ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ & ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
Τομέας Τοπογραφίας
Εργαστήριο Τηλεπισκόπησης

Ανάπτυξη διαδικτυακών εφαρμογών για την υποστήριξη τηλεπισκοπικών μεθόδων σε λογισμικό ανοιχτού κώδικα

Web applications development to support remote sensing methods in open source software

Διπλωματική εργασία

Παραγιουδάκης Λεωνίδας

Αθήνα, 2014

Ευχαριστίες

Ευχαριστώ τον καθηγητή κ. Αργιαλά για τη βοήθεια και τη συμπαράστασή του κατά τη διάρκεια εκπόνησης στη παρούσας εργασίας. Ευχαριστώ επίσης τον Άγγελο Τζώτσο για τις πολύτιμες συμβουλές του στον τομέα του γεωχωρικού λογισμικού ανοιχτού κώδικα.

Περίληψη

Η ανάπτυξη γεωχωρικού λογισμικού ανοιχτού κώδικα τα τελευταία χρόνια έχει γνωρίσει μεγάλη ανάπτυξη. Ο λόγος για αυτό είναι η ύπαρξη των οργανισμών OGC και OpenGeo, οι οποίοι έχουν προτυποποιήσει έναν τρόπο να παρουσιάζονται και να μοιράζονται τα γεωχωρικά δεδομένα στο διαδίκτυο. Καρπός αυτής της ανάπτυξης ήταν μεταξύ άλλων και ο Geoserver, ο οποίος ενσωματώνει GIS επεξεργασίες σε έναν χωρικό server κάτω από το μανδύα του server ανοιχτού κώδικα Tomcat. Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μεθόδων τηλεπισκόπησης σε γλώσσα Java και η χρήση αυτών των εφαρμογών για την επέκταση του Geoserver. Επίσης επιχειρείται να διερευνηθεί κατά πόσο αυτές οι μέθοδοι μπορούν να είναι αξιόπιστες και να συμβάλλουν στην δημιουργία ενός τηλεπισκοπικού σταθμού επεξεργασίας των δεδομένων, ο οποίος να αποτελεί επέκταση του Geoserver.

Πίνακας περιεχομένων

1. Εισαγωγή

Τα τελευταία χρόνια , με την επικράτηση και επέκταση του διαδικτύου και την προώθηση των γλωσσών $4^{ου}$ επιπέδου , αναπτύχθηκαν εφαρμογές γεωπληροφορικής οι οποίες έχουν σαν μοναδικό προαπαιτούμενο την ύπαρξη από πλευράς χρήστη ενός web browser. Η κοινότητα ελεύθερου λογισμικού έρχεται πρώτη να καλύψει αυτήν την ανάγκη με την κοινοποίηση του προβλήματος και την ορισμένη τυποποίηση του σε συγκεκριμένες βάσεις ώστε να αρχίσει από κοινού η διαδικασία υλοποίησης εφαρμογών .

Μία από αυτές τις βάσεις είναι η ίδρυση του οργανισμού OGC (Open Geospatial Consortium) . Η συνεισφορά του συγκεκριμένου οργανισμού είναι η ανάπτυξη προτύπων και διεπαφών καθώς και βελτιστοποιημένες τεχνικές , που επιτρέπουν την ανάπτυξη πληροφοριακών συστημάτων το αντικείμενο των οποίων είναι η γεωγραφική πληροφορία. Τα συστήματα αυτά μπορούν να ανταλλάσουν δεδομένα και διαδικασίες με άλλα συστήματα. Τα πιο βασικά πρότυπα του OGC που θα αναλυθούν στα
παρακάτω κεφάλαια είναι τα: WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service) , CSW (*Catalog* Service for the Web), WPS (Web Processing Service). Η ύπαρξη των ανοιχτών προτύπων είναι πολύ σημαντική αφού καθιστά εφικτή την ανταλλαγή δεδομένων και ευνοεί την περαιτέρω ανάπτυξη των εφαρμογών με βάση το μοντέλο του ελεύθερου λογισμικού .

Στην παρούσα διπλωματική γίνεται μία εκτενής αναφορά στο πρότυπο WPS και τη διαδικασία Web Processing . Γίνεται προσπάθεια επέκτασης του προτύπου αυτού για την υποστήριξη διαδικασιών τηλεπισκόπησης και αλυσιδωτή χρήση(chaining) των υπαρχόντων υπηρεσιών.

Η τηλεπισκόπηση ερευνά , μέσω δορυφορικών πολυφασματικών εικόνων κυρίως , φυσικά φαινόμενα όπως βλάστηση , ρύπανση , ανίχνευση ορίων και πολλά φαινόμενα της φυσικής γήινης επιφάνειας. Όλα τα προαναφερθέντα επιτυγχάνονται με συγκεκριμένες μεθόδους(ανίχνευση ορίων , ταξινόμηση εικόνας , λόγοι , έγχρωμα σύνθετα) οι οποίες γίνεται προσπάθεια να ενσωματωθούν στις υπάρχουσες διαδικασίες επεξεργασίας.

Στα επόμενα θα γίνει μία ανασκόπηση των επιστημών της τηλεπισκόπησης, της γεωπληροφορικής και της αρχιτεκτονικής των διαδικτυακών εφαρμογών .

1.1 Τηλεπισκοπικές εικόνες

Η Τηλεπισκόπηση είναι η επιστήμη της απόκτησης ποιοτικής και μετρητικής πληροφορίας ενός αντικειμένου από απόσταση, χωρίς φυσική επαφή . Αυτό γίνεται δυνατό, με τη χρήση ειδικών καταγραφέων της ακτινοβολίας που

αντανακλάται από όλα τα σώματα, οι οποίοι ονομάζονται δέκτες . Οι δέκτες χωρίζονται σε ενεργητικούς και παθητικούς δέκτες . Οι παθητικοί δέκτες καταγράφουν τη φυσική (ηλιακή) ακτινοβολία που αντανακλάται από όλα τα στερεά σώματα . Συνήθως το φάσμα της ακτινοβολίας καταγράφεται τμηματικά για διακριτά μήκη κύματος . Οι ενεργητικοί δέκτες καταγράφουν την ακτινοβολία που εκπέμπουν οι ίδιοι(Radar, Lidar, Sonar). Τα προϊόντα που παράγονται είναι οι εικόνες τηλεπισκόπησης για τις οποίες αναπτύχθηκαν οι διαδικασίες επεξεργασίας. Ακολουθεί η περιγραφή των εικόνων τηλεπισκόπησης .

Μία εικόνα είναι η καταγραφή σε πινακοποιημένη μορφή της προσπίπτουσας ακτινοβολίας . Έτσι σε κάθε στοιχείο του πίνακα (το οποίο ονομάζεται εικονοστοιχείο - pixel) αντιστοιχεί ένας συγκεκριμένος αριθμός .

100	203	198	195	…					
…									

Εικόνα 1.1.1: Πινακοποιημένη εικόνα(Raster)

Ο αριθμός των εικονοστοιχείων που χωρούν στην οριζόντια και κάθετη διάσταση της εικόνας, δηλαδή ο αριθμός εικονοστοιχείων ανά μονάδα μήκους ονομάζεται **διακριτική ικανότητα** . Με δεδομένες διαστάσεις της εικόνας, η διακριτική ικανότητα μας δίνει την διάσταση της εδαφοψηφίδας δηλαδή του τετραγώνου του εδάφους που συμβολίζει κάθε pixel .

Η **φασματική διακριτική ικανότητα** ή φασματική απόκριση αφορά το εύρος της ακτινοβολίας για την οποία γίνεται καταγραφή . Όπως αναφέρθηκε στα προηγούμενα, συνήθως οι δέκτες καταγράφουν την ακτινοβολία σε διακριτά κανάλια τα οποία αντιστοιχούν σε διαφορετικές περιοχές μήκους κύματος της ακτινοβολίας . Οι έγχρωμες εικόνες δημιουργούνται από την υπέρθεση τριών καναλιών εκ των οποίων το πρώτο ¨βάφεται¨ κόκκινο το δεύτερο πράσινο και το τρίτο μπλε .

Η **ραδιομετρική διακριτική ικανότητα** είναι η δυνατότητα που έχει ένας δέκτης να καταγράψει τις διαφορετικές τιμές έντασης της ακτινοβολίας . Ο αριθμός που αποθηκεύεται σε κάθε pixel ανήκει σε ένα συγκεκριμένο εύρος τιμών π.χ. από 0 έως 15 . Αυτό σημαίνει ότι το ανθρώπινο μάτι είναι σε θέση να ξεχωρίσει 16 τόνους του γκρι σε αυτήν την εικόνα . Όπως προαναφέρθηκε, αν αυτοί οι τόνοι του γκρι «βαφτούν» κόκκινοι , έχουμε το πρώτο στοιχείο μίας έγχρωμης εικόνας.

Τα προηγούμενα χαρακτηριστικά των εικόνων τηλεπισκόπησης χρησιμοποιούνται κατά τη διαδικασία της επεξεργασίας και ανάλυσης εικόνας.

1.2 Γεωγραφικά συστήματα πληροφοριών

Το **Σύστημα Γεωγραφικών Πληροφοριών** (ΣΓΠ), γνωστό ευρέως και ως G.I.S. Geographic Information Systems, είναι σύστημα διαχείρισης χωρικών δεδομένων (spatial data) και συσχετισμένων ιδιοτήτων . Στην πιο αυστηρή μορφή του είναι ένα ψηφιακό σύστημα, ικανό να ενσωματώσει, αποθηκεύσει, προσαρμόσει, αναλύσει και παρουσιάσει γεωγραφικά συσχετισμένες (geographically-referenced) πληροφορίες . Σε πιο γενική μορφή, ένα ΣΓΠ είναι ένα εργαλείο "έξυπνου χάρτη", το οποίο επιτρέπει στους χρήστες του να αποτυπώσουν μια περίληψη του πραγματικού κόσμου, να δημιουργήσουν διαδραστικά ερωτήσεις χωρικού ή περιγραφικού χαρακτήρα (αναζητήσεις δημιουργούμενες από τον χρήστη), να αναλύσουν τα χωρικά δεδομένα (spatial data), να τα προσαρμόσουν και να τα αποδώσουν σε αναλογικά μέσα (εκτυπώσεις χαρτών και διαγραμμάτων) ή σε ψηφιακά μέσα (αρχεία χωρικών δεδομένων, διαδραστικοί χάρτες στο Διαδίκτυο).

Τα συστήματα γεωγραφικών πληροφοριών ανοιχτού κώδικα ειδικότερα, χρησιμοποιούν την προτυποποίηση του OGC και εξελίσσονται από μία κοινότητα. Στην εξέλιξη μπορεί να συμμετέχει όποιος θέλει και τα προβλήματα που παρουσιάζονται επιλύονται από κοινού στα πλαίσια διαλόγου μεταξύ της κοινότητας .

1.3 GIS και Servers

Λόγω της ευρείας εξάπλωσης του διαδικτύου, κρίθηκε σκόπιμο από τους προγραμματιστές που δημιούργησαν τα GIS ανοιχτού κώδικα να τα ενσωματώσουν σε υλοποιημένες εφαρμογές server (π.χ. Tomcat) ώστε να προσδώσουν στις εφαρμογές αυτές χαρακτήρα server. Λόγω αυτού του γεγονότος, η δημιουργία σήμερα διαδικτυακών χωρικών εφαρμογών είναι σχετικά πιο εύκολη υπόθεση από ότι ήταν στο παρελθόν . Παλιότερα ένας προγραμματιστής έπρεπε, πριν κάνει οτιδήποτε άλλο, να σχεδιάσει στη web εφαρμογή που θα ανέπτυσσε συστήματα αποθήκευσης και επεξεργασίας των χωρικών δεδομένων . Δεδομένου ότι θα ανέπτυσσε μία εφαρμογή που

διαχειρίζεται χωρικά δεδομένα, θα έπρεπε να λάβει υπ όψη του το μέγεθος των δεδομένων (π.χ. μία τηλεπισκοπική εικόνα τύπου TIFF απαιτεί χωρητικότητα 200 MB) και τον χρόνο εκτέλεσης της εφαρμογής . Οι GIS servers έχουν λυμένα αυτά τα προβλήματα, καθώς συνήθως έχουν ενσωματωμένη βάση δεδομένων και βελτιστοποιημένα προγράμματα αποθήκευσης, ανάκτησης και επεξεργασίας.

Έτσι μένει για τον προγραμματιστή του σήμερα, το καθήκον να δημιουργήσει μία ευχάριστη οπτικοποίηση της εφαρμογής και έναν αποτελεσματικό τρόπο επικοινωνίας με τον GIS server . Αντικείμενο της παρούσας εργασίας είναι η επέκταση ενός τέτοιου GIS server ανοιχτού κώδικα , συγκεκριμένα του geoserver με προγράμματα επεξεργασίας τηλεπισκοπικών εικόνων.

2. Βιβλιογραφία

Στο παρόν κεφάλαιο θα γίνει μία αναφορά στην υπάρχουσα κατάσταση στο θέμα των GIS ανοιχτού κώδικα και κυρίως των διαδικτυακών. Επίσης θα γίνει εκτενής αναφορά στην υπηρεσία διαδικτυακής επεξεργασίας γεωπληροφορίας (WPS). Στο πρώτο υποκεφάλαιο θα γίνει αναφορά στις γενικές κατευθύνσεις και εξελίξεις που έχουν διαμορφωθεί στο χώρο των GIS ανοιχτού κώδικα. Στο δεύτερο υποκεφάλαιο θα γίνει πιο συγκεκριμένη αναφορά στο προγραμματιστικό περιβάλλον που επιλέχθηκε για την εφαρμογή της συγκεκριμένης εργασίας. Τέλος, στο τρίτο υποκεφάλαιο θα γίνει εκτενής περιγραφή των τηλεπισκοπικών μεθόδων και επεξεργασιών που εφαρμόστηκαν.

2.1 Γενικές κατευθύνσεις

Αναφέρεται εδώ ένας οργανισμός, μία μη-κερδοσκοπική εταιρεία και ένα λογισμικό με καθοριστικό ρόλο στην ανάπτυξη του διαδικτυακού GIS.

2.1.1 Ο οργανισμός Open Geospatial Consortium

Το Open Geospatial Consortium (OGC, http://www.opengeospatial.org/) είναι μια κοινοπραξία 388 εταιριών, κυβερνητικών υπηρεσιών και τα πανεπιστημίων που συμμετέχουν σε μια συναινετική διαδικασία ανάπτυξης προτύπων για τις GIS διεπαφές.

Παρακάτω αναφέρονται τα πρότυπα του OGC που χρησιμοποιήθηκαν και άλλα που δεν χρησιμοποιήθηκαν και είναι δυνατόν να αξιοποιηθούν σε μελλοντικές εφαρμογές.

- **WMS (Web Map Service)**

  Η υπηρεσία WMS είναι η συνηθέστερα χρησιμοποιούμενη υπηρεσία σε τέτοιου είδους γεωχωρικές εφαρμογές. Ο σκοπός που επιτελεί είναι η οπτικοποίηση οποιουδήποτε χαρακτηριστικού (π.χ. ένα shapefile ή μία τηλεπισκοπική εικόνα). Το πρότυπο WMS όπως και τα περισσότερα πρότυπα που υλοποιούνται στα διαδικτυακά GIS βασίζονται σε μία «συνεννόηση» του εξυπηρετούμενου (client) και του εξυπηρετητή (server). Ο client στέλνει ένα αίτημα (request) http στον server. Λόγω του ότι η συγκεκριμένη υπηρεσία δεν «προσπαθεί» να αλλάξει κάποια δεδομένα στο σύστημα, το αίτημα που αποστέλλεται είναι τύπου GET. Ο server από την πλευρά του λαμβάνει αυτό το αίτημα, το αποκωδικοποιεί και στέλνει την απάντηση(response). Η υπηρεσία WMS περιλαμβάνει τρείς λειτουργίες. Αυτές είναι:

GetCapabilities: Σε αυτήν τη λειτουργία το αίτημα περιλαμβάνει τις παραμέτρους και την έκδοση της υπηρεσίας και η απάντηση είναι τα μεταδεδομένα τα οποία χρησιμοποιεί η υπηρεσία και οι αποδεκτές παράμετροι κατά την κλήση της υπηρεσίας

GetMap: Σε αυτήν τη λειτουργία ορίζουμε στο αίτημα μας το όνομα του αντικειμένου και τις παραμέτρους με τις οποίες θέλουμε να απεικονιστεί αυτό και η απάντηση είναι ο χάρτης του αντικειμένου με τις επιλεγμένες παραμέτρους προβολής (π.χ. σύστημα συντεταγμένων ή zoom)

GetFeatureInfo: Σε αυτό το αίτημα ορίζουμε την περιοχή ενδιαφέροντος π.χ. ένα pixel ή ένας δρόμος και η απάντηση είναι τα χαρακτηριστικά ατού του στοιχείου (π.χ. τιμή χρώματος ή όνομα δρόμου αντίστοιχα)

- **WFS (Web Feature Service)**

Η υπηρεσία αυτή ορίζει τις λειτουργίες για την αποθήκευση, ανάκτηση και τροποποίηση των διανυσματικών γεωχωρικών δεδομένων. Όπως και η προηγούμενη υπηρεσία, βασίζεται στη συνεργασία client-server. Διαμορφώνονται έτσι κάποιες λειτουργίες όπως στην υπηρεσία WMS. Αυτές οι υπηρεσίες, λόγω της μεγαλύτερης πολυπλοκότητάς τους είναι πολύ πιο ειδικές και συγκεκριμένες, γι' αυτό και πιο πολλές σε αριθμό. Επίσης λόγω του ότι στη συγκεκριμένη υπηρεσία τροποποιούνται χαρακτηριστικά κάποιων στοιχείων του server, συγκροτείται με αιτήματα POST και GET. Ορισμένες από τις λειτουργίες είναι: ***GetCapabilities, GetFeature, DescribeFeature*** και άλλες.

- **WCS (Web Coverage Service)**

Η υπηρεσία αυτή ορίζει τις λειτουργίες για την ανάκτηση και περιγραφή raster δεδομένων π.χ. αεροφωτογραφιών. Λόγω του μεγάλου αριθμού μεταδεδομένων που απαιτούν τα raster δεδομένα, Η συγκεκριμένη υπηρεσία υλοποιείται με POST αιτήματα. Αυτή η υπηρεσία περιλαμβάνει τις εξής λειτουργίες:

DescribeCoverage: Υποβολή ενός POST αιτήματος στο οποίο δίνεται σαν δεδομένο το όνομα του raster στοιχείου που ζητούνται τα μεταδεδομένα. Η απάντηση του server είναι ένα αρχείο XML στο οποίο περιγράφονται με σαφήνεια όλα τα μεταδεδομένα για αυτό το στοιχείο (π.χ. όνομα, σύστημα συντεταγμένων, γεωγραφικές συντεταγμένες και άλλα)

GetCapabilities: Σε αυτήν τη λειτουργία το αίτημα περιλαμβάνει τις παραμέτρους και την έκδοση της υπηρεσίας και η απάντηση είναι τα μεταδεδομένα τα οποία χρησιμοποιεί η υπηρεσία και οι αποδεκτές παράμετροι κατά την κλήση της υπηρεσίας.

GetCoverage: Σε αυτήν τη λειτουργία υποβάλλεται ένα POST αίτημα στον server και η απάντηση είναι μία γεωαναφερμένη εικόνα συνήθως σε format TIFF.

Αξίζει εδώ να αναφέρουμε ότι οι διάφορες υπηρεσίες και λειτουργίες μπορούν να προηγηθούν η μια της άλλης, εφόσον φυσικά τα δεδομένα εξόδου της μιας αντιστοιχούν στα δεδομένα εισόδου της άλλης. Με αυτόν τον τρόπο μπορούμε υποβάλλοντας ένα και μόνο αίτημα στο server να πάρουμε ένα αποτέλεσμα που θα χρειαζόταν, σε άλλη περίπτωση, πολλαπλή επεξεργασία.

- **WPS (Web Processing Service)**

Η υπηρεσία WPS είναι μία υπηρεσία που έχει να κάνει με την επεξεργασία των δεδομένων που είναι αποθηκευμένα στο server. Η υπηρεσία αυτή λαμβάνει σαν δεδομένα εισόδου σχεδόν οτιδήποτε (π.χ. αριθμούς, αρχεία shapefiles, εικόνες) τα επεξεργάζεται και επιστρέφει τα δεδομένα εξόδου. Λόγω της πολυπλοκότητας των εισαγόμενων στοιχείων και των πολλών μεταδεδομένων που απαιτούνται, τα αιτήματα προς αυτήν την υπηρεσία είναι τύπου POST. Πιο συγκεκριμένα όσον αφορά τον Geoserver, η υπηρεσία WPS υποστηρίζεται σαν επέκταση του βασικού πακέτου. Επιπλέον ενσωματώνει μεγάλο αριθμό λειτουργιών σχετικών με επεξεργασία διανυσματικών και ψηφιδωτών δεδομένων. Αξίζει να σημειωθεί ότι μεγάλος αριθμός WPS λειτουργιών προέρχεται από τη βιβλιοθήκη GeoTools που αναφέρθηκε στα προηγούμενα. Επίσης μεγάλος αριθμός λειτουργιών προέρχεται από τη βιβλιοθήκη JTS η οποία είναι μία βιβλιοθήκη αντίστοιχη με τη GeoTools, αλλά αποκλειστικά για διανυσματικά δεδομένα.

Δυστυχώς, στο πακέτο αυτό δεν περιλαμβάνονται λειτουργίες σχετικές με τις τηλεπισκοπικές μεθόδους. Η παρούσα εργασία διερευνά τη δυνατότητα επέκτασης της υπηρεσίας WPS με λειτουργίες σχετικές με τηλεπισκοπικές μεθόδους.

- **SLD (Style Layer Descriptor)**

Το πρότυπο SLD του OGC αφορά στον τρόπο απόδοσης και οπτικοποίησης των γεωχωρικών δεδομένων. Το πρότυπο αυτό ορίζει μία καινούργια γλώσσα σήμανσης που βασίζεται στην XML. Η γλώσσα αυτή περιέχει ετικέτες για κάθε πιθανό χαρακτηριστικό όπως π.χ. χρώμα γραμμής, πάχος γραμμής, χρωματική απόδοση pixel και πολλά άλλα. Όλα τα χαρακτηριστικά που προαναφέρθηκαν μπορούν να περικλείονται από ένα αρχικό ζεύγος ετικετών οι οποίες ορίζουν σε τι αντικείμενο αναφερόμαστε. Αυτές οι ετικέτες είναι οι παρακάτω:
Point: Αναφερόμαστε σε σημείο
Line: Αναφερόμαστε σε γραμμή

Polygon: Αναφερόμαστε σε πολύγωνο
Raster: Αναφερόμαστε σε ψηφιακή μορφή δεδομένων

2.1.2 Ο οργανισμός OpenGeo

Η OpenGeo είναι ένας νέου τύπου οργανισμός, με υβριδικό χαρακτήρα σε ότι αφορά το μη κερδοσκοπικό χαρακτηριστικό της. Ο στόχος της είναι να κάνει πιο ανοικτή τη γεω-χωρική πληροφορία, αναπτύσσοντας (δωρεάν) λογισμικό ανοιχτού κώδικα και συμμετέχοντας στην κοινότητα ανοιχτού GIS λογισμικού. Δεν στηρίζεται ωστόσο στην εθελοντική εργασία για την ανάπτυξη του λογισμικού, αλλά προσφέρει στα μέλη της ανταγωνιστικές αμοιβές με τις αντίστοιχες εμπορικές επιχειρήσεις. Για την εξασφάλιση των πόρων ανταγωνίζεται στην αγορά για συμβάσεις έργων, όπως μια παραδοσιακή εταιρεία. Η μη κερδοσκοπική δέσμευσή της εξασφαλίζει την επανεπένδυση του συνόλου της κερδοφορίας της για την επίτευξη των σκοπών της.

Η OpenGeo είναι η μετεξέλιξη της ομάδας που ξεκίνησε την ανάπτυξη του Geoserver το 2001 υπό την χρηματοδότηση της The Open Planning Project (μίας επίσης μη κερδοσκοπικής τεχνολογικής επιχείρησης με στόχο την ανάπτυξη και ενσωμάτωση τεχνολογιών από τις κυβερνήσεις για την αύξηση της διαφάνειας στις λειτουργίες του κράτους) . Συμμετέχει ενεργά στην ανάπτυξη των GeoServer, PostGIS και OpenLayers.

2.1.3 Η βιβλιοθήκη GeoTools

Η GeoTools είναι μια open source (LGPL) βιβλιοθήκη Java κώδικα, που δίνει μεθόδους συμβατές με τα OGC πρότυπα για τον χειρισμό των γεω-χωρικών δεδομένων και την ανάπτυξη GIS εφαρμογών. Είναι ένα από τα πρώτα έργα της OSGeo, και έχει χρησιμοποιηθεί σαν υποδομή για την ανάπτυξη εφαρμογών όπως το uDig και ο Geoserver.

2.1.4 Υλοποιημένες εφαρμογές

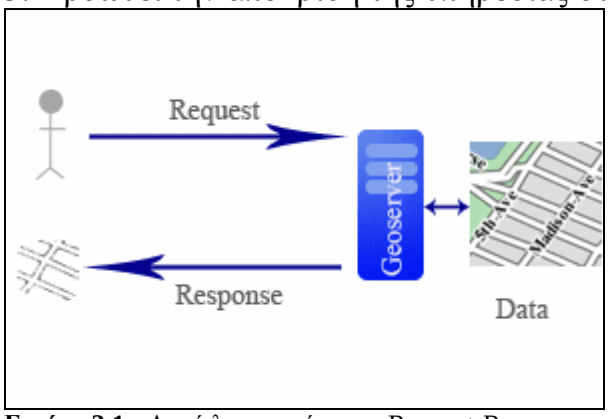
Τα παραπάνω πρότυπα και οργανισμοί έχουν οδηγήσει σε κάποιες υλοποιήσεις λογισμικών και εφαρμογών οι οποίες απαριθμούνται παρακάτω:

GeoServer

Ο **Geoserver** είναι ένας server για τη προβολή και το χειρισμό χωρικών δεδομένων στο διαδίκτυο. Ξεκίνησε το 2001 από την The Open Planning Project. Η εφαρμογή έχει αναπτυχθεί σε Java, ενώ έχει χτισθεί πάνω στη βιβλιοθήκη GeoTools. Χρησιμοποιεί τα ανοιχτά πρότυπα της Open Geospatial Consortium (OGC). Είναι ελεύθερο λογισμικό ανοιχτού κώδικα (**GNU General Public License).** Η ανάπτυξή του καθοδηγείται από την GIS open-source κοινότητα (community- driven). Ένα από τα πολύ δυναμικά χαρακτηριστικά του είναι ότι δίνεται στους χρήστες η δυνατότητα επέκτασης του με τη γνώση και τη χρήση συγκεκριμένων κλάσεων και αντικειμένων από τη βιβλιοθήκη GeoTools.

Ο GeoServer όπως και οι περισσότεροι χωρικοί και μη server «επικοινωνεί» με όσους θέλουν να τον χρησιμοποιήσουν με το πρωτόκολλο HTTP . Η βάση αυτής της επικοινωνίας είναι η αντιστοίχηση σε κάθε αίτημα του χρήστη μίας απάντησης του server . Τα αιτήματα αυτά όπως περιγράφηκε και στα προηγούμενα είναι τύπου POST και GET (κατά την επικύρωση του ονόματος και του κωδικού πρόσβασης ενός χρήστη χρησιμοποιείται και η μέθοδος OPTIONS). Το request-response σύστημα ,

1. δέχεται το αίτημα του πελάτη
2. το προωθεί στην υπηρεσία (WMS, WFS , WCS , WPS) στην οποία απευθύνεται
3. Προωθεί την απόκριση της υπηρεσίας στον πελάτη.



