 4786 )
21  #endif
22
23  //  Software Guide : BeginCommandLineArgs
24  //    INPUTS: {InputImage1.tif}, {InputImage2.tif}
25  //    OUTPUTS: {RatioChDet.tif}
26  //    3
27
28  #include "otbNomarlizedMeanRatioImageFilter.h"
29  #include "otbImageFileReader.h"
30  #include "otbStreamingImageFileWriter.h"
31  #include "otbImage.h"
32  #include "otbCommandProgressUpdate.h"
33  #include "itkAbsImageFilter.h"
34  #include "itkShiftScaleImageFilter.h"
35
36  int main(int argc, char* argv[])
37  {
38
39      if (argc < 5)
40      {
41          std::cerr << "Usage: " << std::endl;
42          std::cerr << argv[0] <<
43          " inputImageFile1 inputImageFile2  outputImageFile
                radius" << std::endl;
```

```
44          return -1;
45      }
46
47      const unsigned int Dimension = 2;
48      typedef float
            InternalPixelType;
49      typedef unsigned char
            OutputPixelType;
50      typedef otb::Image<InternalPixelType, Dimension>
            InputImageType1;
51      typedef otb::Image<InternalPixelType, Dimension>
            InputImageType2;
52      typedef otb::Image<InternalPixelType, Dimension>
            ChangeImageType;
53      typedef otb::Image<OutputPixelType, Dimension>
            OutputImageType;
54      typedef otb::ImageFileReader<InputImageType1> ReaderType1;
55      typedef otb::ImageFileReader<InputImageType2> ReaderType2;
56      typedef otb::StreamingImageFileWriter<OutputImageType>
            WriterType;
57
58
59  typedef itk::AbsImageFilter<ChangeImageType, ChangeImageType>
        AbsType;
60  typedef itk::ShiftScaleImageFilter<ChangeImageType,
        OutputImageType> RescalerType;
61  typedef otb::NomarlizedMeanRatioImageFilter<InputImageType1,
        InputImageType2, ChangeImageType>        FilterType;
62
63    ReaderType1::Pointer  reader1 = ReaderType1::New();
64    ReaderType2::Pointer  reader2 = ReaderType2::New();
65    WriterType::Pointer   writer = WriterType::New();
66    FilterType::Pointer   filter = FilterType::New();
67    AbsType::Pointer      absFilter = AbsType::New();
68    RescalerType::Pointer rescaler = RescalerType::New();
69
70    const char *inputFilename1  = argv[1];
71    const char *inputFilename2  = argv[2];
72    const char *outputFilename = argv[3];
73
74    reader1->SetFileName(inputFilename1);
75    reader2->SetFileName(inputFilename2);
76    writer->SetFileName(outputFilename);
77    float scale = itk::NumericTraits<OutputPixelType>::max();
78    rescaler->SetScale(scale);
79
80    filter->SetRadius(atoi(argv[4]));
81    filter->SetInput1(reader1->GetOutput());
82    filter->SetInput2(reader2->GetOutput());
83    absFilter->SetInput(filter->GetOutput());
84    rescaler->SetInput(absFilter->GetOutput());
```

```
85      writer->SetInput(rescaler->GetOutput());
86
87      typedef otb::CommandProgressUpdate<FilterType> CommandType;
88      CommandType::Pointer observer = CommandType::New();
89      filter->AddObserver(itk::ProgressEvent(), observer);
90
91       try
92       {
93            writer->Update();
94       }
95       catch (itk::ExceptionObject& err)
96       {
97            std::cout << "ExceptionObject caught !" << std::endl;
98            std::cout << err << std::endl;
99           return -1;
100      }
101
102
103     return EXIT_SUCCESS;
104 }
```

7.10 Εκτελέσιμο αρχείο για Διαφορών Κλίσεων