Εικόνα 2.1 : Αρχή λειτουργίας του Request-Response

Ένα σημαντικό χαρακτηριστικό του GeoServer είναι ότι εξυπηρετεί δεδομένα από πολλές πηγές δεδομένων. Όσον αφορά τα διανυσματικά εξυπηρετεί Shapefiles, εξωτερικά WFS, PostGIS, ArcSDE, Oracle Spatial, MySql, SQL Server. Όσον αφορά τα εικονιστικά εξυπηρετεί GeoTiff, JPG, PNG, πυραμίδες εικόνων, πρότυπα της βιβλιοθήκης GDAL και μωσαϊκά εικόνων. Παρακάτω παρουσιάζονται οι πηγές από τις οποίες μπορεί να διαβάσει δεδομένα ο GeoServer :

- Vector μορφές δεδομένων

    - Shapefiles
    - PostGIS βάσεις δεδομένων
    - Εξωτερικά WFS layers
    - Java Properties files
- Raster μορφές δεδομένων
    - ArcGrid
    - GeoTIFF
    - Gtopo30
    - ImageMosaic
    - WorldImage

Ένα από τα σημαντικότερα χαρακτηριστικά του είναι η δυνατότητα που έχει για άμεση γεωαναφορά όλων των προηγούμενων στοιχείων (διανυσματικά και ψηφιογραφικά). Επίσης δίνεται η δυνατότητα επιλογής συστήματος αναφοράς από ένα ενσωματωμένο κατάλογο.

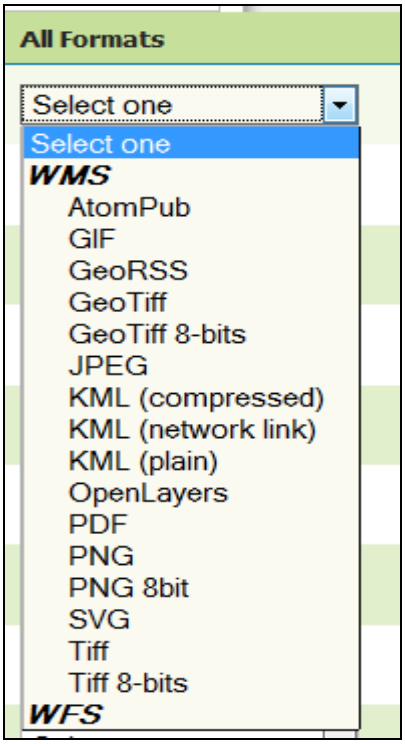
Επίσης, ένα πολύ δυναμικό στοιχείο του Geoserver είναι δυνατότητα οπτικοποίησης των στοιχείων που εισάγονται μέσω της γλώσσας SLD που αναφέρθηκε στα προηγούμενα. Αυτό είναι σημαντικό διότι η γλώσσα SLD είναι μια γλώσσα «κατανοητή» από όλες τις εφαρμογές ανοιχτού κώδικα. Έτσι , είναι δυνατό να συνταχθεί το αρχείο SLD σε άλλη εφαρμογή (π.χ. uDig) και στη συνέχεια να εισαχθεί στον GeoServer και να συνδεθεί με κάποια πηγή δεδομένων, μεταβάλλοντας έτσι τον τρόπο που αυτή εμφανίζεται.

Ένα άλλο στοιχείο που διαφοροποιεί τον GeoServer σε σχέση με άλλα GIS είναι η δυνατότητα που έχει για διατύπωση και εξυπηρέτηση ερωτημάτων. Αυτό είναι δυνατό σε όλες τις υπηρεσίες του (WMS, WFS, WCS, WPS) και γίνεται κυρίως με τη χρήση των φίλτρων του OGC είτε με τη γλώσσα CQL. Αυτά τα ερωτήματα μπορούν να επιστρέφουν για παράδειγμα το shapefile με το μεγαλύτερο πληθυσμό ή ένα συγκεκριμένο γεωαναφερμένο κομμάτι μίας εικόνας.

Στα προηγούμενα έγινε μία περιγραφή των υπηρεσιών του OGC που υλοποιούνται στον GeoServer. Θα ακολουθήσει μία περιγραφή των δεδομένων εξόδου :

- Υπηρεσία WMS

  Υποστηρίζονται οι μορφές εξόδου : AtomPub , GIF , GeoRSS, GeoTiff, JPEG, KML, OpenLayers, PDF, PNG, SVG, Tiff



Εικόνα 2.2 : Μορφές εξόδου της υπηρεσίας WMS

- Υπηρεσία WFS

Υποστηρίζονται οι μορφές εξόδου : CSV, GML ,GeoJSON ,KML ,Shapefile

Εικόνα 2.3 : Μορφές εφόδου υπηρεσίας WFS

- Υπηρεσία WCS

Στην υπηρεσία WCS υποστηρίζονται όλα τα βασικά format εικόνας δηλαδή TIFF, PNG, JPEG, BMP.

- Υπηρεσία WPS

Όπως περιγράφηκε και στα προηγούμενα, η υπηρεσία WPS μπορεί να έχει σαν εξόδους σχεδόν τα πάντα. Στην υπηρεσία αυτή υπάρχουν λειτουργίες σχεδόν για τα πάντα και όλες οι μορφές δεδομένων μπορούν να

μετατραπούν σε άλλες μορφές. Έτσι , στην υπηρεσία αυτή οι έξοδοι μπορούν να πάρουν όλες τις παραπάνω μορφές. Η υπηρεσία αυτή αναλύεται στο επόμενο κεφάλαιο διεξοδικότερα.

MapServer

Ο MapServer είναι μια Open Source (MIT license) πλατφόρμα για τη δημοσίευση των χωρικών δεδομένων στο Web. Γραμμένος σε C, η ανάπτυξή του ξεκίνησε στα μέσα της δεκαετίας του 1990 στο Πανεπιστήμιο της Μινεσότα(UMN) σε συνεργασία με τη NASA και το γραφείο φυσικών πόρων της Μινεσότα. Τρέχει σε όλες τις κύριες πλατφόρμες (Windows, Linux, Mac OS/X).

Ο MapServer απέχει παρασάγγας από το χαρακτηρισμό του ως πλήρες διαδικτυακό GIS. Η δουλειά για την οποία σχεδιάστηκε είναι κυρίως η προβολή χαρτών με μεγάλη πιστότητα. Σε αυτό το σημείο υπερέχει του Geoserver.

OpenLayers

Η OpenLayers είναι μία βιβλιοθήκη της JavaScript προσανατολισμένη στη δημιουργία διαδικτυακών διεπαφών με χωρικούς server (π.χ. GeoServer) και βάσεις δεδομένων. Η OpenLayers είναι ένα Framework της JavaScript, το οποίο χρησιμοποιεί την τεχνική Ajax για να διευκολύνει τους προγραμματιστές να δημιουργήσουν διαδικτυακές GIS εφαρμογές. Η βιβλιοθήκη αυτή περιλαμβάνει αρκετές κλάσεις που επιτρέπουν το χειρισμό όλων των υπηρεσιών του OGC και άλλα που επιτρέπουν την οπτικοποίηση και επεξεργασία τους. Τα βασικότερα από αυτά είναι:

- Map : Η κλάση αυτή κατασκευάζει ένα χάρτη δεχόμενη σαν όρισμα κάποιες παραμέτρους που θέλουμε να έχει αυτός π.χ. σύστημα συντεταγμένων ή χειριστήρια
- Layer : Η κλάση αυτή προσθέτει διαφορετικά θεματικά επίπεδα στο χάρτη που έχουμε κατασκευάσει. Είναι μία πολύ δυναμική κλάση καθώς δέχεται σαν όρισμα πολλούς τύπους δεδομένων όπως π.χ. WMS εξαγόμενα, κανονικές εικόνες, αποτελέσματα επεξεργασίας, shapefiles και πολλά άλλα.
- Controls : Η κλάση αυτή περιέχει όλα τα απαραίτητα για την κατασκευή χειριστηρίων (π.χ. zoom, draw, pan)
- Protocol-Renderer-Request : Κλάσεις που επιτρέπουν τον έλεγχο σε δεδομένα εξόδου WMS, WFS, WPS.

Qgis

Το Quantum GIS (QGIS) είναι ένα μη διαδικτυακό GIS ανοιχτού κώδικα (GNU). Είναι ένα προϊόν του OSGeo (Open Source Geospatial Foundation). Αρχικά αναπτύχθηκε με εθελοντική εργασία και χρηματοδότηση από συνεισφορές (contributions). Η ανάπτυξή του καθοδηγείται πλέον από την open-source GIS κοινότητα. Τρέχει σε Linux, Unix, Mac OSX, και Windows. Η ανάπτυξή του ξεκίνησε τον Μάιο του 2002 και η πρώτη έκδοσή του υποστήριζε μόνο PostGIS layers. Με το QGIS μπορεί κανείς να απεικονίσει, διαχειριστεί, αναλύσει, συντάξει χωρικές πληροφορίες καθώς και να δημιουργήσει εκτυπώσιμους χάρτες.

Το πιο δυναμικό χαρακτηριστικό του Qgis είναι η δυνατότητα επέκτασης του με υπάρχον βελτιστοποιημένο λογισμικό γεωχωρικού σκοπού (π.χ. GDAL ή Python Scripts και πολλά άλλα) που του προσδίδει αυξημένες δυνατότητες.

Udig

Το uDig είναι ένα λογισμικό ανοιχτού κώδικα για την επεξεργασία / επισκόπηση χωρικών δεδομένων, με έμφαση στα πρότυπα του OpenGIS για το διαδικτυακό GIS, το πρότυπο Web Map Service (WMS) και το πρότυπο Web Feature Service (WFS).

Το uDig είναι:

- **Φιλικό πρός το χρήστη**, παρέχοντας ένα γνώριμο γραφικό περιβάλλον για τους χρήστες του GIS,
- **Λογισμικό που εκτελείται τοπικά**, εκτελείται εγγενώς στα Windows, Mac OS/X και Linux,
- **Προσανατολισμένο στο διαδίκτυο**, και χρησιμοποιεί καθιερωμένα πρότυπα και διαδικτυακές υπηρεσίες, και
- **Έτοιμο για GIS**, παρέχοντας το πλαίσιο στο οποίο μπορούν να υλοποιηθούν σύνθετες δυνατότητες ανάλυσης, και στη συνέχεια να ενσωματωθούν στην κύρια εφαρμογή

Για τους προγραμματιστές, το uDig παρέχει μια πλατφόρμα Java για υλοποίηση γεωχωρικών εφαρμογών με εργαλεία ανοιχτού κώδικα. Η κεντρική

ιστοσελίδα του παρέχει μια σειρά από εύκολους οδηγούς εκμάθησης που καλύπτουν από τη χρήση ενός απλού μέχρι την υλοποίηση μιας ιδιοποιημένης εφαρμογής.

Το τελευταίο χαρακτηριστικό του uDig είναι από τα πιο δυναμικά και πολλές εφαρμογές ειδικού σκοπού έχουν υλοποιηθεί με βάση αυτήν . Άλλα πολύ δυναμικά χαρακτηριστικά του είναι :

- Λογισμικό για σταθμό εργασίας με δυνατότητες Drag and Drop για τον διαχειριστή αρχείων αλλά και τον περιηγητή διαδικτύου
- Διασυνδέεται με την υπάρχουσα υποδομή: ArcSDE, Oracle, DB2 και περισσότερα
- Εργασία σε τοπικά αρχεία: Shapefile, jpeg, png, tiff και περισσότερα.
- Λειτουργεί με προχωρημένα πρότυπα εικόνων: ECW, MrSID, JPEG 2000
- Παρέχει υποστήριξη για συμβατούς εξυπηρετητές χαρτών (δοκιμασμένο με GeoServer και MapServer)
- Ενσωματώνει την διαδικτυακή εμπειρία με εσωτερικό περιηγητή που αναγνωρίζει τα πρότυπα OGC και επιτρέπειτην προσθήκη συνδέσμων διαδικτύου στον χάρτη της οθόνης
- Σύνθεση προτύπων χαρτοσύνθεσης Style Layer Descriptor και οπτικοποίηση τους ώστε να μπορείτε να δημοσιεύσετε τους δικούς σας χάρτες χρησιμοποιώντας ρυθμίσεις στο uDig όμοιες με αυτές των δημοφιλών εξυπηρετητών χαρτών
- Βαθιά ενσωμάτωση προτύπων, επιτρέπει στην εφαρμογή να εναλλάσεται μεταξύ διαφορετικών προτύπων για την εύρεση του καταλληλότερου σχετικά με την απαραίτητη οπτικοποίηση, ολοκλήρωση και επεξεργασία
- Εκτύπωση και δημιουργία PDF
- Για προγραμματιστές
- Εφαρμογή Java υλοποιημένη με τις βιβλιοθήκες GeoTools, JTS Topology Suite (JTS) και GeoAPI
- Επαγγελματικές δυνατότητες που παρέχονται από την πλατφόρμα Eclipse Rich Client Platform με τη δυνατότητα πρόσθετων λειτουργιών (add-ons)
- Χρήση εγγενών παραθυρικών διεπαφών

Τέλος το μεγάλο πλεονέκτημα του uDig είναι η δυνατότητα μεταφοράς του τρόπου οπτικοποίησης ενός τύπου δεδομένων (π.χ. raster) σε άλλες εφαρμογές ανοιχτού κώδικα, μέσω της δυνατότητας μεταφοράς του αρχείου SLD.

Παράλληλα με τη χρήση των παραπάνω εφαρμογών και το συνδυασμό τους με άλλες τεχνολογίες έχουν προκύψει ολοκληρωμένα διαδικτυακά GIS τα σημαντικότερα από τα οποία είναι:

Geodjango

Το Geodjango είναι μια διαδικτυακή πλατφόρμα, γραμμένη σε γλώσσα Python, η οποία έχει σκοπό τη διευκόλυνση της ανάπτυξης διαδικτυακών γεωχωρικών εφαρμογών. Αποτελεί επέκταση του Django και ειδικεύεται στις γεωχωρικές εφαρμογές. Η αρχιτεκτονική του φαίνεται παρακάτω :

Εικόνα 2.4 : Αρχιτεκτονική της πλατφόρμας

Η θέση του Geodjango υποδηλώνει τη μεσολάβηση του ανάμεσα στις client και τις server εφαρμογές . Έτσι , η μεριά του πελάτη (client) χρησιμοποιεί την βιβλιοθήκη OpenLayers ενσωματωμένη στις HTML σελίδες και το σύστημα του Geodjango αναλαμβάνει να προωθήσει τα request των πελατών στις server εφαρμογές π.χ. Geoserver ή PostGIS . Αντίστοιχα, αναλαμβάνει να πάρει τα response των server και να τα οπτικοποιήσει χρησιμοποιώντας και πάλι την OpenLayers. Υποστηρίζει τα πρότυπα WMS, WFS και WCS κυρίως.

Cartaro

Το Cartaro είναι μία διαδικτυακή πλατφόρμα GIS εφαρμογών ανάλογη του Geodjango. Το Cartaro περιλαμβάνει και αυτό μερικές από τις πιο επιτυχημένες εφαρμογές των GIS ανοιχτού κώδικα όπως PostGIS, GeoServer, GeoWebCache, OpenLayers. Γραμμένο σε PHP, ενσωματώνει τις παραπάνω εφαρμογές με την γνωστή πλατφόρμα Drupal με στόχο την ευκολία και την απλότητα στο σχεδιασμό ιστοσελίδων και την ευκολία διαχείρισης περιεχομένων. Μπορεί να διαχειριστεί σχεδόν όλα τα πρότυπα του OGC. Κάποια από τα χαρακτηριστικά του είναι :

- Εύκολη γεωαναφορά γεωμετρικών στοιχείων που εισάγονται
- Δημιουργία πρωτότυπων γεωμετρικών στοιχείων μέσα από τη διεπαφή που προσφέρεται

- Εύκολη δημοσίευση χαρτών
- Κατασκευή υπομνημάτων
- Εύκολη δημιουργία συμβόλων με τη γλώσσα SLD
- Δημοσίευση δεδομένων εξόδου από WMS, WFS
- Εύκολη συλλογή μεταδεδομένων για τα γεωμετρικά στοιχεία που εισάγονται
- Δυνατότητα επέκτασης του βασικού πακέτου με άλλες υπηρεσίες

Τα παραπάνω χαρακτηριστικά φαίνονται στο παρακάτω σχήμα

Εικόνα 2.5 : Η αρχιτεκτονική του Cartaro

GeoNode

GeoNode

Το GeoNode είναι μια διαδικτυακή υλοποίηση χρήσης και ανάπτυξης Γεωγραφικών Συστημάτων Πληροφοριών (GIS) και χωρικών δομών δεδομένων ανοιχτού κώδικα. Μπορεί να χρησιμοποιηθεί από απλούς χρήστες αλλά και από προγραμματιστές όπου βασισμένοι σε αυτό, να αναπτύξουν ένα ισχυρό γεωγραφικό σύστημα πληροφοριών βασισμένο σε αυτό.

ZooProject

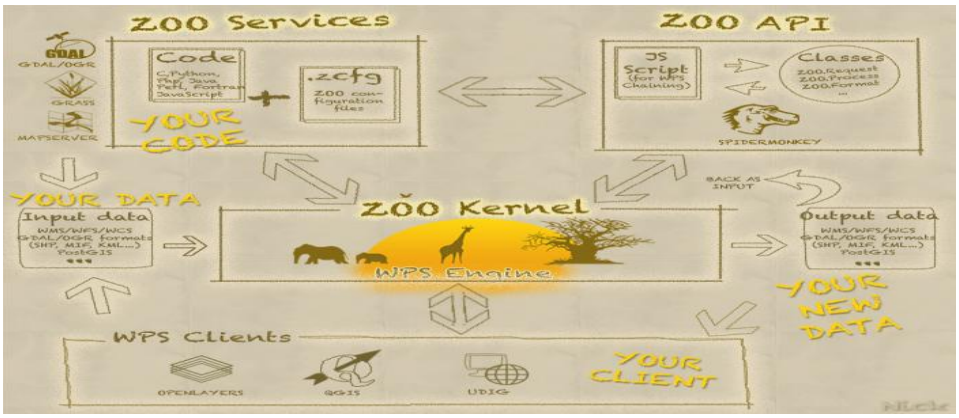
Το ZooProject είναι μία διαδικτυακή πλατφόρμα παρόμοια με τις προηγούμενες που παρουσιάστηκαν με διαφορετικό στοιχείο ότι «εξειδικεύεται» στην υπηρεσία WPS του OGC . Αποτελεί ένα φιλικό προς τον χρήστη και τον προγραμματιστή πλαίσιο για τη δημιουργία και το συνδυασμό των WPS υπηρεσιών. Παράλληλα αποτελεί και ένα πλαίσιο διαχείρισης περιεχομένου, όπως τα προηγούμενα. Τα δομικά χαρακτηριστικά του ZooProject είναι :

- Ο πυρήνας του : είναι γραμμένος σε γλώσσα C αλλά υποστηρίζει αρκετές ακόμα γλώσσες προγραμματισμού, με σκοπό τη δυνατότητα σύνδεσης σε μεγάλο αριθμό βιβλιοθηκών. Σκοπός του είναι να διευκολύνει τη δυνατότητα παροχής client WPS υπηρεσίες και να απλοποιήσει τη δουλειά του προγραμματιστή

- Οι λειτουργίες του : Είναι μία αναπτυσσόμενη σουίτα παραδειγμάτων, διαδικτυακών WPS υπηρεσιών βασισμένων σε ανοιχτού κώδικα βιβλιοθήκες. Μία υπηρεσία του ZOO-Project, συντίθενται από ένα αρχείο μεταδεδομένων (.zcfg) και από το κώδικα της αντίστοιχης εφαρμογής. Το αρχείο μεταδεδομένων περιγράφει όλες τις διαθέσιμες συναρτήσεις, οι οποίες μπορούν να καλεστούν χρησιμοποιώντας ένα αίτημα WPS εκτέλεσης, αλλά και τα επιθυμητά δεδομένα εσόδου/εξόδου. Οι υπηρεσίες περιλαμβάνουν αλγορίθμους και συναρτήσεις και μπορούν να υλοποιηθούν στις γλώσσες προγραμματισμού C / C++, Fortran, Java, Python, PHP, Perl και JavaScript.

- Το περιβάλλον διεπαφής: Ένα περιβάλλον διεπαφής σε γλώσσα JavaScript από τη μεριά του server, το οποίο καθιστά ευκολότερη την ανάπτυξη των αλυσιδωτών επεξεργασιών.

Η αρχιτεκτονική του ZooProject φαίνεται παρακάτω:



Εικόνα 2.6 : Η αρχιτεκτονική του ZooProject

2.2 Προγραμματιστικό Περιβάλλον

Στο παρόν υποκεφάλαιο θα γίνει αναφορά στα συγκεκριμένα εργαλεία και πρότυπα που χρησιμοποιήθηκαν προκειμένου να επιτευχθεί ο σκοπός της παρούσας εργασίας. Γίνεται μία αναφορά στη γλώσσα Java και τον αντικειμενοστραφή χαρακτήρα της. Έπειτα αναλύονται τα διάφορα πλαίσια εργασίας (Frameworks) και ειδικά το Spring Framework. Το τελευταίο εξειδικεύεται στην περίπτωση του GIS που μας αφορά (GeoServer). Στα επόμενα αναλύεται ο τρόπος με τον οποίο το Spring Framework χειρίζεται το πρότυπο WPS στον Geoserver. Επίσης γίνεται μία αναφορά στη βιβλιοθήκη GeoTools και πώς αυτή ενσωματώνεται στον Geoserver. Τέλος αναλύεται ο τρόπος με τον οποίο μπορεί ο κάθε χρήστης να επεκτείνει τον GeoServer.

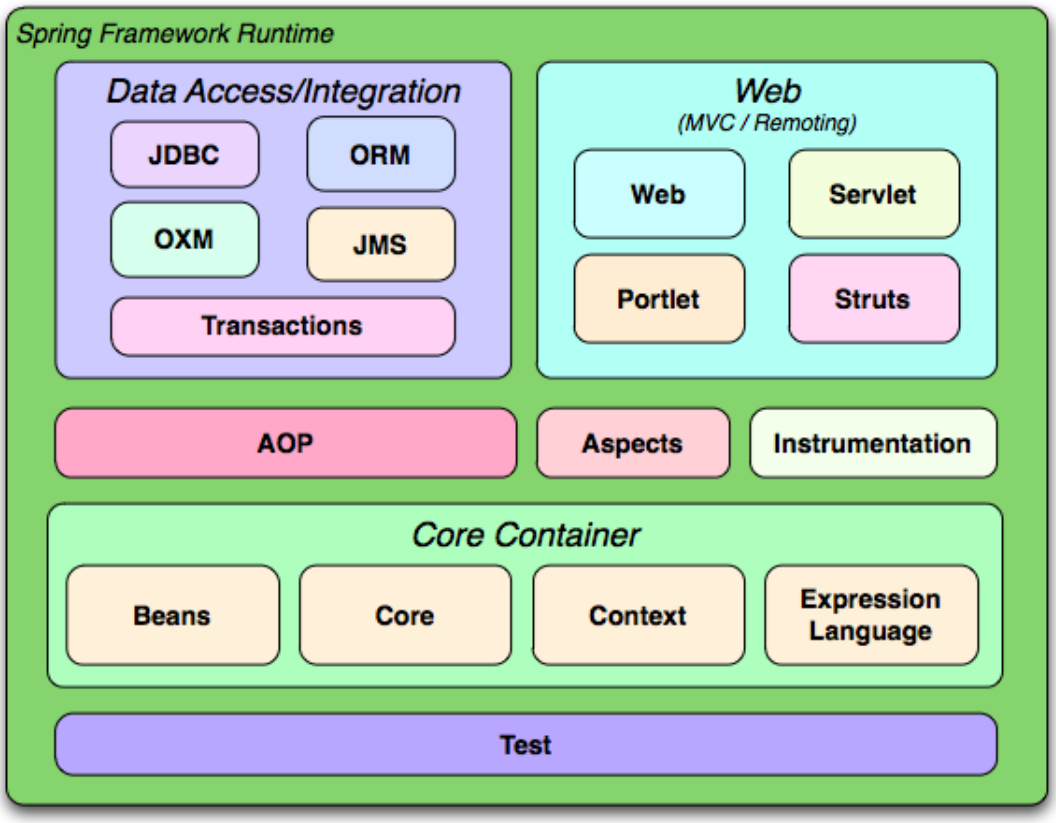
2.2.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής *Sun Microsystems*.
Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε *Java* τρέχουν ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (σύντομα θα τρέχουν και σε Playstation καθώς και σε άλλες κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος (Windows, Unix, Linux, BSD, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο συμβολικός κώδικας (*assembly*) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της *Εικονικής Μηχανής* (*Virtual Machine* ή VM ή ΕΜ στα ελληνικά).

2.2.2 Spring Framework

Το Spring Framework είναι μία εφαρμογή διαχείρισης καθηκόντων. Ο πυρήνας του είναι γραμμένος σε Java. Ο σκοπός της ανάπτυξής του είναι να διευκολύνει την ανάπτυξη εφαρμογών σε σύγχρονα προγραμματιστικά περιβάλλοντα και την απλοποίηση των αναπτυσσόμενων εφαρμογών. Το Spring Framework ήρθε να αποτελέσει μία εναλλακτική και ένα συμπλήρωμα στο EnterpriseJavaBean model.
Το Spring Framework παρέχει τις παρακάτω ενότητες για τη διαχείριση εργασιών:

- Μηχανισμός αντιστροφής του ελέγχου και ολοκληρωμένης διαχείρισης των αντικειμένων της Java. Με αυτόν το μηχανισμό γίνεται δυνατή η συνολική διαχείριση ενός αντικειμένου (π.χ. στην περίπτωσή μας μία κλάση επεξεργασίας εικόνας) με τον καθορισμό των μεταδεδομένων , τη συλλογή δεδομένων από άλλες κλάσεις και την εκτέλεσή της. Η παραπάνω διαδικασία καλείται κύκλος ζωής της κλάσης.
- Ενθάρρυνση του προγραμματισμού που είναι αφιερωμένος στο σκοπό της εργασίας μας (π.χ. βαρύτητα στην ανάπτυξη τηλεπισκοπικών διαδικασιών)
- Σύστημα βάσης δεδομένων. Ενσωματώνει ένα σύστημα βάσης δεδομένων της Java και δεν δημιουργούνται έτσι προβλήματα συμβατότητας με άλλες βάσεις δεδομένων (π.χ. SQL)
- Αποτελεί ένα εργαλείο το οποίο δομείται στη βάση πρωτοκόλλων που χρησιμοποιούνται σε διαδικτυακές εφαρμογές
- Μηχανισμός επαλήθευσης ταυτότητας και αδειοδότησης.



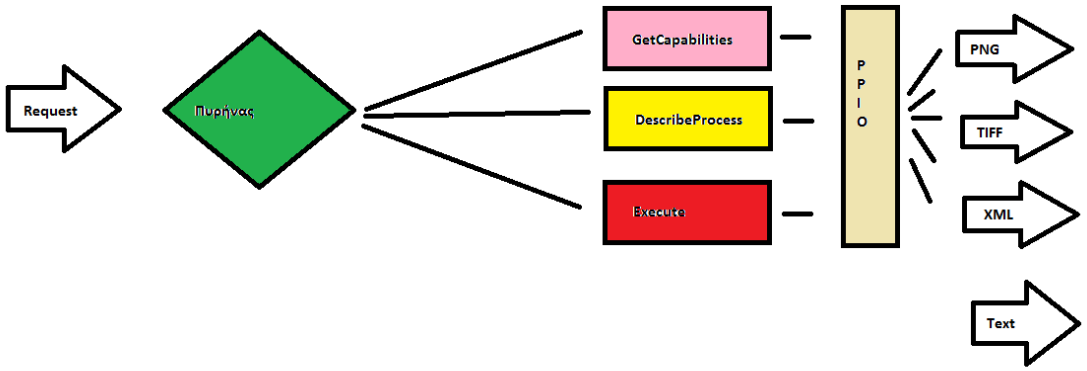
Εικόνα 2.7 : Το Spring Framework

2.2.3 GeoServer WPS

Η υπηρεσία WPS του OGC υλοποιείται στον Geoserver ,όπως και όλες οι άλλες, από το Spring Framework. Όπως έχει ήδη αναφερθεί η υπηρεσία WPS είναι η υπηρεσία εκείνη που ασχολείται με την επεξεργασία των γεωπληροφοριών. Τα δεδομένα εισόδου μπορεί να είναι σχεδόν οτιδήποτε (από

απλούς χαρακτήρες και αριθμούς έως σύνθετα XML και αντικείμενα της βιβλιοθήκης GeoTools). Επίσης τα εξαγόμενα από αυτήν μπορεί να είναι επίσης οτιδήποτε. Επομένως η υπηρεσία αυτή πρέπει να έχει ορισμένες δομές και διαδικασίες ώστε πέραν της ορθότητας του αποτελέσματος να υπάρχει και η ανάλογη απόδοση του. Η γενική αρχιτεκτονική της υπηρεσίας είναι η παρακάτω :

- Πυρήνας : Στον πυρήνα της υπηρεσίας WPS του GeoServer βρίσκονται κάποιες Java κλάσεις οποίες δρομολογούν κάθε αίτημα που δέχεται ο server ως εξής:
    - Αναγνωρίζουν τον τύπο του αιτήματος και τις παραμέτρους που αυτό το αίτημα περνάει στην υπηρεσία
    - Διαχωρίζεται η διαφορετική λειτουργία της υπηρεσίας WPS (π.χ. DescribeProcess, GetCapabilities, Execute) και δίνεται η απάντηση
    - Μετατρέπεται η απάντηση σε κατανοητή μορφή ανάλογα με την κωδικοποίηση που προβάλλεται
- Η παραπάνω «γενική δουλειά» του πυρήνα εφαρμόζεται για όλες της μεθόδους επεξεργασίας. Σε αυτό το σημείο υπεισέρχονται οι διάφορες μέθοδοι που θέλουμε να εκτελέσουμε (στην περίπτωσή μας η οι μέθοδοι επεξεργασίας εικόνας). Επίσης αυτό είναι το σημείο που είναι και επεκτάσιμο από το χρήστη. Αυτό το σημείο θα εξηγηθεί περαιτέρω στο επόμενο κεφάλαιο, καθώς αποτελεί το ουσιαστικό αντικείμενο της εργασίας.
- Τέλος περιέχεται ένα σύνολο από κλάσεις οι οποίες μετατρέπουν τα αντικείμενα της επεξεργασίας σε αντικείμενα ικανά να απεικονιστούν.



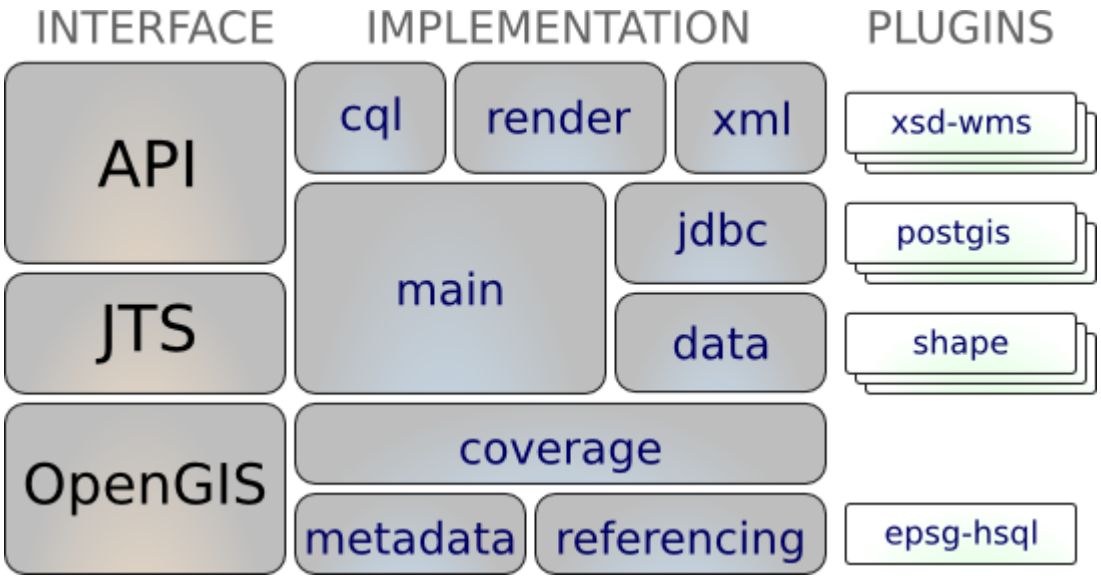
Εικόνα 2.8: Αρχιτεκτονική του Geoserver WPS

2.2.4 GeoTools

Όπως προαναφέρθηκε, η βιβλιοθήκη GeoTools είναι μία βιβλιοθήκη ανοιχτού κώδικα, γραμμένη σε Java. Σκοπός της είναι η διαχείριση γεωχωρικών δεδομένων. Για να το πετύχουν αυτό, οι προγραμματιστές της βιβλιοθήκης επέλεξαν να χρησιμοποιήσουν την προτυποποίηση του OGC.

Όπως αναφέρθηκε και πιο πριν, η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού. Αυτό σημαίνει ότι είναι φτιαγμένη για να ευνοεί την κατασκευή σύνθετων αντικειμένων της πραγματικότητας και να υποστηρίζει σύνθετες έννοιες. Η βιβλιοθήκη GeoTools βασίστηκε ακριβώς σε αυτήν την ιδιότητα της Java για να κατασκευάσει αντικείμενα(objects) τα οποία να αναφέρονται στα γεωχωρικά δεδομένα. Ένα παράδειγμα τέτοιου αντικειμένου είναι το αντικείμενο CRS (Coordinate Reference System).

Στην παρακάτω εικόνα φαίνεται η δομή της βιβλιοθήκης και οι ενότητες οι οποίες υποστηρίζονται.



Τα στοιχεία της βιβλιοθήκης που είναι απαραίτητα για την διεκπεραίωση της εργασίας είναι:

- Ορισμός διεπαφών για την αναπαράσταση χωρικών εννοιών
- Ένα σαφές σύνολο κλάσεων για την εισαγωγή και αποθήκευση δομών δεδομένων που υποστηρίζουν τη χωρική πληροφορία π.χ. του αντικειμένου CRS. Να σημειωθεί εδώ ότι υπάρχουν κλάσεις που ‘συνεργάζονται’ με το Spring Framework και έτσι γίνεται δυνατή η εισαγωγή και αποθήκευση των δεδομένων και των μεταδεδομένων(metadata) π.χ. τα ονόματα των μεταβλητών.
- Στο επίπεδο της εκτέλεσης παρέχει μία υπηρεσία δημιουργίας των αποτελεσμάτων (renderer) υψηλού επιπέδου και χαμηλών απαιτήσεων σε μνήμη.

- Τέλος παρέχει κλάσεις οι οποίες κάνουν δυνατή την αναγνώριση και την αυτόματη παραμετροποίηση των δεδομένων εισόδου και εξόδου, υποστηρίζοντας την κωδικοποίηση/αποκωδικοποίηση σχεδόν όλων των μορφών δεδομένων.

2.3 Ανάλυση μεθόδων τηλεπισκόπησης

Στο παρόν κεφάλαιο θα γίνει μία ανασκόπηση της βιβλιογραφίας της τηλεπισκόπησης και των μεθόδων που χρησιμοποιήθηκαν. Στα προηγούμενα έχει γίνει ήδη μία αναφορά στην τηλεπισκοπική εικόνα και την ψηφιακή αναπαράστασή της . Με βάση αυτήν θα εξηγηθούν οι διαδικασίες που ακολουθούνται και οι μέθοδοι που χρησιμοποιούνται.

2.3.1 Ιστόγραμμα εικόνας

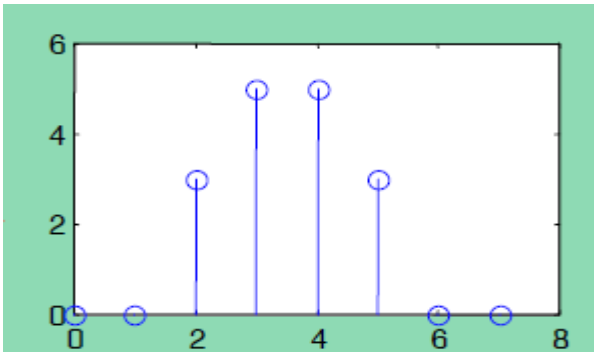
Το ιστόγραμμα εικόνας είναι μία εκτίμηση της διασποράς των διαφορετικών τιμών χρώματος σε κάθε εικόνα. Ουσιαστικά εκφράζει την συνάρτηση πυκνότητας πιθανότητας για τη μεταβλητή «τιμή χρώματος». Για να γίνουν κατανοητά τα παραπάνω, παίρνουμε το παρακάτω παράδειγμα :

Όπως είδαμε στα προηγούμενα, μία ψηφιακή εικόνα καταγράφεται στον υπολογιστή σαν πίνακας του οποίου οι τιμές ορίζουν τους διαφορετικούς τόνους του γκρι (στις έγχρωμες εικόνες έχουμε τρείς πίνακες των οποίων οι τιμές εκφράζουν διαφορετικό τόνο του κόκκινου, κίτρινου και μπλε αντίστοιχα). Έστω η παρακάτω "εικόνα»

A =

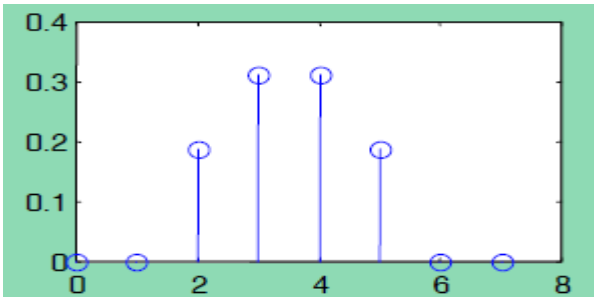
3	3	4	4
2	3	4	5
2	3	4	5
2	3	4	5

Η κατανομή των εικονοστοιχείων στις διάφορες βαθμίδες χρώματος, είναι το ιστόγραμμα της εικόνας.



Εικόνα 2.9 : Το ιστόγραμμα της παραπάνω εικόνας

Αν το ιστόγραμμα διαιρεθεί με το πλήθος των εικονοστοιχείων, παίρνουμε το κανονικοποιημένο ιστόγραμμα. Αυτό φαίνεται παρακάτω :



Εικόνα 2.10 : Κανονικοποιημένο ιστόγραμμα

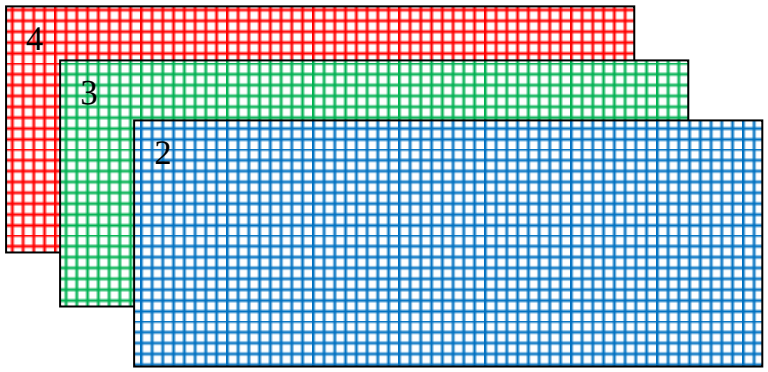
Τα αριθμητικά στοιχεία που συγκροτούν το ιστόγραμμα είναι ο μέσος όρος και η τυπική απόκλιση. Στο παραπάνω παράδειγμα ο μέσος όρος είναι η τιμή 4 και η τυπική απόκλιση είναι περίπου 2.

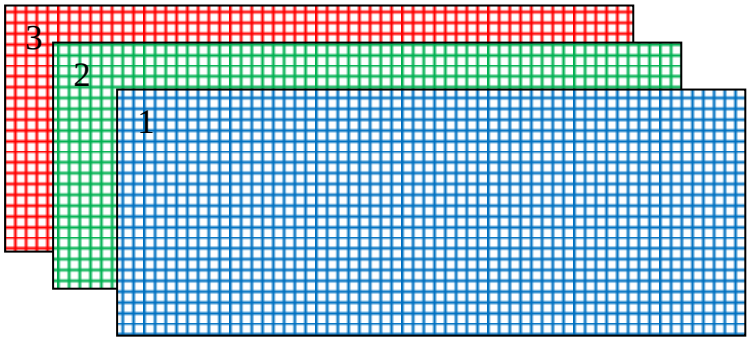
2.3.2 Έγχρωμα σύνθετα

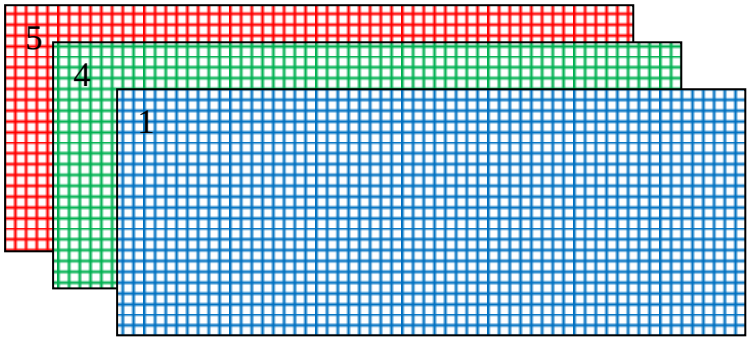
Στα προηγούμενα ορίστηκε η ψηφιακή εικόνα. Θα γίνει τώρα προσπάθεια να αναλυθούν οι συνθέσεις των επιμέρους καναλιών των ψηφιακών εικόνων. Με τον όρο κανάλι αναφερόμαστε στην ικανότητα που έχουν οι ψηφιακοί δέκτες να καταγράφουν τις διαφορετικές βαθμίδες συχνοτήτων του φωτός σε διαφορετικές εικόνες. Για παράδειγμα στο σύστημα Landsat 7 TM υπάρχουν 8 κανάλια, μαζί με το παγχρωματικό. Τα μήκη κύματος που καταγράφονται σε κάθε κανάλι φαίνονται παρακάτω :

TM & ETM+	**TM**							**ETM+**
Κανάλι	**1**	**2**	**3**	**4**	**5**	**6** **Θερμικό**	**7**	**8** **(PAN)**
μm	0,45-0,52	0,52-0,60	0,63-0,69	0,76-0,90	1,55-1,75	10,42-12,50	2,08-2,35	0,52-0,90
Ανάλυση (m)	30	30	30	30	30	120	30	15

Όπως βλέπουμε κάθε βαθμίδα καταγράφεται σε διαφορετικό κανάλι. Με αφετηρία τη θεωρητική γνώση που έχουμε για την σύνθεση των χρωμάτων μίας εικόνας μπορούμε να ανάγουμε τη δημιουργία μίας έγχρωμης εικόνας σε υπέρθεση τριών εικόνων, μίας με διαβαθμισμένους τόνους του κόκκινου, μίας με τόνους του πράσινου και μίας του μπλε. Στην περίπτωση που σε αυτές τις θέσεις τοποθετήσουμε το τρίτο το δεύτερο και το πρώτο κανάλι, έχουμε ένα έγχρωμο σύνθετο που προσιδιάζει στην πραγματική εικόνα. Στην περίπτωση που σε αυτές τις θέσεις τοποθετηθούν το τέταρτο, το τρίτο και το δεύτερο κανάλι, έχουμε ένα ψευδοέγχρωμο σύνθετο με κάποιες ιδιότητες απεικόνισης διαφορετικές από το πρώτο.







2.3.3 Κατώφλι

Η μέθοδος της εφαρμογής κατωφλιού είναι η πιο απλή μέθοδος κατάτμησης και κατηγοριοποίησης μίας εικόνας. Η μέθοδος αυτή δέχεται σαν όρισμα μία εικόνα grayscale (δηλαδή με διαβαθμισμένους τόνους του γκρι) και για ένα κατώφλι γκρίζου τόνου Κ, εφαρμόζει την παρακάτω σχέση :

Αν $\Phi(i,j) \leq K$ τότε $\Phi(i,j) = 0$

Αν $\Phi(i,j) \geq K$ τότε $\Phi(i,j) = 1$

Όπου Φ(i,j) η τιμή του εικονοστοιχείου στη θέση i,j.
Οι τιμές 0 και 1 είναι ενδεικτικές και ορίζουν δύο αντίθετες αποχρώσεις (π.χ. 0 μαύρο και 1 άσπρο ή 0 το βαθύ μπλε και 1 το ανοιχτό γαλάζιο)

2.3.4 Φίλτρα - ανίχνευση ακμών

Το φιλτράρισμα εικόνας είναι ουσιαστικά η πράξη συνέλιξης μεταξύ της αρχικής εικόνας και ενός συνόλου συντελεστών που συνήθως ονομάζονται παράθυρο ή μάσκα. Τα παράθυρα αυτά είναι συνήθως τετραγωνικά και οι συντελεστές (συνήθως) συμμετρικοί. Όταν η εικόνα είναι γεωαναφερμένη, ορίζουμε το χωρικό φιλτράρισμα.

Το χωρικό φιλτράρισμα είναι μια συνηθισμένη λειτουργία που εφαρμόζεται στα δεδομένα ψηφιακής εικόνας για να ενισχύσει ή να υποβαθμίσει τη χωρική λεπτομέρεια και για να βελτιωθεί η οπτική ερμηνεία. Τυπικά παραδείγματα είναι η εφαρμογή φίλτρων για να ενισχυθεί η λεπτομέρεια των ακμών στις εικόνες, ή για να αφαιρεθούν ή να μειωθούν μορφές θορύβου σε μία εικόνα. Στην επεξεργασία εικόνας, το χωρικό φιλτράρισμα καλείται "τοπική λειτουργία", επειδή τροποποιεί την τιμή κάθε εικονοστοιχείου στην εικόνα ανάλογα με τις τιμές των εικονοστοιχείων που το περιβάλλουν. Τα φίλτρα δουλεύουν με την αφαίρεση ορισμένων φασματικών ή χωρικών συχνοτήτων για να βελτιωθούν τα χαρακτηριστικά στην υπόλοιπη εικόνα. Τα χωρικά φίλτρα μπορούν να χωριστούν σε τρεις γενικές κατηγορίες:

- Τα **φίλτρα χαμηλής συχνότητας (χαμηλοδιαβατά φίλτρα)** σχεδιάζονται για να τονίσουν τα χαρακτηριστικά χαμηλών συχνοτήτων (μεταβολές φωτεινότητας μεγάλων περιοχών) και να αποδυναμώσουν τις υψηλές συχνότητες μίας εικόνας (τοπικές λεπτομέρειες). Δεδομένου ότι υποβαθμίζουν τη λεπτομέρεια σε μία εικόνα, τα φίλτρα χαμηλών συχνοτήτων μερικές φορές καλούνται φίλτρα εξομάλυνσης ή φίλτρα μέσης τιμής. Τα φίλτρα αυτά τονίζουν την λεπτομέρεια χαμηλής συχνότητας για να εξομαλύνουν το θόρυβο εικόνας ή να μειώσουν τα ακραία δεδομένα.

- Τα **φίλτρα υψηλών συχνοτήτων (υψιπερατά φίλτρα**) κάνουν ακριβώς το αντίθετο, δηλαδή, τονίζουν τις χωρικές λεπτομέρειες που σχετίζονται με τις υψηλές συχνότητες μίας εικόνας, και αποδυναμώνουν τις πλέον γενικές πληροφορίες, οι οποίες συνδέονται με τις χαμηλές συχνότητες. Τα φίλτρα υψηλών συχνοτήτων καλούνται μερικές φορές φίλτρα όξυνσης επειδή γενικά χρησιμοποιούνται για να ενισχύσουν τη λεπτομέρεια χωρίς να επηρεάζονται τα τμήματα χαμηλής συχνότητας της εικόνας. Τα φίλτρα αυτά τονίζουν τη λεπτομέρεια υψηλής συχνότητας για να ενισχύσουν ή να οξύνουν γραμμικά χαρακτηριστικά όπως **δρόμοι, ρήγματα και όρια εδάφους / νερού.**

- Τα **φίλτρα ανίχνευσης ακμών (edge detection filters)** τονίζουν τις ακμές που περιβάλλουν αντικείμενα ή χαρακτηριστικά σε μία εικόνα για να καταστήσει ευκολότερη την ανάλυση τους. Τα φίλτρα ανίχνευσης ακμών δημιουργούν συνήθως μία εικόνα με γκρίζο φόντο και μαύρες και άσπρες γραμμές που περικλείουν τις ακμές των αντικειμένων και των χαρακτηριστικών στην εικόνα. Η μαθηματική αρχή που εφαρμόζεται κατά την εφαρμογή των φίλτρων ενίσχυσης ακμών είναι ότι μία ακμή έχει θετική εφαπτομένη (κατά μέτρο) κατά τη διεύθυνση της κλίσης της , ενώ έχει μηδενική εφαπτομένη κατά τις άλλες διευθύνσεις. Η αρχή αυτή φαίνεται στις παρακάτω εξισώσεις :

$$\frac{\partial f}{\partial x} = f(i, j) - f(i+1, j+1)$$

$$\frac{\partial f}{\partial y} = f(i+1, j) - f(i, j+1)$$

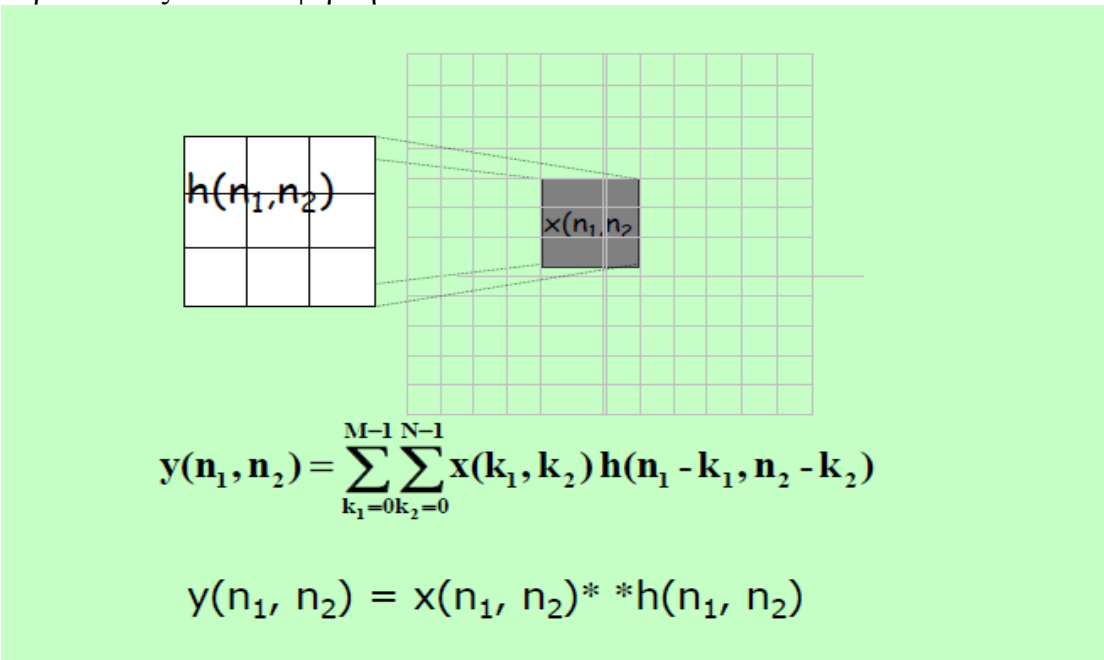
Για μία ψηφιακή εικόνα η παραπάνω εξίσωση εκφράζεται από ένα πολυώνυμο το οποίο έχει σαν συντελεστές τα στοιχεία των μασκών συνέλιξης κατά x και κατά y. Οι συντελεστές του πολυωνύμου αυτού έχουν τις παρακάτω ιδιότητες :

$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

Για την κατασκευή των μασκών απαιτούμε M_x και M_y να είναι 0. Επίσης , ο παράγοντας c εκφράζει την έμφαση που δίνεται στα στοιχεία που είναι γειτονικά του pixel. Η μάσκα κάθε φορά είναι :

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

Στην επόμενη εικόνα φαίνεται η αρχή υπολογισμού της συνέλιξης για τις περιπτώσεις που αναφέρθηκαν :

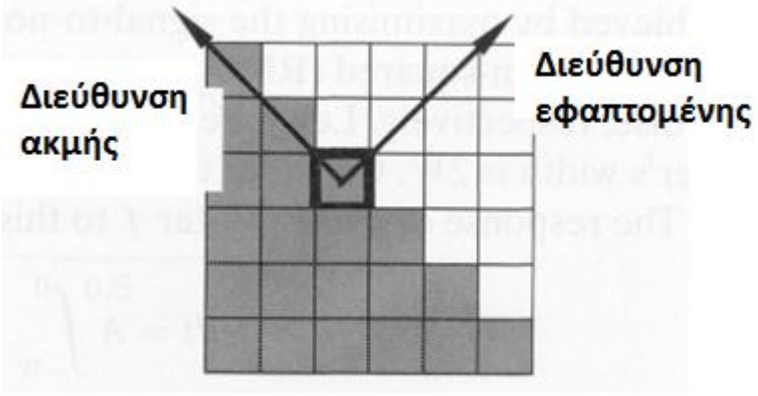


Εικόνα 2.11 : Συνέλιξη εικόνας

- Το **φίλτρο πρώτης παραγώγου** παρέχει πληροφορίες για την ένταση των ακμών ώστε να είναι ευκολότερη η αντίληψη των μορφών των σχημάτων. Μαθηματικά ορίζεται ως ο υπολογισμός για κάθε pixel του μέτρου του διανύσματος της κλίσης σε αυτό το pixel. Η κλίση αντιστοιχεί στις ακμές που ανιχνεύτηκαν κατά την προηγούμενη διαδικασία συνέλιξης. Έτσι, η ακολουθία των διαδικασιών είναι η παρακάτω :

    - Εφαρμόζεται ένα φίλτρο ανίχνευσης ακμών στην εικόνα, κατά την οριζόντια και κατακόρυφη διεύθυνση.
    - Για κάθε pixel επί των ακμών ορίζεται η εφαπτομένη η οποία έχει διεύθυνση κάθετη στη διεύθυνση της ακμής
    - Υπολογίζεται το μέτρο του διανύσματος από τον τύπο : $\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$
      Όπου G_x και G_y είναι οι μάσκες των φίλτρων κατά την οριζόντια και κάθετη διεύθυνση αντίστοιχα. Αυτή η τιμή αποθηκεύεται στο συγκεκριμένο pixel

Συμπεραίνουμε από τα προηγούμενα ότι το φίλτρο πρώτης παραγώγου είναι ισοτροπικό, δηλαδή ανιχνεύει ακμές σε όλες τις διευθύνσεις, καθώς η τελική τιμή του Pixel είναι το μέτρο της κλίσης. Παρακάτω φαίνεται η αρχή λειτουργίας του φίλτρου αυτού :



Εικόνα 2.12 : Εφαπτομένη ακμής

Το μέτρο και η διεύθυνση του διανύσματος της εφαπτομένης :

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$magn(\nabla f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{M_x{}^2 + M_y{}^2}$$

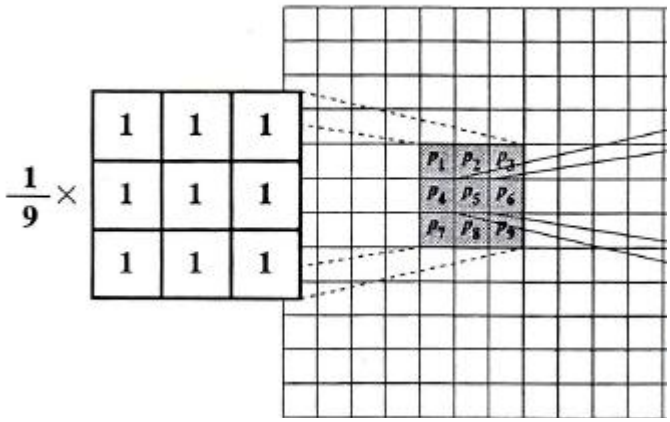
$$dir(\nabla f) = \tan^{-1}(M_y / M_x)$$

Αναφέρονται παρακάτω περιπτώσεις χωρικών φίλτρων που χρησιμοποιήθηκαν στα πλαίσια της παρούσας εργασίας :

- Φίλτρο εξομάλυνσης

Το φίλτρο εξομάλυνσης είναι ένα χαμηλοδιαβατό φίλτρο του οποίου η κύρια χρησιμότητα είναι η απαλοιφή του θορύβου, δηλαδή συστηματικών

σφαλμάτων που έχουν δημιουργηθεί στην εικόνα κατά τη διαδικασία αποθήκευσης. Η μάσκα που χρησιμοποιείται παίρνει τις τιμές της ανάλογα με το μέγεθος που έχει. Συγκεκριμένα, είναι μία μάσκα της οποίας τα στοιχεία είναι τα αντίστροφα του αριθμού των εικονοστοιχείων της. Παρακάτω βλέπουμε την περίπτωση της μάσκας 3X3 :



Εικόνα 2.13 : Μάσκα εξομάλυνσης 3x3

- Φίλτρο Sobel

Το φίλτρο Sobel είναι ένα φίλτρο ανίχνευσης ακμών. Προκύπτει για τιμή c=2. Η συνέλιξη της αρχικής εικόνας με τη μάσκα Sobel έχει σαν αποτέλεσμα μία εικόνα που εμφανίζει ενισχυμένες τις ακμές και τα όρια. Το φίλτρο Sobel έχει δύο μάσκες, μία για εφαρμογή κατά την οριζόντια διεύθυνση και μία κατά την κατακόρυφη. Οι δύο μάσκες είναι οι παρακάτω :

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Τέλος, μετά την εφαρμογή στην οριζόντια και την κατακόρυφη διεύθυνση, εφαρμόζεται το φίλτρο της πρώτης παραγώγου :

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

Επομένως, η διαδικασία του φιλτραρίσματος με το φίλτρο Sobel μπορεί να έχει τρία αποτελέσματα : Φίλτρο μόνο κατά την οριζόντια διεύθυνση, φίλτρο μόνο κατά την κατακόρυφη διεύθυνση, υπολογισμός της φίλτρου πρώτης παραγώγου. Να σημειωθεί εδώ ότι μπορεί να χρησιμοποιηθούν μεγαλύτερες μάσκες για το φίλτρο Sobel (5x5 , 7x7).