```
1  /*=========================================================================
2
3       Program:   ORFEO Toolbox
4       Language:  C++
5       Date:      $Date$
6       Version:   $Revision$
7
8       Copyright (c) Centre National d'Etudes Spatiales. All rights
            reserved.
9       See OTBCopyright.txt for details.
10
11      This software is distributed WITHOUT ANY WARRANTY; without
            even
12      the implied warranty of MERCHANTABILITY or FITNESS FOR A
            PARTICULAR
13      PURPOSE.  See the above copyright notices for more
            information.
14
15      Edited by Maria Vakalopoulou Sep 2011
            mariavakalopoulou@gmail.com
16
17  =========================================================================*/
18
19  #include "otbVectorImage.h"
20  #include "otbImage.h"
21  #include "otbImageFileReader.h"
22  #include "otbImageFileWriter.h"
23  #include "itkImageRegionIterator.h"
24  #include "itkImageRegionConstIterator.h"
25
26  //  Software Guide : BeginCommandLineArgs
27  //    INPUTS: {InputImage.tif}
28  //    OUTPUTS: {OutputImage1.tif}, {OutputImage2.tif}, {
        OutputImage_n.tif}
29
30  int main(int argc, char *argv[] )
31  {
32
33  // Define the dimension of the images
34    const unsigned int Dimension = 2;
35
36  // We start by declaring the types for the input image, wich is
        vector type
37  // and the output image, which is image type
38    typedef double                  PixelType;
39    typedef otb::VectorImage<PixelType, Dimension> VectorImageType
          ;
40    typedef otb::Image<PixelType, Dimension> ImageType;
41    typedef otb::ImageFileReader<VectorImageType> ReaderType;
42
```

```
43 // We declare the types for the iterators. For the vector type
       image,
44 //the iterator is const
45
46   typedef itk::ImageRegionConstIterator<VectorImageType>
       IteratorType2;
47   typedef itk::ImageRegionIterator<ImageType> IteratorType;
48
49 // We can now declare the type for the reader.
50   VectorImageType::ConstPointer inputImage;
51   ReaderType::Pointer  reader = ReaderType::New();
52   reader -> SetFileName(argv[1]);
53   try
54   {
55           reader -> Update();
56           inputImage = reader -> GetOutput();
57   }
58   catch (itk::ExceptionObject& err)
59   {
60           std::cout << "ExceptionObject caught !" << std::endl;
61           std::cout << err << std::endl;
62           return -1;
63   }
64
65 //We create the iterator for the input images and extract the
       number
66 //of the bands
67   IteratorType2 inputIt (inputImage, inputImage ->
       GetRequestedRegion());
68   unsigned int band = reader->GetOutput()->
       GetNumberOfComponentsPerPixel();
69   double pixelInput=0;
70
71 //For every band, the algorithm produce a new output image
72   for (unsigned int i=0; i<band-1; i++)
73   {
74 //We create the output image and its iterator by fill it 0
75           ImageType::Pointer outputImage = ImageType::New();
76       outputImage -> SetRegions(inputImage -> GetRequestedRegion
           ());
77       outputImage ->Allocate();
78       IteratorType outputIt (outputImage, outputImage ->
           GetRequestedRegion());
79
80       outputImage->FillBuffer(0);
81
82 //The iterators go to the beginning of the image that they
       declare
83
84           inputIt.GoToBegin();
85       outputIt.GoToBegin();
```

```
86
87 //We declare the type of the output pixel
88       PixelType pixelOutput;
89
90 //The prossess will continue until the iterator of the input
      image go to end
91           while (!inputIt.IsAtEnd())
92           {
93                     pixelInput = inputIt.Get()[i+1]-inputIt.Get
                          ()[i];
94                      pixelOutput = static_cast<PixelType>(
                           pixelInput);
95                outputIt.Set(pixelOutput);
96                ++inputIt;
97                ++outputIt;
98                pixelInput=0;
99            }
100
101
102 //We create the writer
103   typedef otb::ImageFileWriter<ImageType> WriterType;
104   WriterType::Pointer writer= WriterType::New();
105
106   writer->SetFileName(argv[2+i]);
107   writer->SetInput(outputImage);
108
109     try
110     {
111         writer->Update();
112     }
113     catch (itk::ExceptionObject& err)
114     {
115         std::cout << "ExceptionObject caught !" << std::endl;
116         std::cout << err << std::endl;
117        return -1;
118     }
119 }
120
121   return EXIT_SUCCESS;
122 }
```