- Φίλτρο Prewitt

Το φίλτρο Prewitt ανήκει και αυτό στα φίλτρα ανίχνευσης ακμών. Προκύπτει για τιμή c=1. Έχει παρόμοια λογική με το φίλτρο Sobel αλλά χρησιμοποιεί διαφορετικές μάσκες για τον υπολογισμό της τελικής τιμή. Αυτές είναι :

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Επίσης μπορεί και εδώ να υπολογιστεί η πρώτη παράγωγος όπως ορίστηκε στα προηγούμενα :

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

- Φίλτρο Roberts

Το Φίλτρο Roberts είναι ένα φίλτρο ανίχνευσης ακμών. Έχει την ίδια λογική με τα φίλτρα Sobel και Prewitt, με μόνη διαφορά τη χρήση διαφορετικών μασκών. Αυτές είναι :

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}.$$

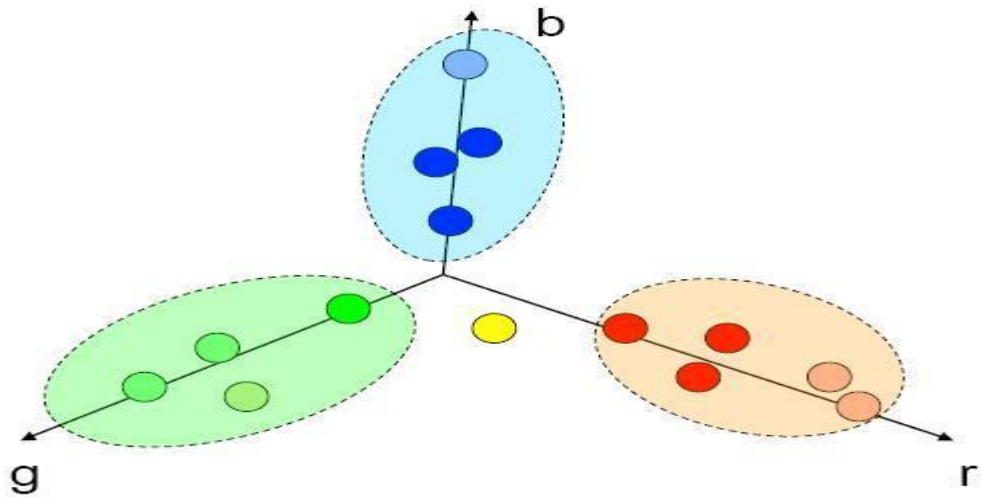
Επίσης μπορεί και εδώ να υπολογιστεί η πρώτη παράγωγος όπως ορίστηκε στα προηγούμενα :

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

2.3.5 Μη επιβλεπόμενη ταξινόμηση – αλγόριθμος K – means

Η μη επιβλεπόμενη ταξινόμηση δεν χρησιμοποιεί δεδομένα εκπαίδευσης. Αξιοποιεί αλγορίθμους, οι οποίοι εξετάζουν τα άγνωστα εικονοστοιχεία μίας εικόνας και τα ομαδοποιούν σε έναν αριθμό κατηγοριών με βάση τις φυσικές ομαδοποιήσεις ή συσσωρεύσεις που ενυπάρχουν στις ψηφιακές τιμές της εικόνας. Η βασική αρχή είναι ότι, οι τιμές που προέρχονται από μία συγκεκριμένη κατηγορία κάλυψης γης, βρίσκονται κοντά η μία στην άλλη στο χώρο προτύπων, ενώ τιμές από διαφορετικές κατηγορίες χρήσεων γης θα πρέπει συγκριτικώς να είναι καλά διαχωρίσιμες.

Ο αλγόριθμος K – means είναι μία ειδική περίπτωση μη επιβλεπόμενης ταξινόμησης. Η βασική ιδέα που υπάρχει πίσω από τον αλγόριθμο αυτό είναι ο χωρισμός μίας εικόνας σε K διακριτές κλάσεις και η ένταξη κάθε pixel σε κάποια από αυτές. Η ένταξη αυτή, πραγματοποιείται με το κριτήριο της ελάχιστης απόστασης στο επίπεδο των τιμών χρώματος. Παρακάτω φαίνεται ο τρόπος αυτού του διαχωρισμού σε μία έγχρωμη εικόνα :



Εικόνα 2.14 : Ελάχιστη απόσταση της τιμής χρώματος

Ο αλγόριθμος δέχεται σαν ορίσματα την αρχική εικόνα, τον αριθμό των κλάσεων, τον παράγοντα σύγκλισης και τον αριθμό μέγιστων επαναλήψεων και δίνει σαν έξοδο την κατατετμημένη εικόνα.

Η υλοποίηση αυτού του αλγορίθμου ακολουθεί συγκεκριμένα βήματα :

- Αρχικοποίηση των κέντρων των κλάσεων : Αυτό γίνεται συνήθως με τυχαία επιλογή
- Κατάταξη όλων των pixel σε κάποια κλάση με βάση την απόσταση τους από αυτή
- Εκ νέου υπολογισμός των κέντρων των κλάσεων. Αυτά υπολογίζονται ως ο μέσος όρος κάθε κλάσης.
- Επανάληψη των βημάτων 2 και 3 μέχρι να ικανοποιηθεί ο παράγοντας σύγκλισης και μέχρι να πραγματοποιηθεί ο αριθμός μέγιστων επαναλήψεων.

Πρέπει να σημειωθεί εδώ ότι διάφορες εκδόσεις αυτού του αλγορίθμου, αλλάζουν τη μάσκα χρώματος, χρωματίζουν δηλαδή τα pixel κάθε κλάσης με χρώματα έντονης αντίθεσης, έτσι ώστε να είναι ευδιάκριτες οι διαφορετικές κλάσεις. Στην υλοποίηση που έγινε στην παρούσα εργασία, χρησιμοποιήθηκε το φυσικό χρώμα κάθε κλάσης.

2.3.6 Pansharpening

Η τεχνική Pansharpening έχει σκοπό τη βελτίωση της χωρικής ανάλυσης της τηλεπισκοπικής εικόνας με ταυτόχρονη βελτίωση των χρωματικών της χαρακτηριστικών. Όπως είδαμε παραπάνω ο δέκτης Landsat έχει 6 κανάλια ορατής και υπέρυθρης ακτινοβολίας(μέγεθος εδαφοψηφίδας 30 m), ένα θερμικό με μικρότερη ανάλυση (μέγεθος εδαφοψηφίδας 120 m) και ένα παγχρωματικό κανάλι (μέγεθος εδαφοψηφίδας 15 m). Το τελευταίο είναι ένα κανάλι το οποίο καταγράφει όλο το φάσμα της ορατής ακτινοβολίας σε μία Grayscale εικόνα (δηλαδή με διαβαθμισμένους τόνους του γκρι). Αυτό ισχύει για τα περισσότερα συστήματα δορυφορικών απεικονίσεων (π.χ. Spot, Geoeye, DigitalGlobe) . Στόχος της τεχνικής Pansharpening είναι να εκμεταλλευτεί τα καλύτερα δεδομένα από δυο πλευρές. Αφ' ενός να εκμεταλλευτεί την καλή ανάλυση της παγχρωματικής εικόνας και αφ' ετέρου την σωστή καταγραφή των χρωμάτων από τα επιμέρους κανάλια.

Ο αλγόριθμος Pansharpening παίρνει σαν δεδομένα τέσσερις εικόνες (κανάλια 3,2,1 και παγχρωματικό) και δίνει σαν αποτέλεσμα μία εικόνα που μοιάζει με το έγχρωμο σύνθετο 3-2-1, αλλά έχει καλύτερη ανάλυση και πιστότερα χρώματα. Τα βήματα που ακολουθεί ο αλγόριθμος Pansharpening είναι τα παρακάτω :

- Δημιουργία έγχρωμης εικόνας από τα κανάλια 3-2-1 με συνένωση των επιμέρους καναλιών (band merge)
- Υποδιπλασιασμός του μεγέθους του pixel της (καθώς θα πάρουμε δεδομένα γι' αυτό από την παγχρωματική εικόνα διπλάσιας χωρικής ανάλυσης).
- Παρεμβολή στην παγχρωματική εικόνα και υπολογισμός της τιμής χρώματος.
- Ανάλυση της τιμή αυτής σε τρείς συνιστώσες μέσω διάφορων διαδικασιών (Στη συγκεκριμένη υλοποίηση έγινε μέσω μετατροπής του χρώματος από το σύστημα απόχρωσης και κορεσμού στο σύστημα RGB) και τελικός υπολογισμός της τιμής χρώματος
- Επανάληψη της προηγούμενης διαδικασίας για κάθε pixel.

3. Υλοποίηση μεθόδων

Σε αυτήν την ενότητα, θα αναλυθεί η υλοποίηση της εφαρμογής και η διαδικασίες που ακολουθήθηκαν προκειμένου να επιτευχθεί. Στο πρώτο κεφάλαιο εξετάζεται το λογισμικό που χρησιμοποιήθηκε και αναλύεται όλη η διαδικασία εγκατάστασης και χρήσης του. Στο δεύτερο κεφάλαιο εξετάζεται το προγραμματιστικό κομμάτι στη γλώσσα Java που αναπτύχθηκε με σκοπό την υλοποίηση των τηλεπισκοπικών διαδικασιών. Το τελευταίο κεφάλαιο αυτής της ενότητας εξηγεί τη διαδικασία ενσωμάτωσης αλγορίθμων γραμμένων σε Java στην υπηρεσία WPS.

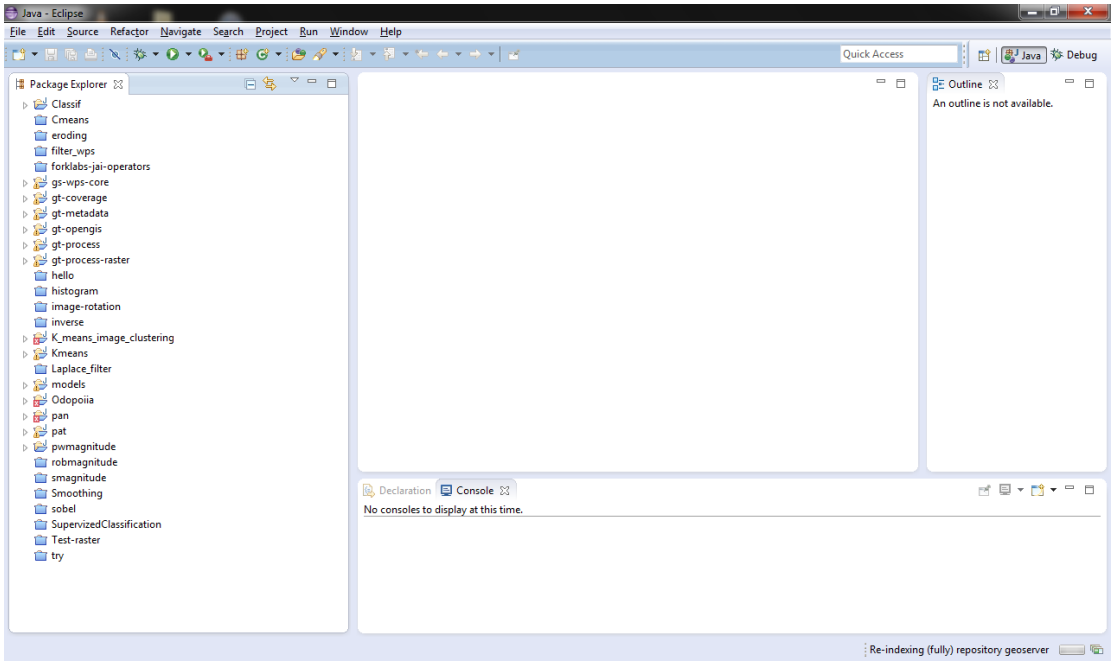
3.1 Εγκατάσταση λογισμικού

Στο κεφάλαιο αυτό θα περιγραφούν όλα τα λογισμικά, οι βιβλιοθήκες, τα περιβάλλοντα ανάπτυξης και τα δίκτυα που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Η διαδικασία περιγράφεται για την εγκατάσταση του λογισμικού σε τοπικό υπολογιστή.

Η ανάπτυξη υλοποιήθηκε σε λειτουργικό σύστημα Windows 7. Αυτό όμως δεν επηρεάζει τη διαδικασία που περιγράφεται παρακάτω, καθώς όπως είδαμε η γλώσσα Java είναι ανεξάρτητη από λειτουργικά συστήματα και επεξεργαστές και εκτελείται κάθε φορά από την ψηφιακή μηχανή (virtual machine). Επίσης αναπτύσσεται συνήθως σε ολοκληρωμένα περιβάλλοντα ανάπτυξης (όπως το eclipse ή το NetBeans) και εγκιβωτίζεται από εργαλεία της γλώσσας Java (όπως το maven ή το Ant). Επομένως η περιγραφή που ακολουθεί είναι ανεξάρτητη από λειτουργικά συστήματα.

3.1.1 Eclipse

Το πρώτο βήμα είναι η εγκατάσταση του ολοκληρωμένου περιβάλλοντος ανάπτυξης. Για την παρούσα εργασίας επιλέχθηκε το eclipse kepler v.10. Το συγκεκριμένο περιβάλλον έχει έναν επεξεργαστή κειμένου για την ανάπτυξη κώδικα και ενσωματώνει τον μεταγλωττιστή της Java. Επίσης έχει πληθώρα βοηθητικών εφαρμογών, προκειμένου ο χρήστης να διευκολυνθεί στην ανάπτυξη. Τέτοια χαρακτηριστικά είναι η δυνατότητα άμεσης σύνδεσης με άλλες κλάσεις (στον τοπικό υπολογιστή ή στο διαδίκτυο), η αυτόματη διόρθωση συντακτικών λαθών, η ευκολία διενέργειας ελέγχου, η αποσφαλμάτωση , η δυνατότητα σύνδεσης στο διαδίκτυο (χωρίς χρήση browser) και πολλά άλλα. Επίσης όλα τα παραπάνω υποστηρίζονται από ένα γραφικό περιβάλλον φιλικό προς το χρήστη. Παρακάτω φαίνεται αυτό το περιβάλλον.



Εικόνα 3.1 : Το γραφικό περιβάλλον του eclipse

3.1.2 Apache Maven

Το αμέσως επόμενο βήμα είναι η εγκατάσταση του προγράμματος εγκιβωτισμού. Αυτό επιλέχθηκε να είναι το Apache Maven. Το maven είναι ένα λογισμικό το οποίο έχει σαν στόχο την δημιουργία ενιαίων πακέτων Java σε format jar συνήθως. Όπως αναφέρθηκε στα προηγούμενα, η Java είναι μία γλώσσα η οποία ευνοεί την δημιουργία συναρτήσεων μεθόδων και πεδίων τα οποία να μπορούν να χρησιμοποιηθούν και από άλλα τμήματα κώδικα. Στην πράξη κατά την ανάπτυξη κώδικα Java σχεδόν πάντα υπάρχει η ανάγκη χρήσης τέτοιων μεθόδων και συναρτήσεων. Η δουλειά του maven είναι να ενσωματώσει όλα τα διαφορετικά τμήματα κώδικα σε ένα ενιαίο πακέτο σε κωδικοποίηση jar. Το maven δεν έχει γραφικό περιβάλλον επεξεργασίας. Ελέγχεται από το τερματικό. Για να ορίσουμε κάποιες παραμέτρους στο maven θα πρέπει να του δώσουμε συγκεκριμένες εντολές από το τερματικό. Τέτοιες εντολές είναι :

```
mvn --version
```

και η απάντηση που εμφανίζει το maven στο command line είναι :

```
Apache Maven 3.0.5
(r01de14724cdef164cd33c7c8c2fe155faf9602da;
2013-02-19 14:51:28+0100)
Maven home: D:\apache-maven-3.0.5\bin\..
Java version: 1.6.0_25, vendor: Sun Microsystems
Inc.
Java home: C:\Program Files\Java\jdk1.6.0_25\jre
Default locale: nl_NL, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch:
"amd64", family: "windows"
```

Δηλαδή επεξηγεί την έκδοση που είναι εγκατεστημένη και την έκδοση της Java που είναι εγκατεστημένη.

Το maven είναι ένα εργαλείο που δημιουργεί Java projects. Το Java project είναι ένα καθορισμένο σύστημα φακέλων και υπό-φακέλων οι οποίοι είναι απαραίτητοι για την ανάπτυξη κώδικα. Η δημιουργία γίνεται με την παρακάτω εντολή :

```
mvn archetype:generate -
DgroupId=com.mycompany.app -DartifactId=my-app
-DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

Και η δομή φακέλων που δημιουργείται είναι η παρακάτω :

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Στο φάκελο src\main\java βρίσκεται ο κώδικας Java, ενώ στο φάκελο src\test\java βρίσκονται οι κλάσεις ελέγχου.

Με τη δημιουργία του συστήματος αυτού δημιουργείται και το pom το οποίο είναι ένα αρχείο xml το οποίο πληροφορεί το maven που βρίσκονται οι κλάσεις που χρησιμοποιούμε στον κώδικά μας. Το pom έχει συνήθως την παρακάτω δομή :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.
0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Τέλος με το τερματικό στο φάκελο src και την πληκτρολόγηση της παρακάτω εντολής :

```
mvn clean install
```

Δημιουργείται ένα νέος φάκελος στο παραπάνω σύστημα φακέλων το οποίο περιέχει το αρχείο jar. Αυτό το αρχείο είναι εκτελέσιμο από οποιαδήποτε ψηφιακή μηχανή java.

3.1.3 Εγκατάσταση Geoserver

Τέλος μετά την εγκατάσταση των δύο προαναφερθέντων εργαλείων εγκαθίσταται ο Geoserver. Λόγω της χρήση λειτουργικού συστήματος windows εγκαταστάθηκε χάριν ευκολίας με τον windows installer η έκδοση 2.4.0 και λόγω κάποιων προβλημάτων στην έξοδο δεδομένων από την υπηρεσία WPS προστέθηκε το patch main 2.4-Snapshot. Επίσης εγκαταστάθηκε ο υπηρεσία WPS με την εγκατάσταση των patch από τη σελίδα : http://sourceforge.net/

Ο Geoserver τώρα ανταποκρίνεται στη διεύθυνση http://localhost:8080/geoserver/web/

3.1.4 Εγκατάσταση GeoTools

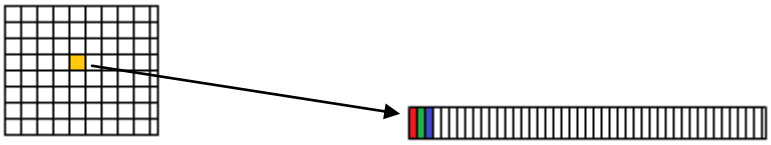
Η βιβλιοθήκη GeoTools αν και δεν είναι άμεσα απαραίτητη στη ανάπτυξη των εφαρμογών, περιέχει εντούτοις μεγάλο μέρος των κλάσεων που χρησιμοποιούνται και είναι πολύ χρήσιμη στη διενέργεια ελέγχων. Μεταφορτώνεται η έκδοση που προτιμάται από τη διεύθυνση : http://sourceforge.net/ . Στην παρούσα εργασία χρησιμοποιήθηκε η έκδοση 10.0.

3.2 Ανάπτυξη εφαρμογών στη γλώσσα Java

Στο παρόν κεφάλαιο εξετάζεται η ανάπτυξη εφαρμογών τηλεπισκόπισης στη γλώσσα Java. Ο λόγος που επιλέχθηκε αρχικά η Java είναι η δυνατότητα που έχει ο Geoserver για επέκταση, η οποία πρέπει να είναι σε γλώσσα Java. Διαπιστώθηκε όμως ότι υπάρχουν εξαιρετικές δυνατότητες της γλώσσας στο επίπεδο της επεξεργασίας εικόνας και ήδη έτοιμες βελτιστοποιημένες βιβλιοθήκες, οι οποίες μπορούν να χρησιμοποιηθούν για την εφαρμογή που μας ενδιαφέρει.

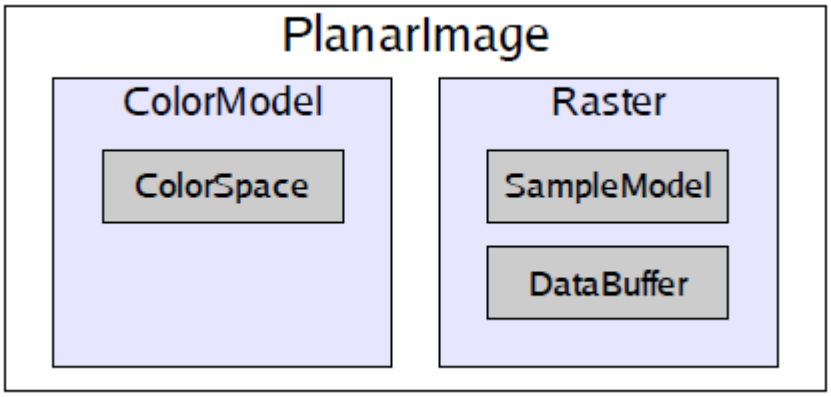
3.2.1 Εικόνες στη Java και τη βιβλιοθήκη JAI

Στην παρούσα ενότητα θα γίνει προσπάθεια να αναλυθεί η έννοια της εικόνας στη Java και τη βιβλιοθήκη JAI. Όπως αναφέρθηκε στα προηγούμενα κεφάλαια, η Java είναι μία γλώσσα προγραμματισμού που εκτελείται ανεξάρτητα από πλατφόρμες και συσκευές. Πρέπει επομένως να είναι σε θέση να μπορεί να περιγράψει και να διαχειριστεί όλες τις πιθανές περιπτώσεις εικόνων. Για το λόγο αυτό η περιγραφή μίας εικόνας είναι σύνθετη διαδικασία και περιέχει πολλές παραμέτρους. Μία εικόνα στη βιβλιοθήκη JAI περιγράφεται κυρίως από την κλάση PlanarImage. Αυτή η κλάση περιέχει δύο άλλες βασικές κλάσεις, οι οποίες περιγράφουν δύο συστατικά στοιχεία της εικόνας. Αυτές οι κλάσεις είναι οι ColorModel και Raster. Η κλάση ColorModel περιγράφει το μοντέλο με το οποίο οπτικοποιούνται οι αριθμητικές τιμές ενός πίνακα. Αντίστοιχα η κλάση Raster περιγράφει το πώς αποθηκεύονται οι αριθμοί στον πίνακα. Η κλάση Raster απαιτεί για τον ορισμό της δύο άλλες κλάσεις : την κλάση SampleModel και την κλάση DataBuffer. Η κλάση SampleModel περιέχει στοιχεία για τον τρόπο αναπαράστασης της πληροφορίας μέσω των στοιχείων του πίνακα. Για παράδειγμα μπορεί μία εικόνα να αποθηκεύεται σε έναν πίνακα μίας διάστασης, αλλά το SampleModel της να περιέχει την πληροφορία ότι σε κάθε pixel αντιστοιχούν τρείς τιμές οι οποίες αντιστοιχούν σε τρία γειτονικά στοιχεία του πίνακα. Το παρακάτω σχήμα εξηγεί την προηγούμενη πρόταση :



Επίσης, η κλάση DataBuffer περιέχει το είδος των στοιχείων π.χ. ακέραιοι αριθμοί, στοιχεία 8-bit, προσημασμένοι ακέραιοι, ρητοί ακρίβειας double, ρητοί ακρίβειας float και άλλα. Επίσης περιέχει πληροφορίες για τον αριθμό των πινάκων που χρησιμοποιούνται π.χ. τρείς στην πρώτη περίπτωση ή ένας στη δεύτερη.

Η εικόνα που ακολουθεί συνοψίζει όλα τα παραπάνω :



Εικόνα 3.2 : Η κλάση PlanarImage

3.2.2 Κόμβοι επεξεργασίας εικόνας στη βιβλιοθήκη JAI

Η βιβλιοθήκη JAI (Java Advanced Imaging) κατοχυρώνει σαν καινοτομία την ύπαρξη της αλυσιδωτής επεξεργασίας εικόνας κατά κόμβους. Η βασική ιδέα που υλοποιεί η συγκεκριμένη διαδικασία είναι η επεξεργασία εικόνας κατά βήματα. Κάθε βήμα ορίζει και μία διαδικασία επεξεργασίας εικόνας. Κάθε βήμα της διαδικασίας είναι αυστηρά καθορισμένο και εκτελέσιμο. Υλοποιείται όμως μόνο στην περίπτωση που ζητηθεί (deferred execution – αναβαλλόμενη εκτέλεση).

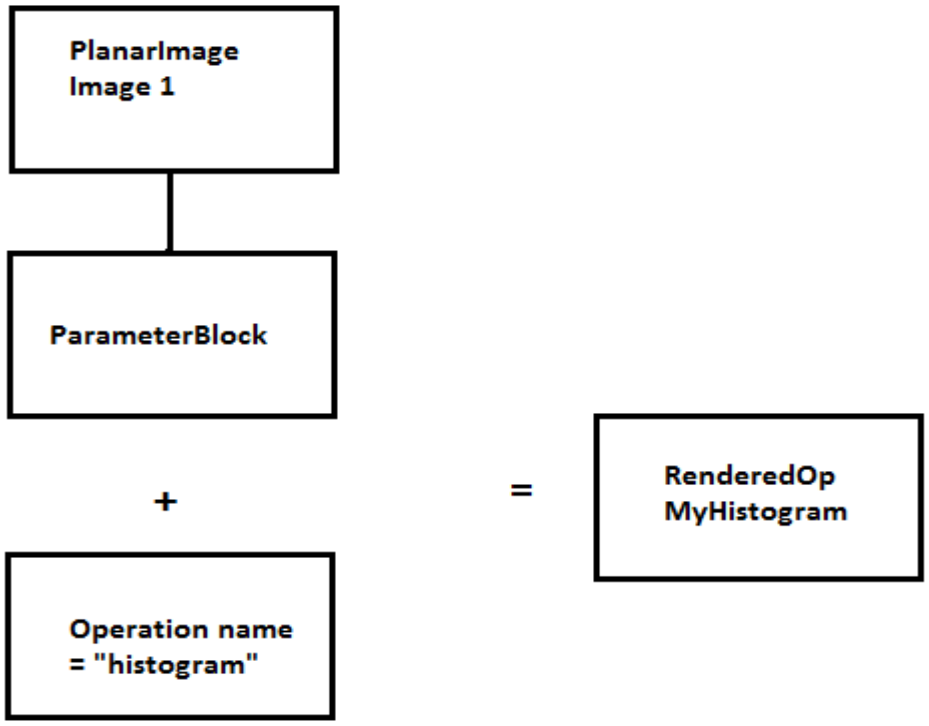
Μία κλάση που ονομάζεται OperationDescriptorImpl περιγράφει όλες αυτές τις διαδικασίες. Κάθε διαδικασία ορίζεται με τον εξής τρόπο :

- Ορίζεται πρώτα ένα διάνυσμα το οποίο έχει σαν στοιχεία του τις τιμές των παραμέτρων που απαιτεί κάθε διαδικασία για να οριστεί (π.χ. εικόνα και τιμή για την οποία θα γίνει κατωφλίωση μίας εικόνας). Το διάνυσμα αυτό αντιπροσωπεύεται από μια κλάση στη Java. Η κλάση αυτή ονομάζεται ParameterBlock

- Τα στοιχεία του ParameterBlock δίνονται σαν ορίσματα στην κλάση JAI, μαζί με το όνομα κάθε διαδικασίας του OperationDescriptorImpl. Η κλάση αυτή είναι η υπεύθυνη κλάση για την «πυροδότηση» ή μη της κομβικής επεξεργασίας που περιγράφεται παραπάνω
- Η κλάση OperationDescriptorImpl επεξεργάζεται κατά τον τρόπο που ορίσαμε την εικόνα και επιστρέφει το αποτέλεσμα.
- Το αποτέλεσμα είναι της μορφής RenderedOp και εκφράζει κάθε φορά την ενεργοποίηση της διαδικασίας της αλυσιδωτής επεξεργασίας κατά κόμβους

3.2.3 Ιστόγραμμα

Η εμφάνιση αναλυτικών στοιχείων ιστογράμματος είναι μία διαδικασία που περιέχεται στη βιβλιοθήκη JAI. Δέχεται σαν όρισμα την εικόνα και μέσω της διαδικασίας που περιγράφεται στα προηγούμενα ενεργοποιεί τον κόμβο του ιστογράμματος. Το αποτέλεσμα είναι μία ακολουθία χαρακτήρων που περιγράφει το μέσο όρο, την τυπική απόκλιση και την εντροπία (για κάθε κανάλι της εικόνας). Παρακάτω φαίνεται η διαδικασία που ακολουθείται :

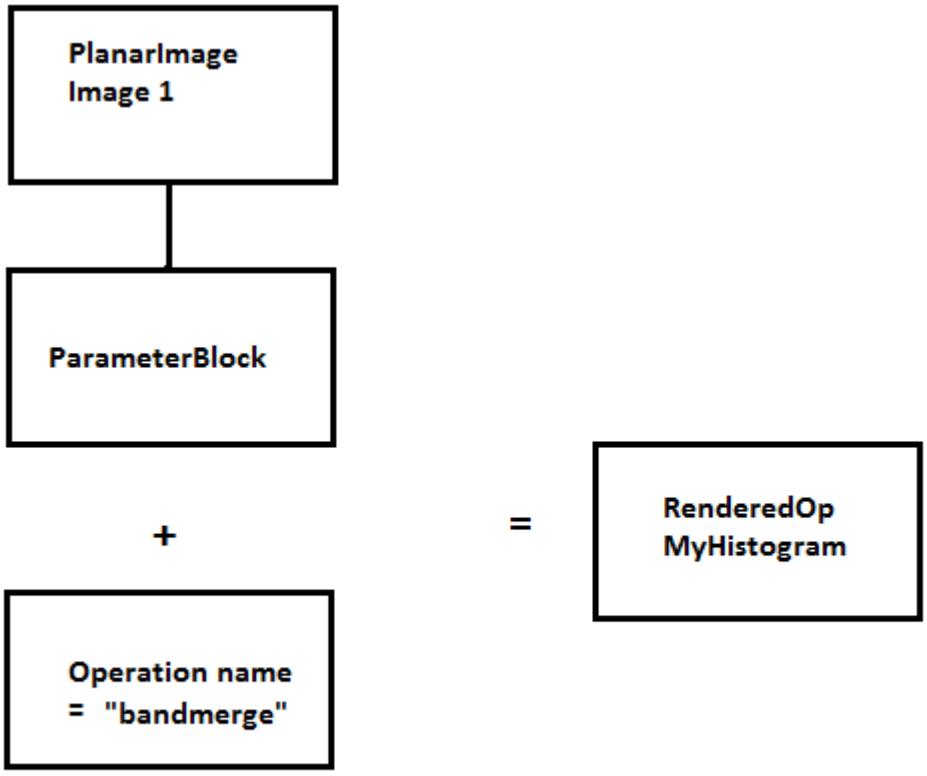


Εικόνα 3.3 : Κόμβος δημιουργίας ιστογράμματος

Ο κώδικας για τη συγκεκριμένη διαδικασία βρίσκεται στο παράρτημα.

3.2.4 Έγχρωμα σύνθετα

Η δημιουργία έγχρωμων σύνθετων είναι μία διαδικασία που περιέχεται στην βιβλιοθήκη JAI. Αποτελεί και αυτή μία περίπτωση κόμβου της αλυσιδωτής επεξεργασίας εικόνας. Η διαδικασία δέχεται σαν ορίσματα τρείς εικόνες τύπου Rendered (ή Renderable) Image οι οποίες αντιστοιχούν στις εικόνες που θα υπερτεθούν ώστε να σχηματισθεί το έγχρωμο σύνθετο. Η διαδικασία δίνει σαν αποτέλεσμα μία εικόνα η οποία αποτελεί σύνθεση των τριών εικόνων εισόδου, όπως περιγράφηκε στα προηγούμενα.



Εικόνα 3.4 : Κόμβος δημιουργίας έγχρωμου σύνθετου

3.2.5 Φίλτρα – ανίχνευση ακμών

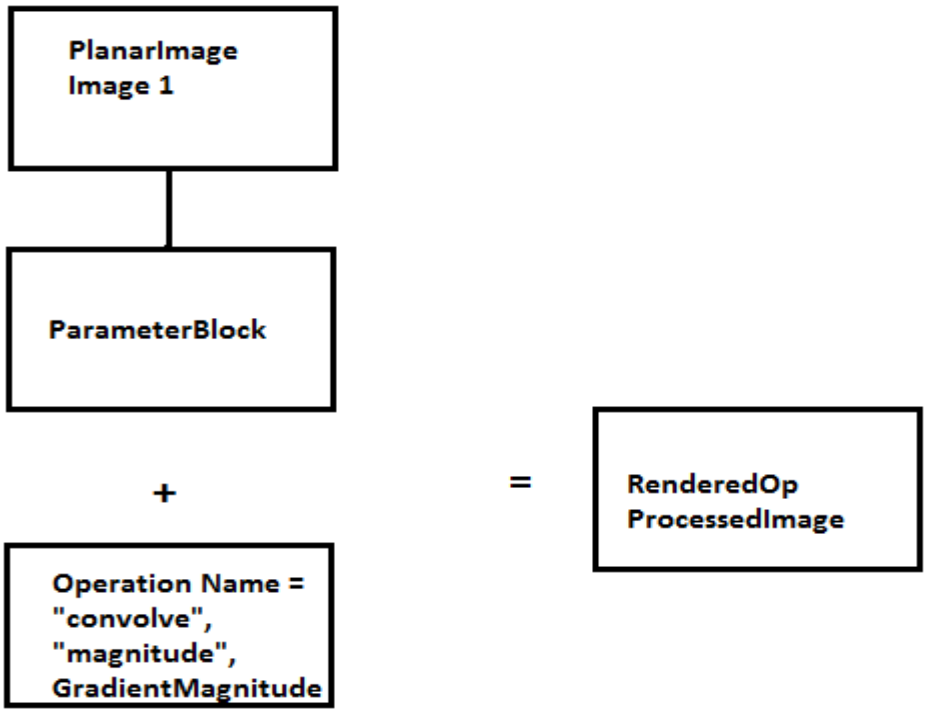
Όπως και με το ιστόγραμμα, η εφαρμογή φίλτρων σε μία εικόνα είναι ένας κόμβος στην αλυσιδωτή επεξεργασία εικόνας. Ο συγκεκριμένος κόμβος, δέχεται σαν όρισμα την εικόνα μέσω του μπλοκ παραμέτρων και η κλάση JAI ενεργοποιεί τον κόμβο αυτόν, δεχόμενη με τη σειρά της σαν όρισμα το όνομα της διαδικασίας.

Όπως είδαμε και στα προηγούμενα, η εφαρμογή των φίλτρων έχει τις ίδιες μαθηματικές επεξεργασίες κατά περίπτωση με μόνη διαφορά τα περιεχόμενα

και το μέγεθος της μάσκας. Επίσης όλα τα φίλτρα εφαρμόζονται για τις εξής περιπτώσεις : κατά την οριζόντια διεύθυνση, πρώτης παραγώγου και δευτέρας παραγώγου. Επομένως τρείς είναι οι κόμβοι της αλυσιδωτής επεξεργασίας που ενεργοποιούνται κατά περίπτωση εδώ :

- Οριζόντια συνέλιξη : Convolve
- Πρώτη παράγωγος : GradientMagnitude

Παρακάτω φαίνεται η διαδικασία που ακολουθείται κατά περίπτωση :



Εικόνα 3.5 : Κόμβος συνέλιξης εικόνας

Παρατίθενται παρακάτω κάποια δείγματα για τις προηγούμενες επεξεργασίες. Οι εικόνες είναι από το κανάλι 1 του Landsat για το νησί της Σκιάθου

Ο κώδικας για τις συγκεκριμένες διαδικασίες βρίσκεται στο παράρτημα

3.2.6 Αλγόριθμος K-means

Σε αντίθεση με τις προηγούμενες διαδικασίες, η διαδικασία μη επιβλεπόμενης ταξινόμησης με τον αλγόριθμο K-means δεν αποτελεί κόμβο της αλυσιδωτής επεξεργασίας εικόνας. Για το σκοπό αυτό αναπτύχθηκε κώδικας ο οποίος έχει σκοπό την μη επιβλεπόμενη ταξινόμηση. Ο κώδικας αυτός περιλαμβάνει δύο κλάσεις. Η μία έχει την ευθύνη της επεξεργασίας εικόνας και η άλλη χρησιμοποιείται για την εισαγωγή των παραμέτρων. Η δεύτερη καλεί την

πρώτη και τις μεθόδους της. Η αρχιτεκτονική αυτή είναι βολική διότι με τον ίδιο τρόπο η πρώτη μπορεί να κληθεί και από κλάσεις του Geoserver και με αυτόν τον τρόπο να ενσωματωθεί σε αυτόν. Ονομάζεται KmeansImageClustering και περιλαμβάνει τις παρακάτω μεθόδους :

- getExtrema : Παίρνει σαν όρισμα την αρχική εικόνα και επιστρέφει έναν πίνακα στοιχείων ακρίβειας double τα οποία αντιστοιχούν στο μέγιστο και το ελάχιστο στοιχείο κάθε εικόνας
- run : Μέθοδος η οποία πραγματοποιεί την κύρια επεξεργασία της ταξινόμησης. Είναι επαναληπτική και υλοποιεί τα βήματα του αλγορίθμου K-means όπως περιγράφηκε παραπάνω. Δεν επιστρέφει κάποιο αποτέλεσμα, αλλά με την επεξεργασία που πραγματοποιεί γεμίζει ένα πίνακα δύο διαστάσεων ο οποίος έχει το μέγεθος της εικόνας και σε κάθε στοιχείο του περιέχει την κλάση στην οποία ανήκει το συγκεκριμένο pixel.
- getClassFor : Η συγκεκριμένη βοηθητική μέθοδος παίρνει σαν όρισμα έναν πίνακα ακέραιων στοιχείων, που αντιστοιχεί σε ένα pixel. Η δουλειά της είναι να υπολογίσει την απόσταση του pixel αυτού από το κέντρο κάθε κλάσης (όπως ορίστηκε στο προηγούμενο κεφάλαιο) και να επιστρέψει την κλάση στην οποία ανήκει. Χρησιμοποιείται από την επόμενη μέθοδο
- getOutput : Η συγκεκριμένη μέθοδος έχει το καθήκον να επιστρέψει την ταξινομημένη εικόνα. Για να το φέρει σε πέρας δημιουργεί μία καινούργια άδεια εικόνα με τα αντίστοιχα ColorModel και Raster (στο σημείο αυτό δόθηκε ιδιαίτερη βαρύτητα, καθώς οι κλάσεις που χειρίζονται τα δεδομένα εξόδου του Geoserver «καταλαβαίνουν» μόνο ένα είδος SampleModel, το PixelInterleaved SampleModel). Έπειτα διατρέχει την αρχική εικόνα και για κάθε pixel υπολογίζει την κλάση στην οποία ανήκει με την προηγούμενη μέθοδο. Έπειτα τοποθετεί την κλάση αυτή στο αντίστοιχο pixel της άδειας εικόνας.

Ένα δείγμα των αποτελεσμάτων αυτού του αλγόριθμου για 2 κλάσεις, 10 επαναλήψεις και παράγοντα σύγκλισης 1 φαίνονται παρακάτω :

Εικόνα 3.6 : Εικόνα των Σποράδων από το κανάλι 3 του Landsat

Εικόνα 3.7 : Η ταξινομημένη εικόνα

Τέλος, να σημειωθεί ότι ο κώδικας της συγκεκριμένης υλοποίησης βρίσκεται το παράρτημα.

3.2.7 Pansharpening

Η διαδικασία Pansharpening δεν αποτελεί κόμβο της αλυσιδωτής επεξεργασίας εικόνας της βιβλιοθήκης JAI. Μπορούν όμως να χρησιμοποιηθούν αλυσιδωτά άλλες υπάρχουσες διαδικασίες της βιβλιοθήκης. Στο προηγούμενο κεφάλαιο εξετάστηκε ο τρόπος και ο σκοπός της συγκεκριμένης επεξεργασίας. Παρακάτω αναλύονται οι κόμβοι της βιβλιοθήκης JAI που χρησιμοποιούνται :

- Κόμβος bandmerge : Συγχώνευση καναλιών σε μία εικόνα
- Κόμβος scale : Διπλασιασμός της ανάλυσης της αρχικής εικόνας
- Κόμβος colorconvert : μετατροπή της εικόνας από το σύστημα RGB στο σύστημα απόχρωσης και κορεσμού IHS
- Κόμβος bandselect : Διαχωρισμός των τριών καναλιών του συστήματος HIS για τη λήψη του καναλιού της απόχρωσης και του κορεσμού.
- Κόμβος bandmerge : Ένωση των καναλιών της απόχρωσης και του κορεσμού σε μία εικόνα
- Κόμβος colorconvert : Επαναφορά της δημιουργημένης εικόνας στο σύστημα RGB

Παρακάτω φαίνεται η ίδια εικόνα των Σποράδων μετά την επεξεργασία της από τον παραπάνω αλγόριθμο

Εικόνα 3.8 : Pansharpening σε εικόνα των Σποράδων

3.3 Ενσωμάτωση αλγορίθμων στην υπηρεσία WPS

Στο κεφάλαιο αυτό θα εξετάσουμε τον τρόπο με τον οποίο οι παραπάνω κλάσεις που αναπτύχθηκαν σε Java και δουλεύουν σαν απλά προγράμματα, μπορούν να γίνουν μέρος του Geoserver ως επεκτάσεις στην υπηρεσία WPS και όχι μόνο. Παράλληλα θα αναλυθεί ο τρόπος με τον οποίο κάποιες τηλεπισκοπικές μέθοδοι μπορούν να εφαρμοστούν ήδη στον Geoserver χωρίς να χρειάζεται κάποια διαδικασία επέκτασης.

3.3.1 Γιατί επιλέγουμε την υπηρεσία WPS

Στο δεύτερο κεφάλαιο αναλύθηκε η υπηρεσία WPS και πώς αυτή υλοποιείται στον Geoserver. Είδαμε ότι υπάρχουν ειδικές κλάσεις του Geoserver οι οποίες μπορούν να διαχειριστούν και να αποκωδικοποιήσουν τα δεδομένα εισόδου και ειδικές κλάσεις οι οποίες διαχειρίζονται τα δεδομένα εξόδου και δημιουργούν τις κατάλληλες αναπαραστάσεις από αντικείμενα που είναι αποθηκευμένα στη μνήμη (PPIO). Για παράδειγμα μία εικόνα για να απεικονιστεί πρέπει να περάσει από αυτές τις κλάσεις.

Κατά την υλοποίηση των διαδικασιών σε γλώσσα Java έγινε σαφές ότι σε κάποιο βαθμό θα χρειαστεί να χρησιμοποιήσουμε αυτές τις κλάσεις τόσο για να εισάγουμε δεδομένα, όσο κα για να αναπαραστήσουμε τα αποτελέσματα. Οι άλλες υπηρεσίες που υλοποιούνται στον Geoserver δεν έχουν αυτές τις κλάσεις και κυρίως δεν έχουν τη δυνατότητα να διαχειριστούν τα ψηφιογραφικά δεδομένα.

Επίσης στην υπηρεσία WPS είναι πολύ εύκολο να περάσουμε μεταδεδομένα μαζί με τα δεδομένα που μας αφορούν, καθώς υπάρχουν οι κλάσεις DescribeProcess, DescribeParameter και DescribeResult οι οποίες μπορούν να «ονομάσουν» τα δεδομένα και να είναι έτσι πιο εύκολη η διαχείρισή τους από κάποια Java εφαρμογή .

Τέλος, οι κλάσεις που περιγράφηκαν παραπάνω, συμμετέχουν και χειρίζονται ταυτόχρονα το Spring Framework. Με τον τρόπο αυτό μπορούμε να χειριζόμαστε το γραφικό περιβάλλον εργασίας στο οποίο υλοποιούνται και οι υπόλοιπες λειτουργίες της WPS. Επίσης δίνεται δυνατότητα να επιλέξουμε τις παραμέτρους που χρειάζονται για την μέθοδο που πραγματοποιούμε από φιλικά προς το χρήστη περιβάλλοντα (π.χ. η επιλογή μίας εικόνας που είναι ήδη αποθηκευμένη στον Geoserver μπορεί να γίνει από drop down list).

3.3.2 Επέκταση των ανεπτυγμένων κλάσεων στην υπηρεσία WPS

Σε αυτή την ενότητα θα εξεταστούν οι κλάσεις που χρησιμοποιούνται για την υλοποίηση των τηλεπισκοπικών μεθόδων στην υπηρεσία WPS. Η βασική μορφή των κλάσεων αυτών παραμένει η ίδια με το τοπικό επίπεδο. Οι προσθήκες που απαιτούνται αναλύονται παρακάτω :

- Χρήση των κλάσεων **DescribeProcess, DescribeParameter και DescribeResult** για τον ορισμό τιμών των παραμέτρων :

Σε τοπικό επίπεδο ο τρόπος με τον οποίο περνούν οι τιμές των παραμέτρων σε μία εφαρμογή είναι μέσω της λίστας παραμέτρων του ολοκληρωμένου περιβάλλοντος ανάπτυξης. Η λίστα αυτή καλείται από μία μέθοδο με την ονομασία main η οποία παίρνει σαν όρισμα τη λίστα του ολοκληρωμένου περιβάλλοντος ανάπτυξης. Έπειτα η μέθοδος αυτή καλεί τις παραμέτρους σαν στοιχεία του πίνακα της λίστας των παραμέτρων. Ένα παράδειγμα κλήσης των παραμέτρων φαίνεται παρακάτω στη μέθοδο φίλτρου Sobel :

```
public class Sobel {

        public static void main (String[] args ){

                        .
                        .
                        .
}
}
```

Αντιθέτως, όταν θέλουμε να περάσουμε τις παραμέτρους στην υπηρεσία WPS πρέπει να χρησιμοποιήσουμε τις κλάσεις **DescribeProcess, DescribeParameter και DescribeResult.** Όπως αναφέρθηκε στα προηγούμενα, αυτές ο κλάσεις συνεργάζονται με το Spring Framework το οποίο όπως είδαμε ενσωματώνει ένα σύστημα αναστροφής του ελέγχου. Με το σύστημα αυτό και με τη χρήση των παραπάνω κλάσεων, το Spring εμφανίζει το γραφικό περιβάλλον που επιτρέπει στο χρήστη να περάσει τιμές στις παραμέτρους χωρίς να είναι γνωστές από πριν και χωρίς να εκτελεστεί η κλάση. Το σύστημα αυτό λέγεται και μηχανισμός εκτόξευσης εξαρτήσεων. Για να επιτευχθεί αυτό πρέπει να εισαχθούν σχόλια Java (Java annotation) πριν από κάθε κλάση, πριν από κάθε παράμετρο και πριν από κάθε αποτέλεσμα. Με αυτόν τον τρόπο το Spring ξέρει κατά τη διάρκεια της μεταγλώττισης τις παραμέτρους που θα χρειαστεί κάθε κλάση και το όνομά τους. Ένα παράδειγμα του μηχανισμού αυτού δίνεται παρακάτω για την κλάση που υλοποιεί το φίλτρο Sobel :

```
@DescribeProcess(title="Sobel", description="Edge detection with sobel filter")
public class Sobel implements GeoServerProcess {

        @DescribeResult(name="result", description="output raster")
    public RenderedImage execute(
            @DescribeParameter(name = "coverage", description =
            "Input raster") GridCoverage2D coverage
            )throws ProcessException {

                .
                .
                .
                }
}
```

Παρατηρώντας το παραπάνω τμήμα κώδικα παρατηρούμε το πρώτο Java σχόλιο που πληροφορεί την κλάση DescribeProcess και αυτή με τη σειρά της το Spring για τον τίτλο και την περιγραφή της επεξεργασίας που θα ακολουθήσει.

Επίσης παρατηρούμε ότι η κλάση «εκτελεί» ένα interface που ονομάζεται GeoServerProcess. Το interface αυτό παρέχει πληροφορίες στο Spring για την κατάταξη της συγκεκριμένης διαδικασίας και τι είδος των δεδομένων εξόδου(raster, vector, μικτά). Το συγκεκριμένο interface εκτελείται από διαδικασίες που εξάγουν μικτά αποτελέσματα.

Παρακάτω παρατηρούμε το σχόλιο που πληροφορεί την κλάση DescribeResult το όνομα και την περιγραφή του αποτελέσματος.

Τέλος, η κλάση περιέχει τη μέθοδο execute (σε αντικατάσταση της μεθόδου main που χρησιμοποιείται στο τοπικό επίπεδο) της οποίας το είδος είναι το είδος του αντικειμένου που εξάγεται (εδώ είναι το είδος RenderedImage). Η μέθοδος αυτή παίρνει σαν παραμέτρους ένα αντικείμενο της κλάσης GridCoverage2D της βιβλιοθήκης GeoTools το οποίο αντιπροσωπεύει μία γεωαναφερμένη εικόνα. Και εδώ υπάρχει το σχόλιο το οποίο πληροφορεί την κλάση DescribeParameter και αυτή με τη σειρά της το Spring Framework το όνομα και το είδος της παραμέτρου. Το spring εδώ εκτοξεύει αυτόματα το γραφικό περιβάλλον επιλογής του συγκεκριμένου αντικειμένου , που είναι συνήθως μία λίστα με όλα τα Raster Layers που είναι αποθηκευμένα στον Geoserver. Επίσης ονομάζει τη μεταβλητή στην οποία εκχωρείται το συγκεκριμένο Layer και μπορεί με αυτόν τον τρόπο να χρησιμοποιηθεί στο εσωτερικό της κλάσης και της μεθόδου. Εδώ ονομάζει αυτήν τη μεταβλητή coverage.

Ακολουθεί η εκτέλεση του κυρίως μέρους του κώδικα ο οποίος είναι παρόμοιος με την εκτέλεση σε τοπικό επίπεδο. Η μόνη διαφορά είναι ο τρόπος επιστροφής των δεδομένων όπου εδώ, αντί για κωδικοποίηση και αποθήκευση, τα δεδομένα επιστρέφονται με την εντολή return στις κλάσεις PPIO , αποκωδικοποιούνται και προβάλλονται σαν επιστροφή στο χρήστη.

- Μετά την ανάπτυξη των κλάσεων απαιτείται να γίνει η διαδικασία του εγκιβωτισμού με το εργαλείο maven. Το πρώτο που πρέπει να γίνει είναι η διαδικασία δημιουργίας Java Project με το maven. Αυτό γίνεται με την παρακάτω εντολή για την περίπτωση του φίλτρου Sobel :

```
mvn archetype:generate -
DgroupId=org.geoserver.edge -DartifactId=sobel
-DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

Η παραπάνω εντολή δημιουργεί την εξής δομή φακέλων :

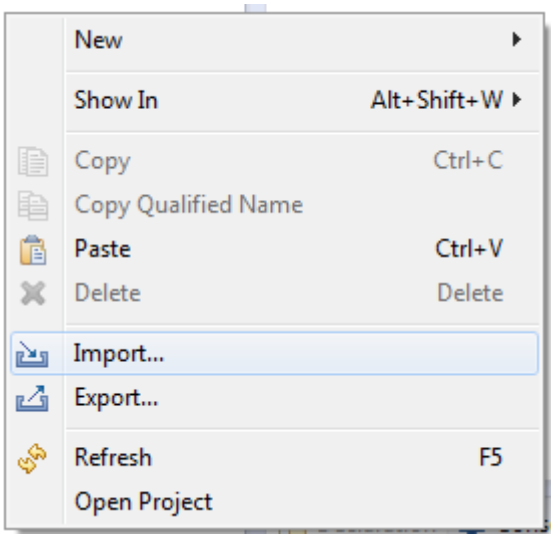
```
sobel
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- org
    |           `-- geoserver
    |               `-- edge
    |
    `-- test
        `-- java
            `-- org
                `-- geoserver
                    `-- edge
```

Ο κώδικας που αναπτύξαμε πρέπει να βρίσκεται στη διεύθυνση src/main/java/org/geoserver/edge. Επίσης οι κλάσεις ελέγχου θα πρέπει να βρίσκονται στην διεύθυνση src/test/java/org/geoserver/edge. Για να ολοκληρωθεί η δημιουργία του project, θα πρέπει αυτό να εισαχθεί στο eclipse. Είναι ασύμφορο να γραφεί ο κώδικας έξω από ένα περιβάλλον ολοκληρωμένης διαχείρισης (eclipse) καθώς αυτό μας προσφέρει πλεονεκτήματα ελέγχου, αποσφαλμάτωσης, αυτόματης διόρθωσης των συντακτικών σφαλμάτων και δυνατότητα άμεσης επισκόπησης του java api. Για το λόγο αυτό, δίνεται η παρακάτω εντολή στο τερματικό :

```
mvn eclipse:eclipse
```

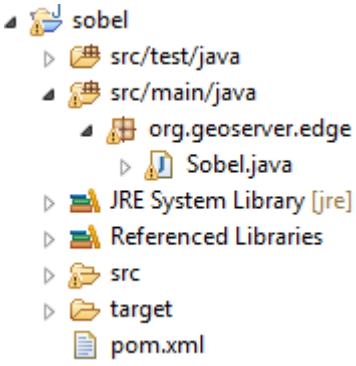
Με την παραπάνω εντολή δημιουργούνται τα αρχεία .classpath και .project τα οποία πληροφορούν το eclipse για τη δομή των φακέλων και την διεύθυνση του πηγαίου κώδικα.

Στη συνέχεια, από την αρχική σελίδα του eclipse κατευθυνόμαστε στην αριστερή στήλη όπου είναι τοποθετημένα τα διάφορα project επιλέγουμε την εντολή import



Εικόνα 3.9 : Η εντολή import

Τέλος μετά την εισαγωγή στο eclipse, το project που έχουμε εισάγει φαίνεται στην αριστερή στήλη :



Εικόνα 3.10 : Το project Sobel

Στη συνέχεια, αφού αναπτυχθούν οι κύριες κλάσεις και οι κλάσεις ελέγχου, πρέπει να συνταχθούν τα αρχεία pom.xml και applicationContext.xml . Το αρχείο pom περιέχει τις πληροφορίες για το που βρίσκεται κάθε κλάση (σε ποιο διαδικτυακό τόπο αποθήκευσης) . Αναλύθηκε στο προηγούμενο κεφάλαιο η δομή και τα περιεχόμενα ενός αρχείου pom. Για τις μεθόδους που χρησιμοποιήθηκαν, χρειάζονται πανομοιότυπες εξωτερικές κλάσεις (που δεν περιέχονται στο Java api) . Επομένως η μορφή του pom είναι παρόμοια :

```
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.geoserver.edge</groupId>
  <artifactId>sobel</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>sobel</name>
  <url>http://maven.apache.org</url>
```

Στις συγκεκριμένες ετικέτες δίνονται πληροφορίες στο maven σχετικά με το όνομα που θα έχει το project, τη συμπίεσή του (εδώ θα είναι jar) και τον αριθμό έκδοσής του.

```
    <dependency>
      <groupId>org.geotools</groupId>
      <artifactId>gt-process</artifactId>
      <version>10-SNAPSHOT</version>
    </dependency>

    <dependency>
      <groupId>org.geotools</groupId>
      <artifactId>gt-process-raster</artifactId>
      <version>10.0</version>
    </dependency>

    <dependency>
      <groupId>org.geotools</groupId>
      <artifactId>gt-coverage</artifactId>
      <version>10-SNAPSHOT</version>
    </dependency>
```

Σε αυτό το τμήμα εκφράζεται η εξάρτηση (dependency) που έχει η κλάση που δημιουργήσαμε από τη βιβλιοθήκη GeoTools και συγκεκριμένα από τις ενότητες gt-process, gt-process-raster, gt-coverage. Επίσης, δίνει πληροφορίες για τον αριθμό της έκδοσής τους

```
<dependency>
  <groupId>org.geoserver.extension</groupId>
  <artifactId>wps-core</artifactId>
  <version>2.4-SNAPSHOT</version>
</dependency>

<dependency>
  <groupId>org.geoserver</groupId>
  <artifactId>main</artifactId>
  <version>2.5-SNAPSHOT</version>
  <classifier>tests</classifier>
  <scope>test</scope>
</dependency>
```

Σε αυτό το κομμάτι του pom εκφράζεται η εξάρτηση από το κομμάτι του Geoserver που αποτελεί τον κύριο πυρήνα της υπηρεσίας WPS. Αυτό το κομμάτι είναι που περιέχει όλες τις κλάσεις που χρειάζονται ώστε να αναπαρασταθεί ένα αντικείμενο της μνήμης σε οπτική αναπαράσταση. Επίσης εκφράζεται η εξάρτηση της κλάσης μας από τον κύριο πυρήνα του Geoserver. Αυτός ο πυρήνας περιέχει όλες τις κλάσεις που απαιτούνται προκειμένου να αναπαρασταθεί η κλάση μας σαν τμήμα του Geoserver και να δημιουργηθεί το κατάλληλο γραφικό περιβάλλον για αυτή. Τέλος αναφέρονται οι αριθμοί της έκδοσης τους.

```
<build>
 <plugins>
  <plugin>
   <artifactId>maven-compiler-plugin</artifactId>
   <configuration>
    <source>1.6</source>
    <target>1.6</target>
   </configuration>
  </plugin>
 </plugins>
</build>

<repositories>
 <repository>
  <id>opengeo</id>
  <name>opengeo</name>
  <url>http://repo.opengeo.org</url>
 </repository>
</repositories>
```

Σε αυτό το κομμάτι του pom, υπάρχουν οι πληροφορίες για το που είναι αποθηκευμένες οι κλάσεις που αναφέρθηκαν ήδη (στο online repository του OpenGeo στη διεύθυνση http://repo.opengeo.org). Επίσης, δίνονται οι πληροφορίες για την έκδοση του Apache maven.

Σε αυτό το σημείο έχει ολοκληρωθεί η διαδικασία της σύνταξης του pom και πρέπει να γίνει η σύνταξη του applicationContext. Το αρχείο αυτό είναι απαραίτητο προκειμένου να πληροφορηθεί το σύστημα διαχείρισης περιεχομένου (εδώ το Spring Framework) ότι η κλάση μας είναι μέρος του συστήματος. Επίσης παρέχει πληροφορίες στο Spring για τον τρόπο με τον οποίο η κλάση θα αποκτήσει το ανάλογο γραφικό περιβάλλον απεικόνισης. Η γενική μορφή του applicationContext για τις λειτουργίες της WPS είναι :

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
 <beans>
   <bean id="classId" class="classpath"/>
 </beans>
```

Το applicationContext ενημερώνει το Spring ότι θα πρέπει να δημιουργήσει ακόμα ένα Bean (κόμβο). Ο κόμβος αυτός θα έχει το όνομα classId και η κλάση θα πρέπει να βρίσκεται στη διεύθυνση classpath.

Στην ειδική μορφή του φίλτρου Sobel έχουμε το ακόλουθο applicationContext

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
 <beans>
   <bean id="SobelFilter" class="org.geoserver.edge.Sobel"/>
 </beans>
```

Το maven, με δεδομένα το pom και το applicationContext πραγματοποιεί τον «εγκιβωτισμό». Με το τερματικό στο φάκελο που περιέχει το pom δίνουμε την εντολή :

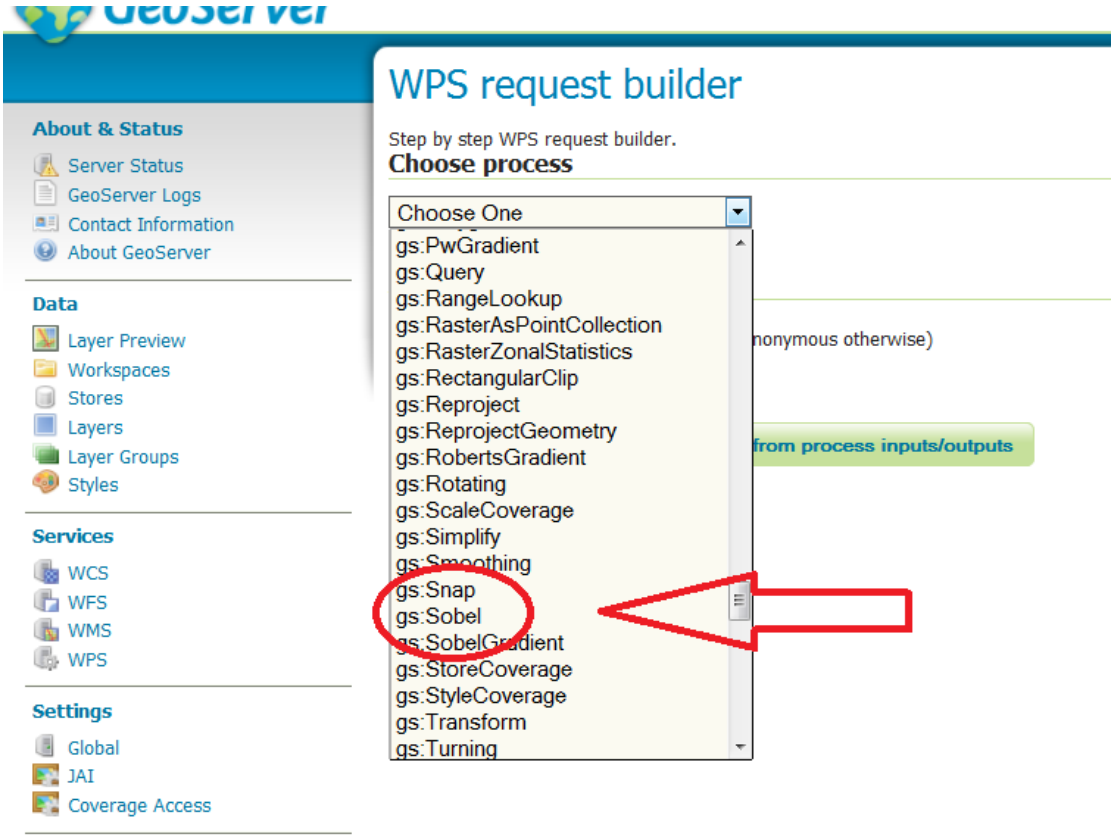
```
mvn clean install
```

Με την εντολή αυτή, το maven αρχίζει και συνδέει τις παραπάνω εργασίες μεταξύ τους. Αρχικά συνδέεται στο repository που του υποδεικνύει το pom και μεταφορτώνει τις κλάσεις που περιγράφονται σε αυτό. Έπειτα, δημιουργεί τα κατάλληλα σχόλια (annotations) για να ενημερώσει το Spring και να ανανεωθεί το γραφικό περιβάλλον του Geoserver. Τέλος, μεταγλωττίζει τις κλάσεις μας και τις κλάσεις που μεταφόρτωσε. Μετά τη μεταγλώττιση, δημιουργεί ένα νέο υποφάκελο στο φάκελο του project με την ονομασία target. Αυτός ο φάκελος περιέχει το jar αρχείο το οποίο περιέχει όλη την πληροφορία για την ενσωμάτωση της κλάσης στον Geoserver. Σύμφωνα με τα προηγούμενα, η δομή του συστήματος φακέλων είναι η εξής :

```
sobel
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |    `-- org
    |   |         `-- geoserver
    |   |             `-- edge
    |    ‘resources
    `-- test
        `-- java
            `-- org
                `-- geoserver
                    `-- edge
‘—target
    |--Sobel.jar
```

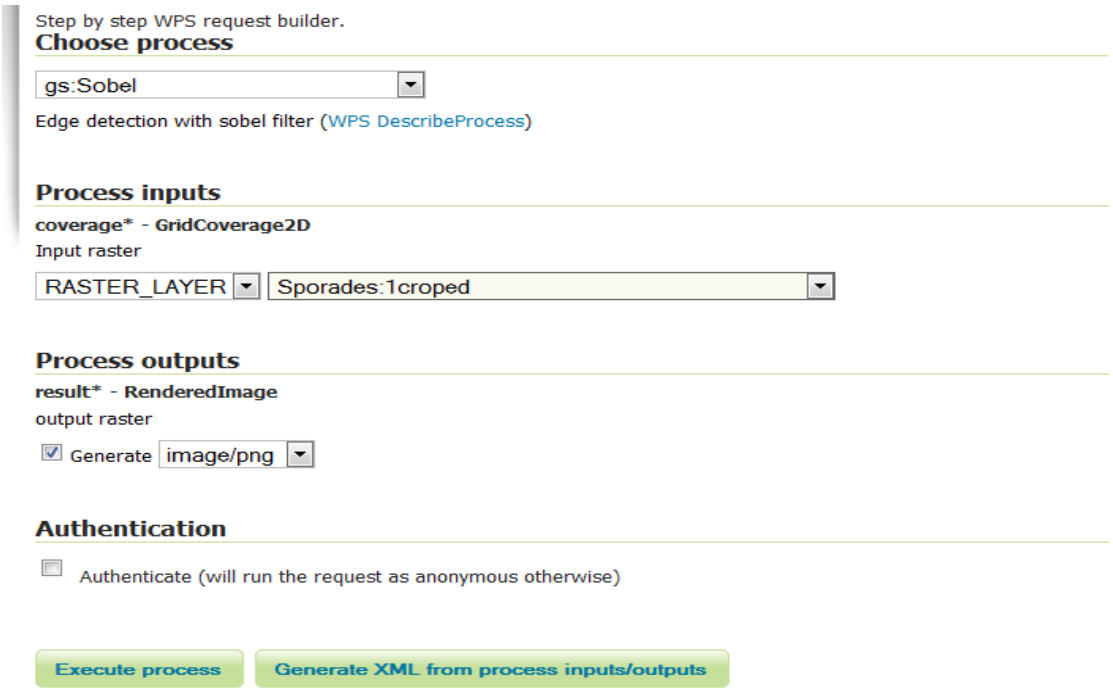
Το αρχείο jar είναι ένα συμπιεσμένο αρχείο (σαν το ZIP). Στην πράξη περιέχει όλες τις κλάσεις που του έχουμε υποδείξει σε μεταγλωττισμένη μορφή (γλώσσα που καταλαβαίνει η εικονική μηχανή της Java). Αποσυμπιέζοντας ένα αρχείο jar με οποιοδήποτε πρόγραμμα αποσυμπίεσης (π.χ. WinRAR) βλέπουμε ότι στο εσωτερικό του έχει την ίδια δομή με το project, με μόνη διαφορά την ύπαρξη στη θέση του πηγαίου κώδικα, μεταγλωττισμένα αρχεία class .

- Το τελικό στάδιο της διαδικασίας επέκτασης του geoserver είναι η προσθήκη των αρχείων jar στη βιβλιοθήκη του Geoserver. Αυτή βρίσκεται στο βασικό φάκελο εγκατάστασης του Geoserver στη διεύθυνση webapps\geoserver\WEB-INF\lib . Σε αυτό το φάκελο βρίσκονται όλα τα αρχεία jar που χρησιμοποιεί ο geoserver κατά τη λειτουργία του. Μεταφέρουμε το αρχείο jar που μόλις δημιουργήθηκε και κάνουμε επανεκκίνηση τον Geoserver. Μετακινούμαστε στην καρτέλα Demos και επιλέγουμε τον WPS request builder. Αυτή η καρτέλα περιέχει μία εφαρμογή ενσωματωμένη στον Geoserver, η οποία χρησιμεύει για να πραγματοποιεί ο χρήστης WPS requests και για να δημιουργείται αυτόματα το XML αρχείο με το οποίο υποβάλλεται το αίτημα στον Geoserver.



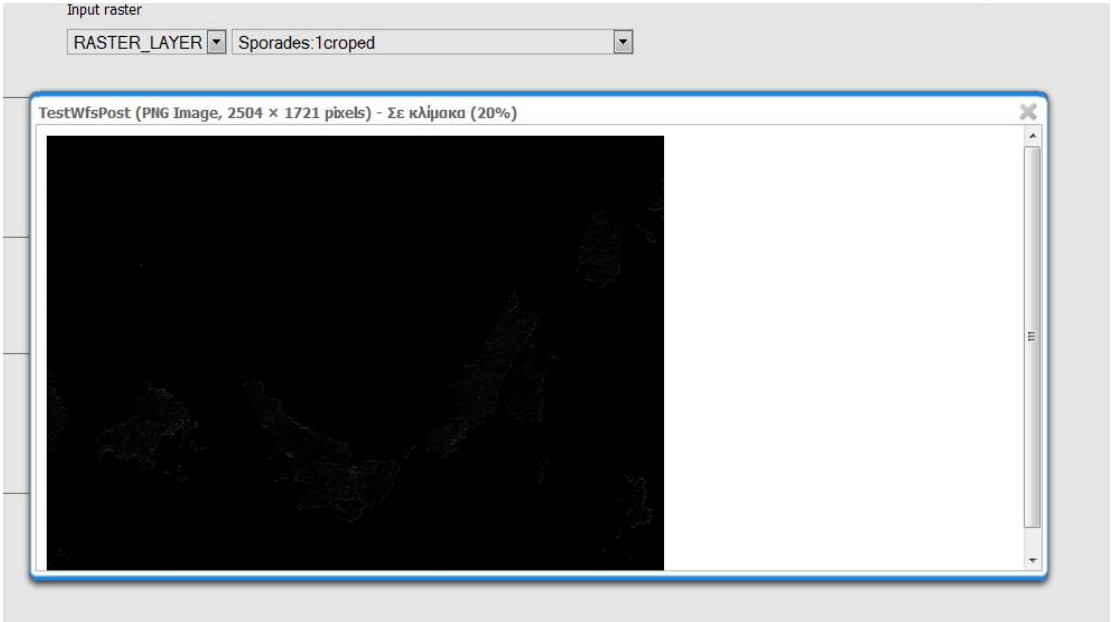
Εικόνα 3.11 : Δημιουργία της διαδικασίας Sobel

Η διαδικασία έχει δημιουργηθεί . Επιλέγοντας την εμφανίζεται η παρακάτω γραφική διεπαφή :



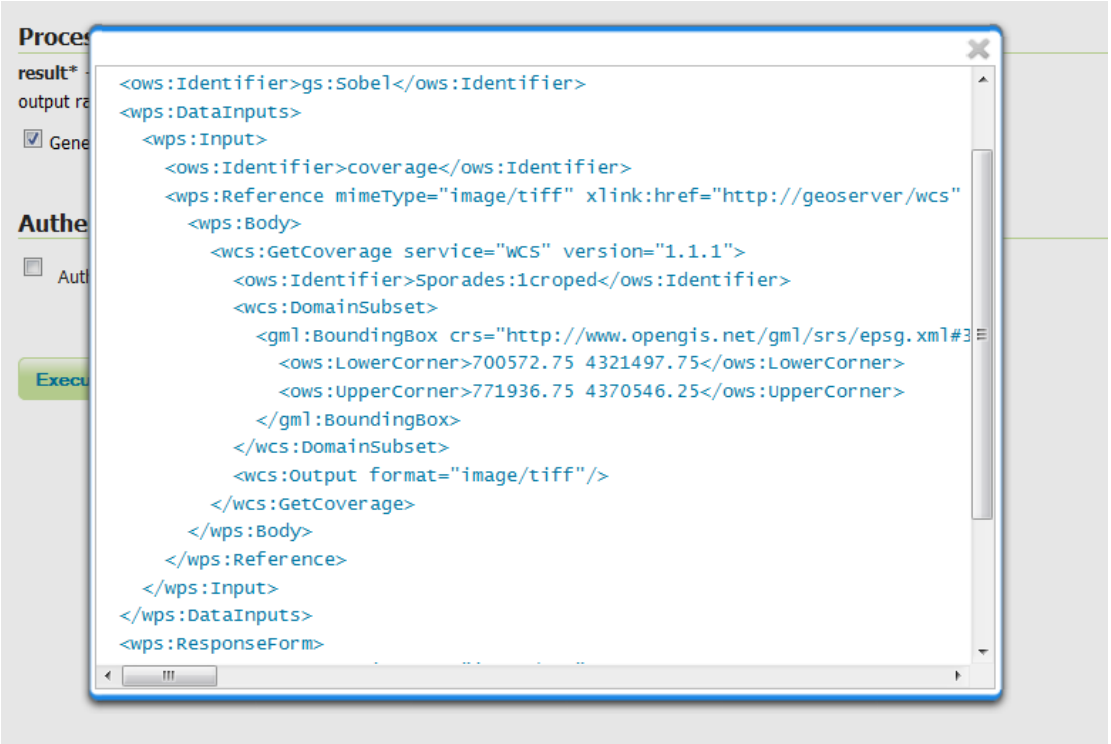
Εικόνα 3.12 : Γραφική διεπαφή της διαδικασίας Sobel

Μπορούμε να επιλέξουμε το raster layer το οποίο θα δοθεί ως όρισμα στη διαδικασία. Επιλέγοντας Execute process παίρνουμε το παρακάτω αποτέλεσμα



Εικόνα 3.13 : Φίλτρο Sobel ως επέκταση του Geoserver

Επίσης επιλέγοντας Generate XML from process inputs/outputs παίρνουμε το παρακάτω αρχείο XML :



Εικόνα 3.14 : Αυτόματη παραγωγή αρχείου XML

Το αρχείο αυτό, αν υποβληθεί στο σώμα ενός POST αιτήματος προς την υπηρεσία WPS, ο Geoserver θα επιστρέψει σαν response την ίδια εικόνα.

Σε αυτό το σημείο, θα πρέπει να τονιστεί ότι ο Geoserver είναι σε στάδιο ανάπτυξης και παρουσιάζει κάποιες φορές μερικά λειτουργικά προβλήματα. Έτσι, κατά την απόπειρα πραγματοποίησης ενός POST αιτήματος στην υπηρεσία WPS από έναν client (π.χ. OpenLayers, GeoNode και άλλους), δεν επιστέφεται η αντίστοιχη απάντηση . Αντιθέτως, η απάντηση επιστρέφεται κανονικά όταν υποβάλλεται από την εφαρμογή curl.

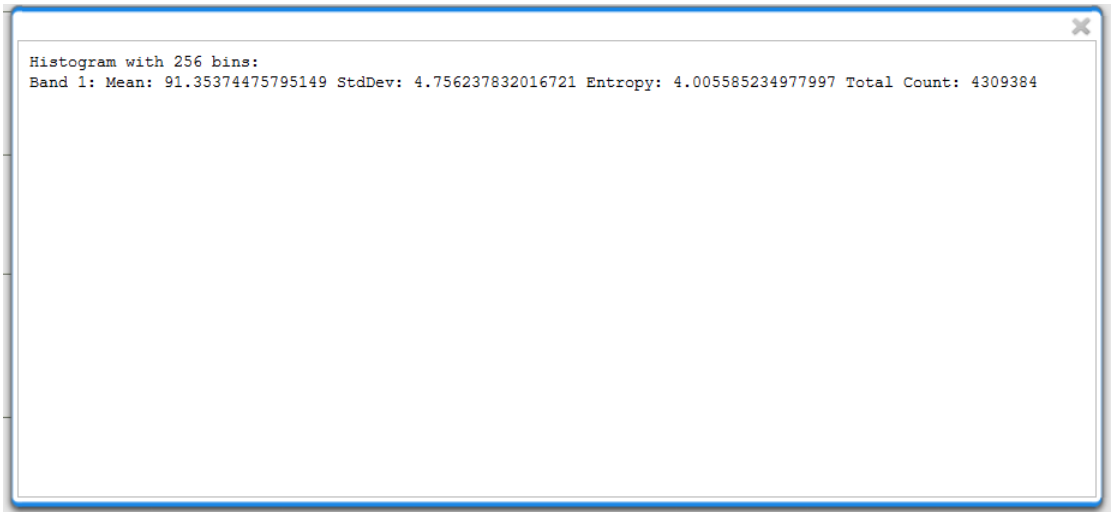
Η curl,είναι μία εφαρμογή η οποία αναπτύχθηκε από προγραμματιστές που ασχολούνται με διαδικτυακό προγραμματισμό. Σκοπός της είναι η άμεση υποβολή αιτημάτων του πρωτοκόλλου http και η άμεση επέμβαση στις επικεφαλίδες και το σώμα των αιτημάτων. Οι συνηθισμένες γλώσσες προγραμματισμού των client εφαρμογών (π.χ. PHP, ASP, JavaScript) είναι πολύ τυποποιημένες και περιορισμένες σχετικά με την επέμβαση σε αυτά τα δεδομένα. Η curl εγκαθίσταται στον τοπικό υπολογιστή και με το τερματικό στο φάκελο εγκατάστασης δέχεται εντολές από αυτό (σαν το maven). Οι εντολές αυτές αρχίζουν με την εντολή curl :

```
curl -H "Content-type: xml" -X POST -d@request.xml
http://localhost:8080/geoserver/wps -o sobel.tiff
```

Με αυτή την εντολή, η curl πραγματοποιεί ένα POST αίτημα στη διεύθυνση που της παρέχουμε. Επίσης, της παρέχονται πληροφορίες για το σώμα του POST αιτήματος , το οποίο είναι αποθηκευμένο στο αρχείο request.xml. Σε αυτό το αρχείο αποθηκεύουμε το XML που παράγεται για τη διαδικασία Sobel από τον request builder. Τέλος η curl αποθηκεύει την απάντηση που παίρνει στο αρχείο sobel.tiff. Να σημειωθεί εδώ ότι το αρχείο request.xml πρέπει να βρίσκεται στο φάκελο εγκατάστασης της curl και το αρχείο sobel.tiff αποθηκεύεται στον ίδιο φάκελο. Με την υποβολή αυτής της εντολής ολοκληρώνεται η λήψη του αρχείου αυτού με επιτυχία. Επομένως διαπιστώνεται ότι ο Geoserver απαιτεί αποσφαλμάτωση στο κομμάτι της αναγνώρισης και κατάταξης ενός αιτήματος προς την WPS από clients που τον χειρίζονται.

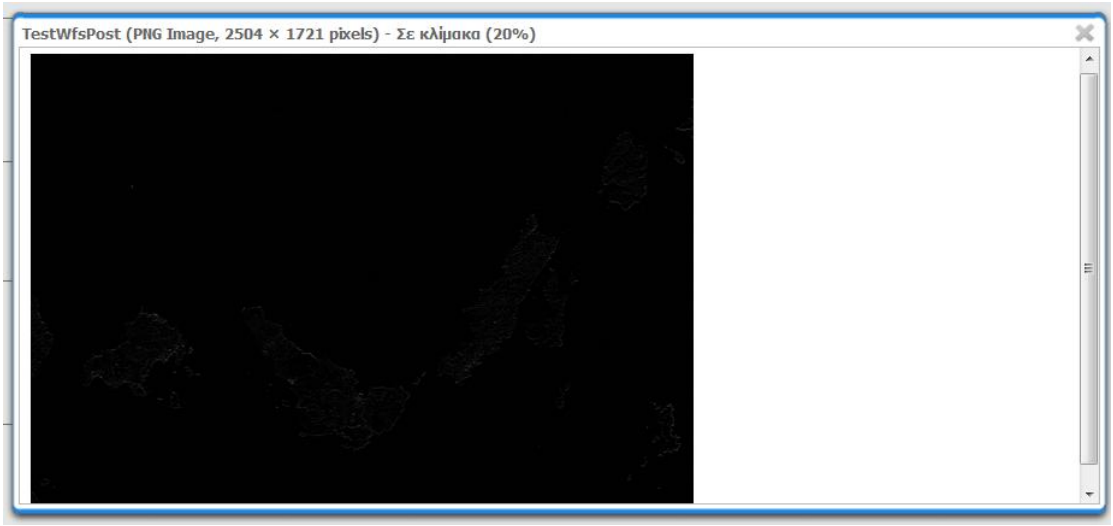
Οι κλάσεις που υλοποιούν τις τηλεπισκοπικές διαδικασίες και αναφέρθηκε η διαδικασία της ανάπτυξης τους στο κεφάλαιο 2, ενσωματώνονται με την ίδια μεθοδολογία στον Geoserver. Αναφέρονται συνοπτικά παρακάτω και προβάλλονται τα αποτελέσματά τους. Τα παραδείγματα αφορούν απεικόνιση των Σποράδων από το κανάλι 1 του Landsat.

- Υπολογισμός αναλυτικών στοιχείων ιστογράμματος :



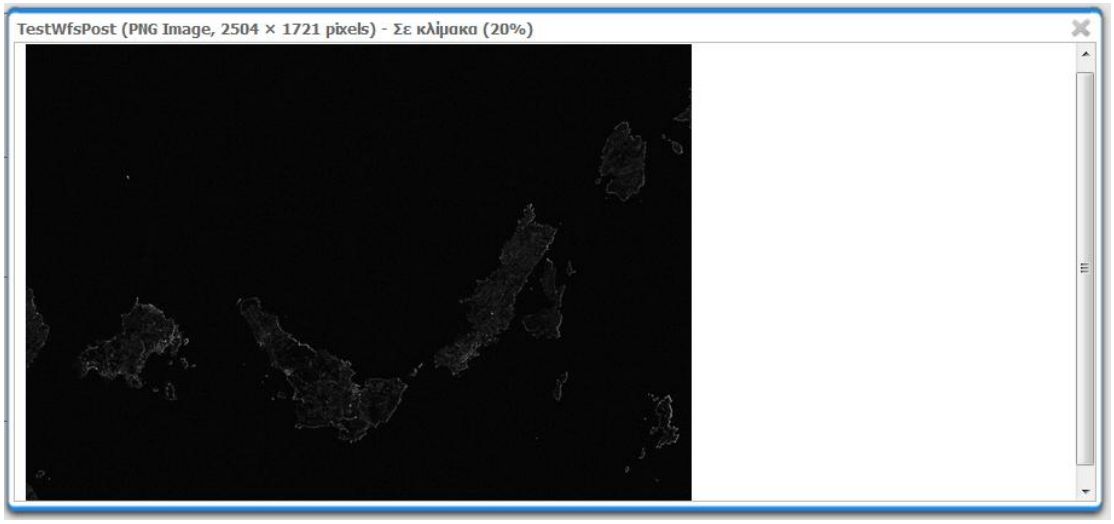
Εικόνα 3.15 : Αναλυτικά στοιχεία ιστογράμματος

- Εφαρμογή οριζόντιου φίλτρου Sobel



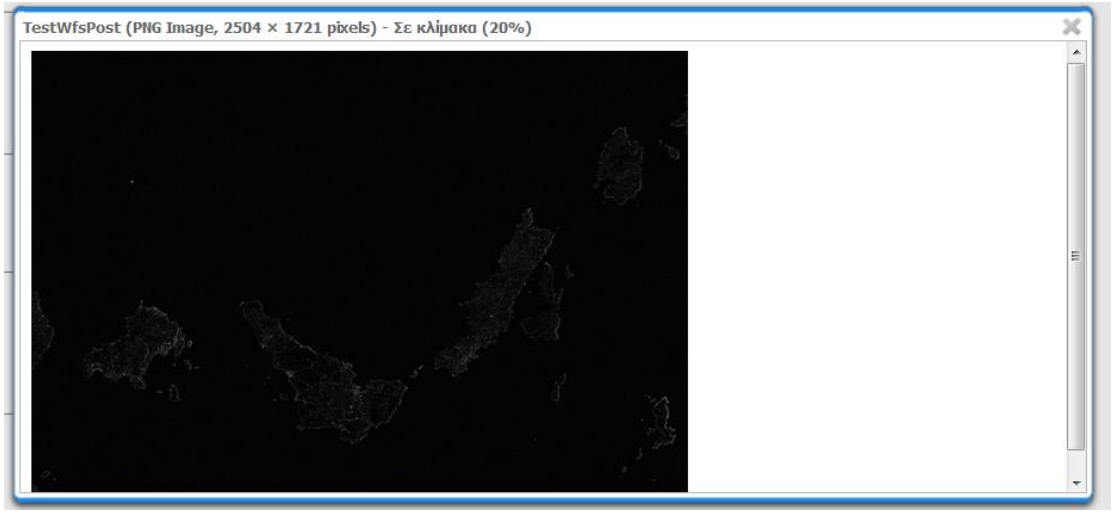
Εικόνα 3.16 : Οριζόντιο φίλτρο Sobel

- Φίλτρο πρώτης παραγώγου με μάσκα Sobel



Εικόνα 3.17 : Φίλτρο πρώτης παραγώγου με μάσκα Sobel

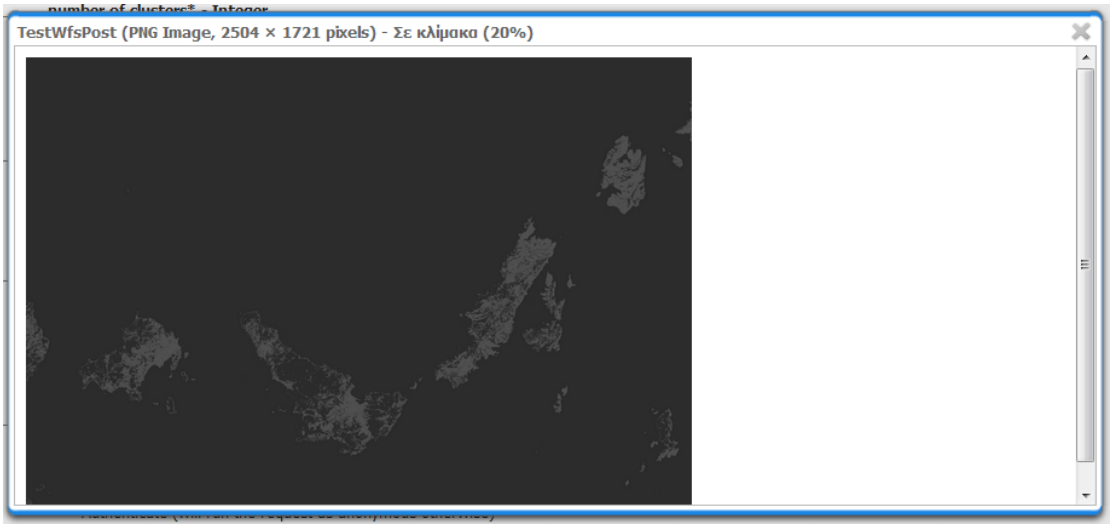
- Φίλτρο πρώτης παραγώγου με μάσκα Prewitt



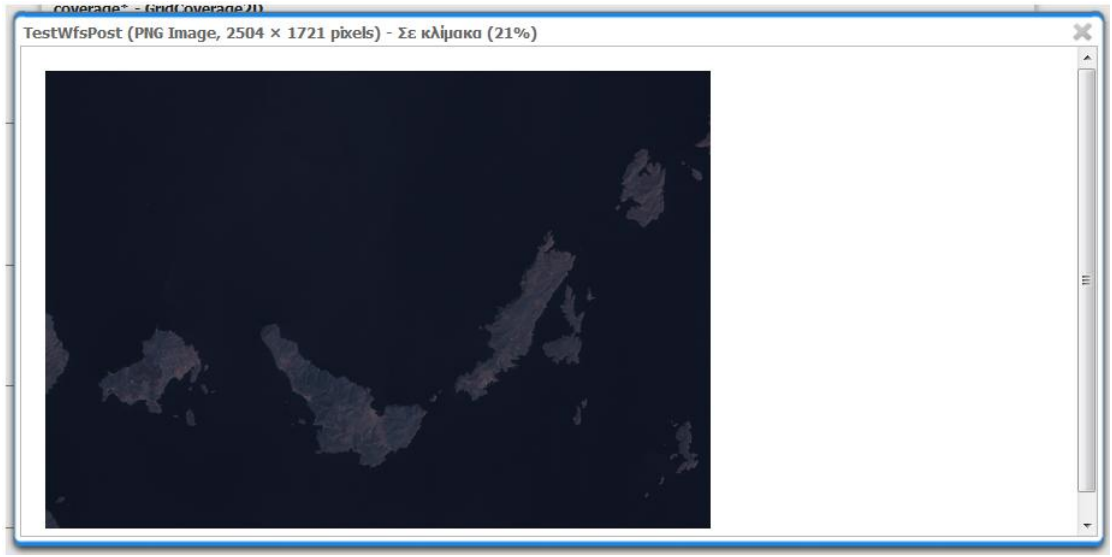
Εικόνα 3.18 : Φίλτρο πρώτης παραγώγου με μάσκα Prewitt

- Φίλτρο πρώτης παραγώγου με μάσκα Roberts

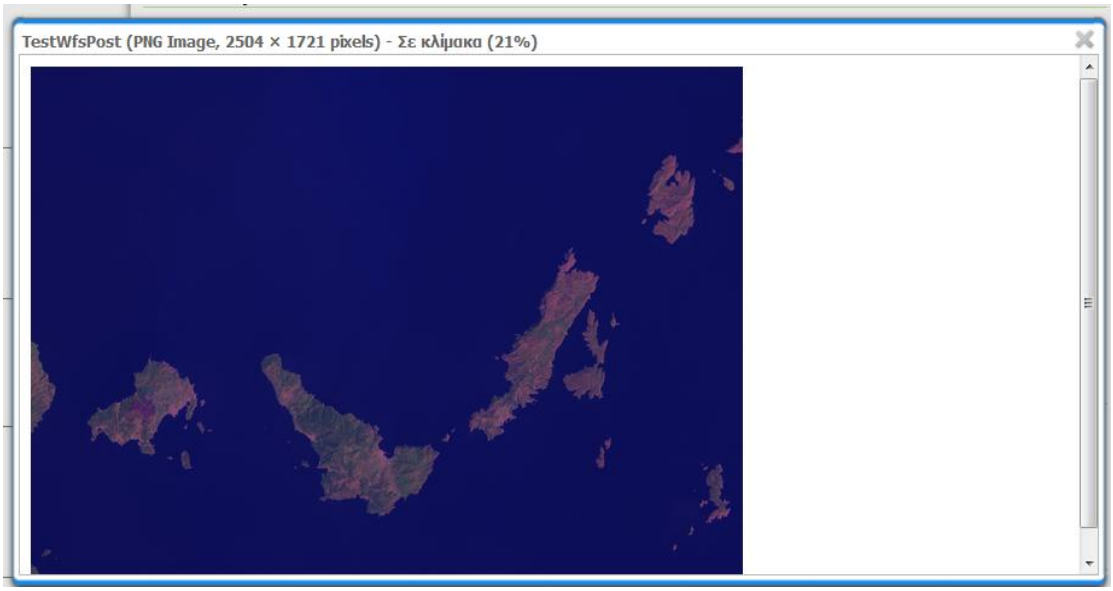
- K means με δύο κλάσεις (κανάλι 3, 3 επαναλήψεις, παράγοντας σύγκλισης 1)



- Pansharpening



Εικόνα 3.19 : Pansharpening



Εικόνα 3.20 : Έγχρωμο σύνθετο 5-4-1

3.3.3 Η περίπτωση της κατωφλίωσης

Η περίπτωση της κατωφλίωσης είναι η μόνη διαδικασία η οποία υλοποιείται ήδη στον Geoserver και δεν χρειάζεται κάποια επέκταση. Η μέθοδος αυτή είναι εύκολο να υλοποιηθεί μέσω της υπηρεσίας WMS.

Όπως περιγράφηκε, η υπηρεσία WMS αφορά την προβολή στατικών και ήδη αποθηκευμένων δεδομένων (ψηφιογραφικά και διανυσματικά). Για να επιτευχθεί αυτό , η υπηρεσία WMS πρέπει να χρησιμοποιήσει τις αποθηκευμένες δομές δεδομένων που μέσω των αρχείων SLD και τις κλάσεις που χειρίζονται τις εξαγωγές και την κωδικοποίηση των δεδομένων. Ένα μεγάλο πλεονέκτημα της υπηρεσίας WMS είναι ότι καλείται από GET αιτήματα αποκλειστικά καθώς δεν χρειάζεται να μεταβληθούν τα πρωτογενή δεδομένα που είναι αποθηκευμένα στον server, απλώς αλλάζει ο τρόπος απεικόνισής τους. Για το λόγο αυτό, τα WMS αιτήματα υποβάλλονται μέσω URL στον server. Για να προβληθεί μία εικόνα με κατώφλι πρέπει να δοθεί το παρακάτω URL :

```
http://localhost:8080/geoserver/RemoteSensing/wms?service=WMS
&version=1.1.0&request=GetMap&layers=RemoteSensing:p183r033
_7t20000824_z34_nn10&styles=Raster_threshold&bbox=595379.25,
4199688.75,843956.25,4417713.75&width=512&height=449&srs=E
PSG:32634&format=application/openlayers
```

Πληκτρολογώντας το παραπάνω URL στο πρόγραμμα περιήγησης, υποβάλλουμε στον Geoserver ένα αίτημα. Ο geoserver ενσωματώνει μία σειρά από κλάσεις, οι οποίες αναγνωρίζουν και αποκωδικοποιούν αυτό το αίτημα. Αυτές οι κλάσεις ονομάζονται KVPparsers. Το παραπάνω URL, αποκωδικοποιημένο παρέχει τις εξής πληροφορίες :

- request = GetMap : Ζητά από τον server την λειτουργία GetMap της υπηρεσίας WPS.
- layers = RemoteSensing:p183r033_7t20000824_z34_nn10 το θεματικό επίπεδο του οποίου ζητείται η προβολή. Κάθε θεματικό επίπεδο που προβάλλεται από την υπηρεσία WMS πρέπει να έχει προηγουμένως αποθηκευτεί στο server.
- styles= Raster_threshold : προβάλλει τα στοιχεία του παραπάνω θεματικού επιπέδου με το στυλ Raster_threshold. Και σε αυτήν την περίπτωση πρέπει το αρχείο sld που περιέχει το στυλ να έχει αποθηκευτεί στον server.
- bbox=595379.25,4199688.75,843956.25,4417713.75 : Οι γεωδαιτικές συντεταγμένες της πάνω αριστερής και της κάτω δεξιάς γωνίας του πλαισίου μέσα στο οποίο θα προβληθεί το layer. Να σημειωθεί ότι το layer πρέπει να είναι γεωαναφερμένο.

Με δεδομένες αυτές τις πληροφορίες, ο Geoserver, απαντά στο GET αίτημα με την προβολή της παρακάτω εικόνας :

Εικόνα 3.21 : Κατωφλίωση σε εικόνα της Εύβοιας

Επίσης το αρχείο sld που ορίζει την κατωφλίωση της εικόνας είναι το παρακάτω :

```
<?xml version="1.0" encoding="UTF-8"?>
<sld:StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
xmlns:sld="http://www.opengis.net/sld"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml" version="1.0.0">
   <sld:UserLayer>
      <sld:LayerFeatureConstraints>
         <sld:FeatureTypeConstraint/>
      </sld:LayerFeatureConstraints>
      <sld:UserStyle>
         <sld:Name>1croped</sld:Name>
         <sld:Title/>
         <sld:FeatureTypeStyle>
            <sld:Name>name</sld:Name>
            <sld:Rule>
               <sld:RasterSymbolizer>
                  <sld:Geometry>
                     <ogc:PropertyName>grid</ogc:PropertyName>
                  </sld:Geometry>
                  <sld:ColorMap>
                     <sld:ColorMapEntry color="#FFFFFF" opacity="1.0"
quantity="0.0"/>
                     <sld:ColorMapEntry color="#FFFFFF" opacity="1.0"
quantity="84.5"/>
                     <sld:ColorMapEntry color="#000000" opacity="1.0"
quantity="84.5"/>
                     <sld:ColorMapEntry color="#000000" opacity="1.0"
quantity="255.0"/>
                  </sld:ColorMap>
               </sld:RasterSymbolizer>
            </sld:Rule>
         </sld:FeatureTypeStyle>
      </sld:UserStyle>
   </sld:UserLayer>
</sld:StyledLayerDescriptor>
```

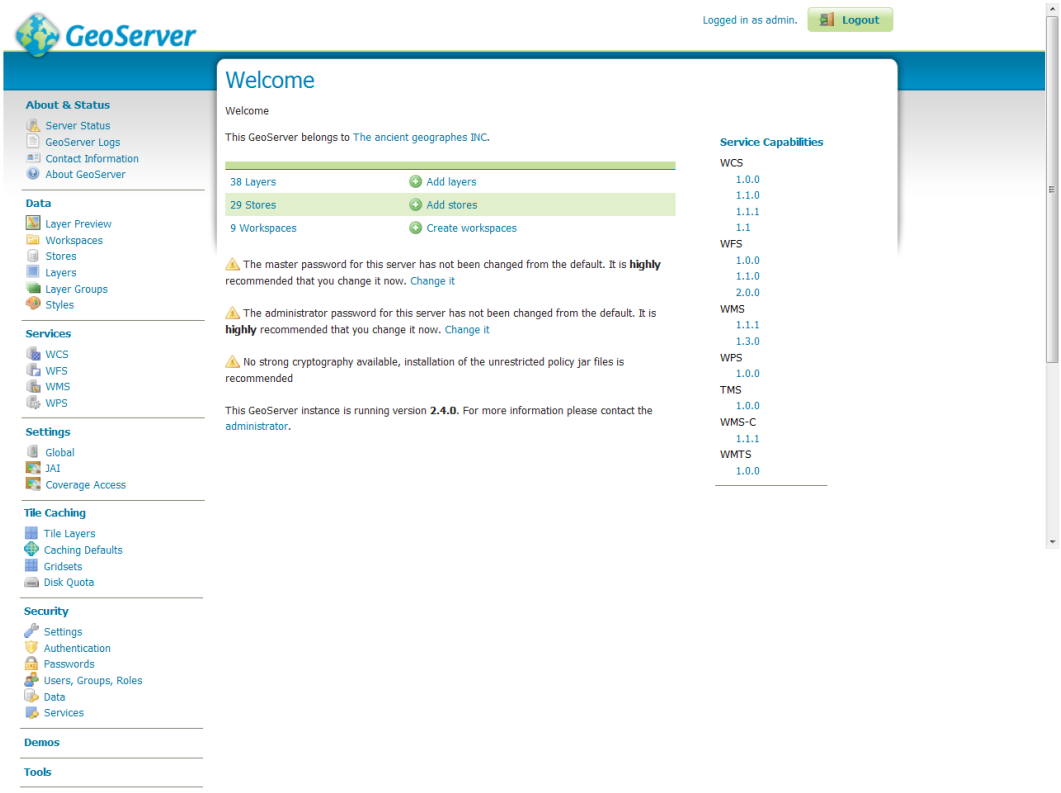
Στην ετικέτα sld:ColorMap καθορίζονται οι παράμετροι της κατωφλίωσης. Στη συγκεκριμένη περίπτωση, όλα τα εικονοστοιχεία που έχουν τιμή χρώματος μικρότερη από 84.5 απεικονίζονται άσπρα και όσα έχουν μεγαλύτερη, απεικονίζονται μαύρα. Να σημειωθεί ότι το συγκεκριμένο αρχείο μπορεί να χρησιμοποιηθεί για την προβολή πολλών θεματικών επιπέδων, κάτι που αυξάνει την χρηστικότητά του.

4. Αξιολόγηση των αποτελεσμάτων

Στην ενότητα αυτή θα εξεταστεί η αποτελεσματικότητα της διαδικασίας που αναπτύχθηκε στα προηγούμενα και θα αξιολογηθούν τα αποτελέσματά της. Στο πρώτο κεφάλαιο θα παρουσιαστεί το περιβάλλον της εφαρμογής και η λειτουργικότητα του σε σχέση με τις υλοποιημένες διαδικασίες. Στη συνέχεια θα παρουσιαστούν οι εφαρμογές ξεχωριστά και θα εξεταστεί η χρησιμότητα και η εκπλήρωση ή μη του σκοπού για τον οποίο αναπτύχθηκαν.

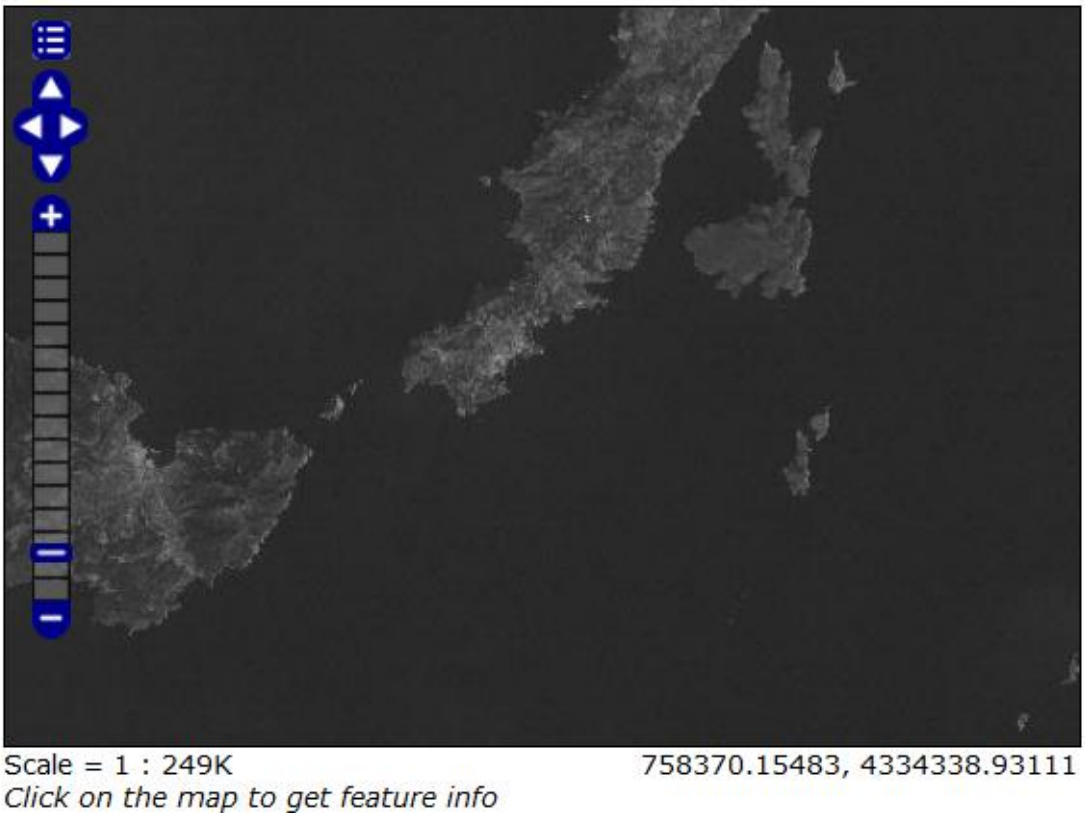
4.1 Το περιβάλλον

Ο Geoserver κατά τα προηγούμενα είναι ένας server χωρικών δεδομένων. Αυτό σημαίνει ότι ο κύριος σκοπός του είναι να εξυπηρετεί αιτήματα από clients και να τους παρέχει τις ανάλογες απαντήσεις. Παρόλα αυτά, ο Geoserver ενσωματώνει ένα γραφικό περιβάλλον, το οποίο επιτρέπει τον εύκολο χειρισμό των δεδομένων (π.χ. αποθήκευση δεδομένων, αλλαγή των στυλ, επεξεργασία). Αυτό το περιβάλλον είναι δημιουργημένο με τη γλώσσα HTML. Αποτελεί δηλαδή έναν ιδιότυπο client που χειρίζεται τον server (με τη μόνη διαφορά ότι ο client αυτός είναι ενσωματωμένος στον server). Παρακάτω παρουσιάζεται το περιβάλλον :



Εικόνα 4.1 : Το γραφικό περιβάλλον του Geoserver

Στην αριστερή στήλη βρίσκεται το γραφικό περιβάλλον των εντολών που δέχεται ο server. Στην καρτέλα Data ο χρήστης μπορεί να χειριστεί τα δεδομένα που είναι αποθηκευμένα στον server και να μεταφορτώσει τα δικά του (raster και vector). Κατά την επιλογή αυτής της καρτέλας, ο χρήστης πρέπει να επιλέξει τη μορφή των δεδομένων προς εισαγωγή (π.χ. GEOTIFF, Shapefile, CSV και άλλα). Κατά την εισαγωγή της πηγής των δεδομένων που γίνεται με αναφορά (δίνεται η διεύθυνση του αρχείου στο υπολογιστή του χρήστη) ο Geoserver αναγνωρίζει τα δεδομένα και δημιουργεί το θεματικό επίπεδο για κάθε πηγή δεδομένων. Τα θεματικά επίπεδα που δημιουργούνται, μπορούν να γίνουν ορατά από την επιλογή Layer preview. Ο χρήστης μπορεί να επιλέξει τον τρόπο προβολής του κάθε θεματικού επιπέδου (PNG, JPEG, OpenLayers, GIF και άλλους). Στην περίπτωση μας εισάγουμε στον server μία πολυφασματική εικόνα των βόρειων Σποράδων σε μορφή GeoTiff και προβάλουμε το τρίτο κανάλι της σε μορφή OpenLayers :



Εικόνα 4.2 : Προβολή του τρίτου καναλιού εικόνας των βόρειων Σποράδων

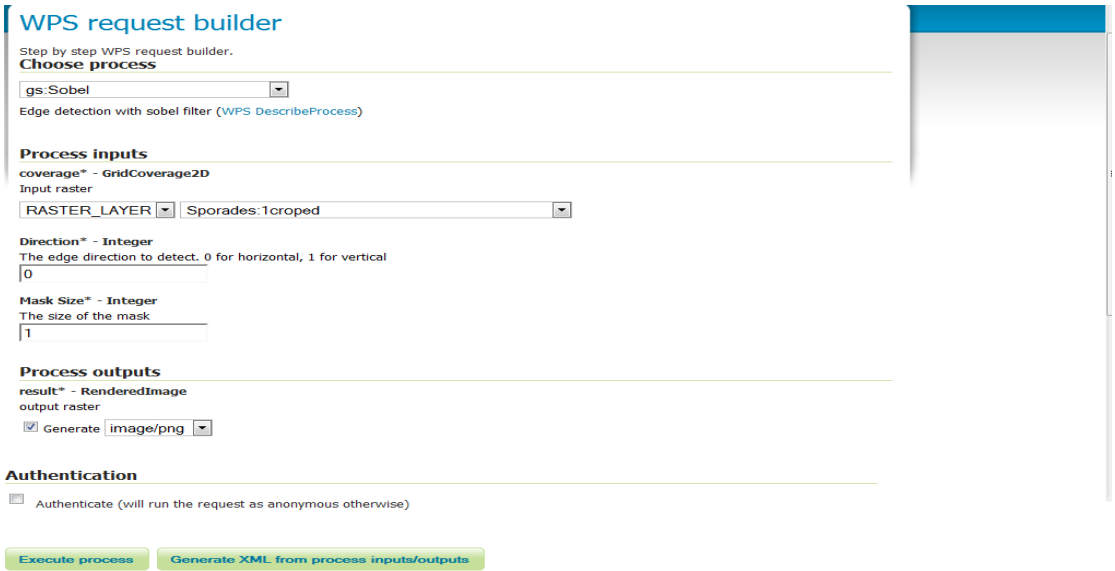
Εναλλακτικά, η εικόνα αυτή μπορεί να προκύψει σαν απάντηση του server στο WMS αίτημα :

http://localhost:8080/geoserver/Sporades/wms?service=WMS&version=1.1.0&request=GetMap&layers=Sporades:3croped&styles=&bbox=700572.75,4321497.75,771936.75,4370546.25&width=512&height=351&srs=EPSG:32634&format=application/openlayers

Στην καρτέλα Services φαίνονται οι υπηρεσίες του OGC που υλοποιούνται στον Server. Στην περίπτωση αυτή υλοποιούνται οι υπηρεσίες WMS, WFS, WCS και WPS. Κατά την επιλογή των αντίστοιχων πεδίων μπορούν να ρυθμιστούν οι παράμετροι που ορίζουν τις υπηρεσίες αυτές (π.χ. το μέγιστο μέγεθος της μνήμης που μπορεί να χρησιμοποιηθεί ή το URL στο οποίο ανταποκρίνεται η κάθε μία).

Η καρτέλα security ορίζει τις παραμέτρους ασφαλείας που εφαρμόζονται από τον Geoserver κατά την διαδικασία προβολής και επεξεργασίας των δεδομένων.

Η καρτέλα Demos περιέχει ενδεικτικά αιτήματα προς τον Geoserver προκειμένου να διευκολυνθεί η σύνταξη πιο πολύπλοκων αιτημάτων. Κατά την επιλογή αυτής της καρτέλας, εμφανίζονται οι Request builders για κάθε υπηρεσία. Αυτοί ενσωματώνουν ένα γραφικό περιβάλλον το οποίο κάνει το χειρισμό των υπηρεσιών πιο εύκολο ακόμα και χωρίς την ύπαρξη client. Μία ειδική περίπτωση είναι των Request Builders είναι και ο WPS Request Builder στον οποίο ενσωματώνονται όλες οι WPS επεξεργασίες. Ο WPS Builder ενσωματώνει ένα γραφικό περιβάλλον για το χειρισμό των κλάσεων DescribeProcess, DescribeParameter και DescribeResult (ακριβέστερα οι κλάσεις αυτές δημιουργούν τα αντίστοιχα πεδία στον Builder με τη βοήθεια του μηχανισμού αντιστροφής του ελέγχου του Spring Framework). Το περιβάλλον του Request builder για την περίπτωση του φίλτρου Sobel είναι το παρακάτω :



Εικόνα 4.3 : Request Builder για την περίπτωση του φίλτρου Sobel

Στην ενότητα Process inputs φαίνονται τα δεδομένα εισόδου της διαδικασίας. Τα αντίστοιχα πεδία δημιουργούνται από τις κλάσεις DescribeProcess, DescribeParameter και DescribeResult. Ένα θετικό στοιχείο είναι το γεγονός ότι στο πεδίο επιλογής της εικόνας φαίνονται με drop down list όλα τα ψηφιογραφικά layers που είναι αποθηκευμένα στον server. Τέλος, η διαδικασία παρέχει δύο επιλογές : την επιλογή Execute process και την επιλογή Generate XML from process inputs/outputs. Η πρώτη εντολή εκτελεί την διαδικασία με τις δεδομένες παραμέτρους (εικόνα, διεύθυνση εφαρμογής φίλτρου, μέγεθος μάσκας) και δίνει το αποτέλεσμα (τη φιλτραρισμένη εικόνα). Η δεύτερη επιλογή παράγει το αρχείο XML αρχείο το οποίο αποτελεί το σώμα του POST αιτήματος το οποίο πρέπει να υποβληθεί στον server από έναν client προκειμένου να ληφθεί το ίδιο αποτέλεσμα. Στο σημείο αυτό είναι που παρουσιάζεται το λειτουργικό πρόβλημα που αναφέρθηκε στο προηγούμενο κεφάλαιο. Όταν το συγκεκριμένο σώμα του POST αιτήματος υποβάλλεται στα πλαίσια κάποιου client (π.χ. OpenLayers, GeoNode, 52N) ο Geoserver δεν ανταποκρίνεται με ικανοποιητικό τρόπο και η απάντησή του δεν φτάνει στον client.

Το περιβάλλον όπως παρουσιάστηκε παραπάνω, είναι φιλικό προς το χρήστη και δεν απαιτεί ιδιαίτερες γνώσεις προγραμματισμού για να το χειριστεί κάποιος. Ένα μεγάλο πλεονέκτημα είναι η συγκέντρωση όλων των επιλογών δεδομένων(εισαγωγή, εξαγωγή, αλλαγή format) σε μία καρτέλα. Άλλο πλεονέκτημα είναι η γεωαναφορά όλων των θεματικών επιπέδων (raster και vector) και η δυνατότητα προβολής ετερόκλητων θεματικών επιπέδων (π.χ. ένα shapefile και ένα DEM) σε έναν ενιαίο χάρτη, του οποίου μάλιστα ο χρήστης επιλέγει το format.

Επίσης είναι θετικό πως στο ίδιο περιβάλλον υπάρχουν οι επεξεργασίες της υπηρεσίας WPS και δίνεται η δυνατότητα στο χρήστη να χρησιμοποιήσει θεματικά επίπεδα από αυτά που είναι αποθηκευμένα στον server ή να φορτώσει τα δικά του.

Τέλος όπως έχει αναφερθεί ο Geoserver είναι δυνατόν να ενσωματωθεί σε server Java (όπως ο Tomcat) και να είναι διαθέσιμος στο διαδίκτυο στη διεύθυνση του παρόχου. Αυτό είναι πολύ σημαντικό, καθώς ο χρήστης δεν χρειάζεται να εγκαταστήσει τίποτα στο σύστημά του. Χρειάζεται μόνο σύνδεση στο διαδίκτυο και ένα πρόγραμμα περιήγησης.

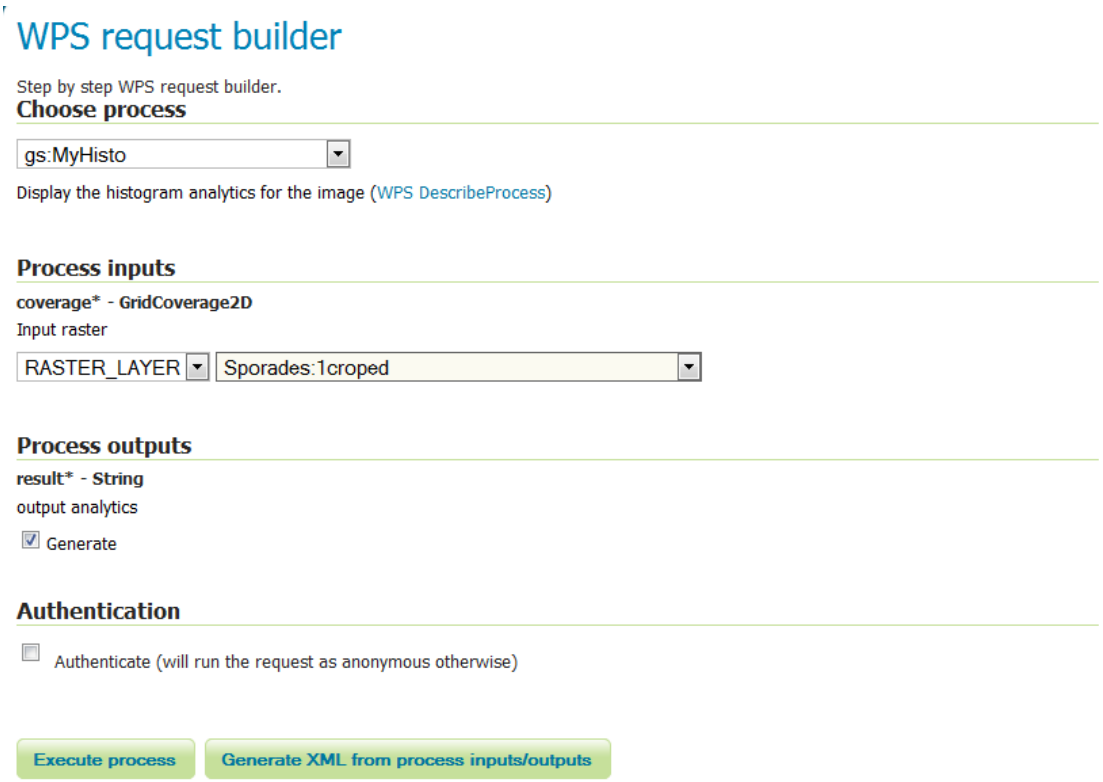
Στο σημείο αυτό ολοκληρώνεται η αξιολόγηση του περιβάλλοντος και ακολουθεί η αξιολόγηση των τηλεπισκοπικών μεθόδων που αναπτύχθηκαν, σαν επέκταση των δυνατοτήτων της υπηρεσίας WPS.

4.2 Αξιολόγηση μεθόδων τηλεπισκόπησης

Σε αυτήν την ενότητα γίνεται προσπάθεια να αξιολογηθούν οι μέθοδοι τηλεπισκόπησης που έχουν αναπτυχθεί σαν επέκταση του Geoserver και να αναδειχτεί η σημασία τους και η επίτευξη ή μη του σκοπού για τον οποίο χρησιμοποιούνται. Σαν παράδειγμα χρησιμοποιείται μία πολυφασματική εικόνα του Landsat 7 η οποία λήφθηκε την 24η Αυγούστου 2000 στην διαδρομή 183 και τη σειρά 33. Η αρχική εικόνα περιέχει το μεγαλύτερο μέρος της Στερεάς Ελλάδας, της Θεσσαλίας, της Εύβοιας, των Κυκλάδων και των Σποράδων. Για τις ανάγκες της παρούσας επίδειξης χρησιμοποιήθηκε ένα τμήμα της εικόνας των Βορείων Σποράδων.

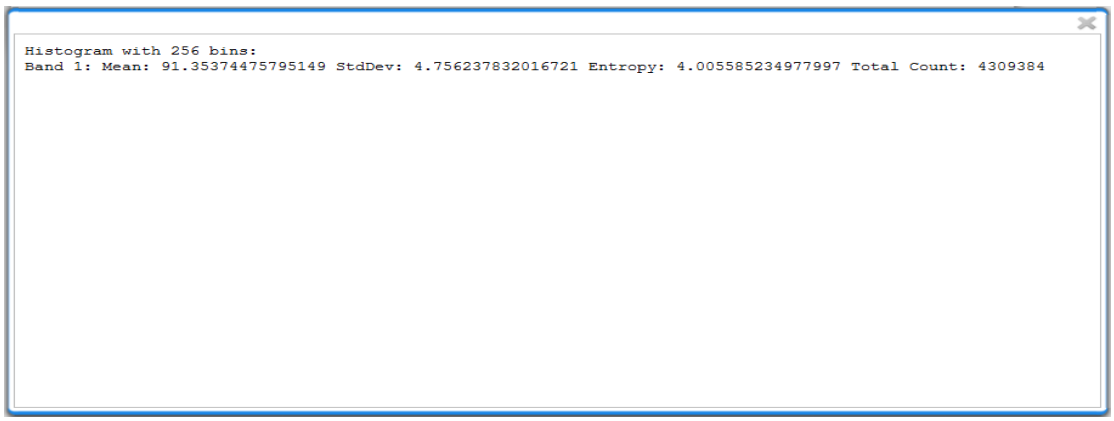
4.2.1 Ιστόγραμμα εικόνας

Όπως αναφέρθηκε το ιστόγραμμα εικόνας ενσωματώθηκε στην υπηρεσία WPS με τη βοήθεια του κόμβου ιστογράμματος της βιβλιοθήκης JAI. Το αποτέλεσμα της αλλαγής στο γραφικό περιβάλλον του Geoserver φαίνεται στην καρτέλα Demos όπου έχει προστεθεί στις διαδικασίες gs η διαδικασία με όνομα gs: MyHisto. Κατά την επιλογή αυτής της διαδικασίας παρουσιάζεται το παρακάτω γραφικό περιβάλλον :



Εικόνα 4.4 : Γραφικό περιβάλλον της διαδικασίας του ιστογράμματος

Για το κανάλι 1 της πολυφασματικής εικόνας που αναφέρθηκε (το θεματικό επίπεδο της οποίας ονομάζεται Sporades: 1croped στον Geoserver), το αποτέλεσμα της διαδικασίας είναι το παρακάτω :



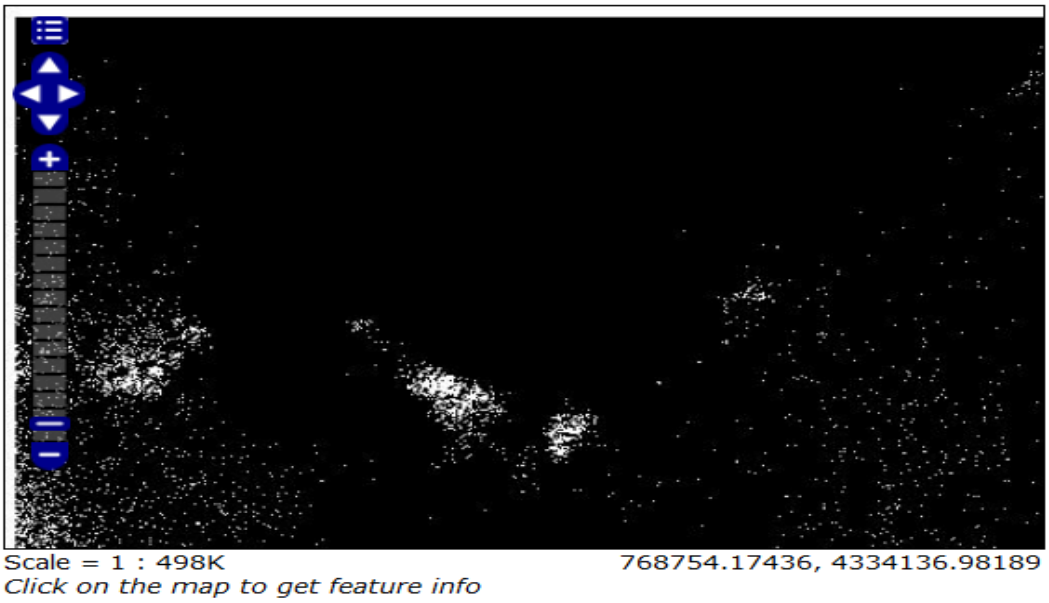
Εικόνα 4.5 : Αποτέλεσμα της διαδικασίας gs: MyHisto

Παρατηρούμε ότι η διαδικασία, αφού αναγνώρισε τον τύπο της εικόνας (εικόνα με ένα κανάλι και 256 χρώματα), υπολόγισε το μέσο όρο των τιμών χρώματος, την τυπική απόκλιση και την εντροπία του δείγματος. Επίσης υπολόγισε το πλήθος των pixel στην εικόνα.

Η διαδικασία αυτή μας δίνει ένα άμεσο μέτρο της κατανομής των τιμών χρώματος των pixel και δίνει με άμεσο τρόπο τις τιμές προς χρήση. Το μειονέκτημα της επεξεργασίας αυτής είναι η έλλειψη γραφικής παράστασης για τα συγκεκριμένα στοιχεία. Αυτό γίνεται , διότι η υπηρεσία WPS μπορεί να παρέχει μόνο ένα είδος αποτελέσματος κάθε φορά (εδώ είναι ακολουθία χαρακτήρων). Έτσι μεταξύ της εμφάνισης εικόνας της γραφικής παράστασης του ιστογράμματος και των αναλυτικών στοιχείων του επιλέχθηκε το δεύτερο, καθώς είναι πιο χρήσιμο για κάποιον που χρειάζεται να κάνει ανάλυση εικόνας. Έτσι η συμβολή αυτής της διαδικασίας στην επέκταση του Geoserver αποτιμάται θετικά καθώς συμβάλλει στην χρησιμότητά του και προς κατευθύνσεις οι οποίες δεν υποστηρίζονταν μέχρι τώρα.

4.2.2 Κατώφλι

Η διαδικασία της κατωφλίωσης είναι η πιο απλή περίπτωση κατάτμησης εικόνας. Χρησιμοποιείται σε πληθώρα εφαρμογών και συνήθως είναι το πρώτο βήμα που πραγματοποιείται κατά τη διαδικασία της ανάλυσης κάποιας εικόνας. Όπως αναφέρθηκε στα προηγούμενα, η συγκεκριμένη μέθοδος αναπτύχθηκε στην υπηρεσία WMS και κάνει χρήση της γλώσσας SLD. Παρακάτω φαίνεται η κατωφλιωμένη εικόνα των Σποράδων από το κανάλι 1 του Landsat για τιμή κατωφλιού 84.5 :



Εικόνα 4.6 : Εφαρμογή κατωφλιού

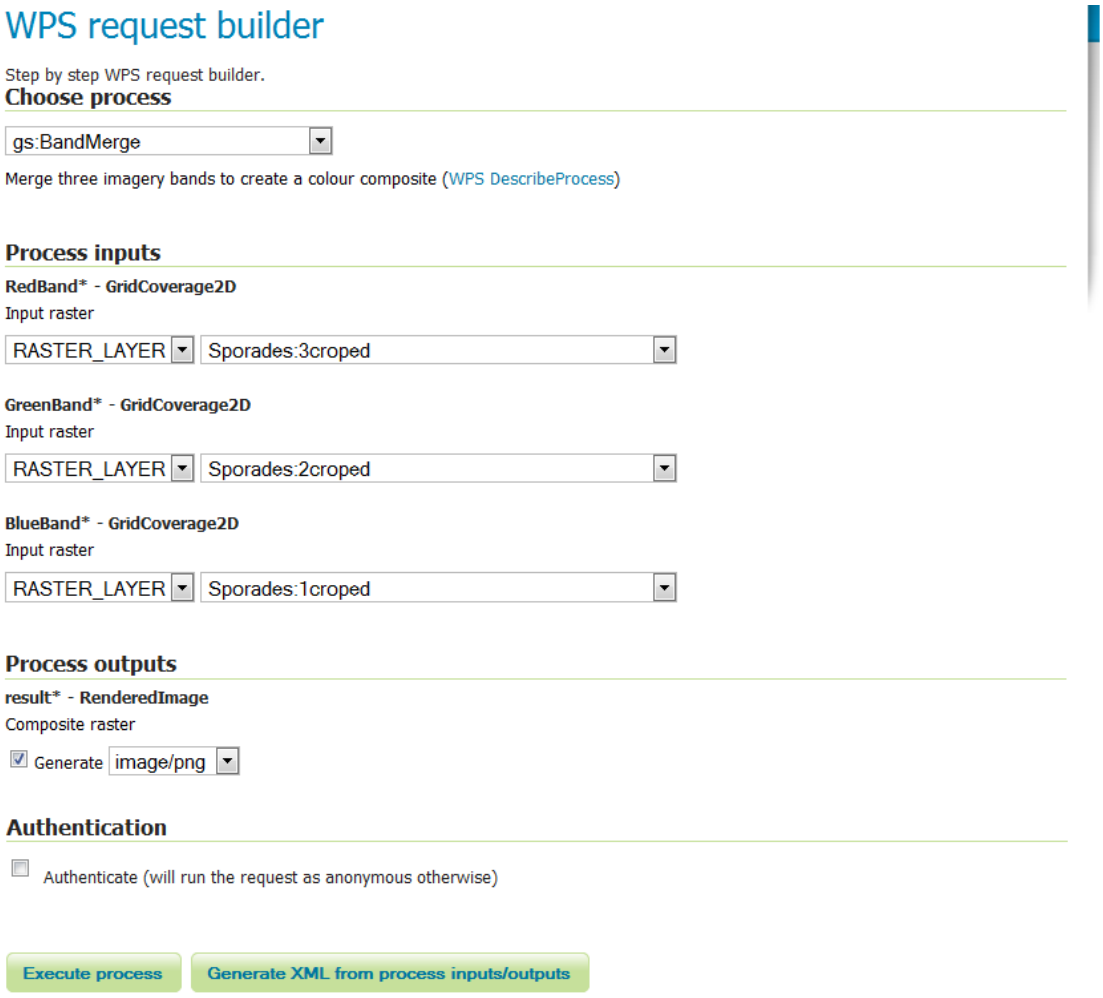
Η συγκεκριμένη επεξεργασία δεν απαιτεί κάποια επέκταση καθώς απλώς αλλάζει τον τρόπο προβολής του συγκεκριμένου ψηφιογραφικού θεματικού επιπέδου μέσω του αρχείου SLD . Το πλεονέκτημα της μεθόδου είναι ότι απαιτεί πολύ μικρή υπολογιστική ισχύ για να τρέξει. Επίσης το αρχείο SLD που συντάσσεται μπορεί να χρησιμοποιηθεί σαν στυλ για όλα τα ψηφιογραφικά θεματικά επίπεδα που είναι αποθηκευμένα στο server. Ένα μειονέκτημα της μεθόδου είναι η μειωμένη αυτοματοποίηση. Για να οριστεί καινούργια τιμή κατωφλιού, θα πρέπει να συνταχθεί εκ νέου άλλο αρχείο SLD και να επαναπροσδιοριστούν οι παράμετροι προβολής. Επίσης η σύνταξη αυτή πρέπει να γίνει από γνώστη της συγκεκριμένης γλώσσας. Τέλος η συγκεκριμένη μέθοδος έχει το πλεονέκτημα ότι τα δεδομένα εξόδου μπορούν να παραδοθούν στο χρήστη σε διάφορα format π.χ. GeoTiff, PNG, JPEG, GIF και άλλα.

Το αποτέλεσμα με βάση όλα τα παραπάνω κρίνεται θετικό, εφόσον προσδίδει στον Geoserver άλλη μία μέθοδο επεξεργασίας εικόνας και συμβάλλει έτσι στην μετατροπή του σε server επεξεργασίας εικόνας.

4.2.3 Έγχρωμα σύνθετα

Τα έγχρωμα σύνθετα, όπως αναλύθηκε στα προηγούμενα, δημιουργούνται με εφαρμογή του κόμβου «bandmerge» της βιβλιοθήκης JAI. Η κλάση ενσωματώνεται στον Geoserver και την υπηρεσία WPS. Τα εξαγόμενα της διαδικασίας είναι μία εικόνα στην οποία τα συγχωνεύονται τρία κανάλια. Το γραφικό περιβάλλον της υπηρεσίας μεταβάλλεται για μπορέσει να υποστηρίξει την μέθοδο. Από τη λίστα των επεξεργασιών του server επιλέγουμε τη

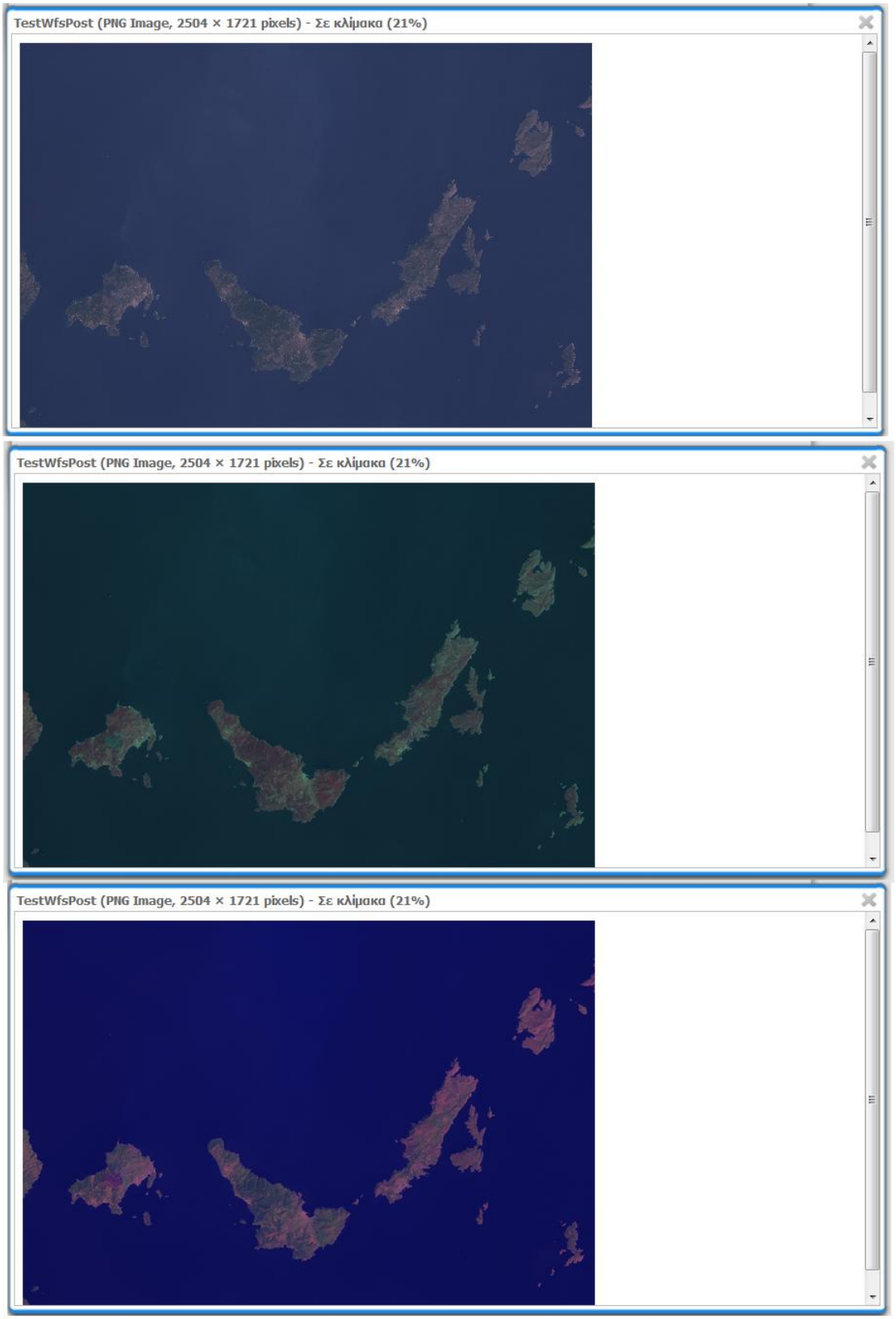
διαδικασία gs : BandMerge. Κατά την επιλογή αυτή, εμφανίζεται το παρακάτω γραφικό περιβάλλον :



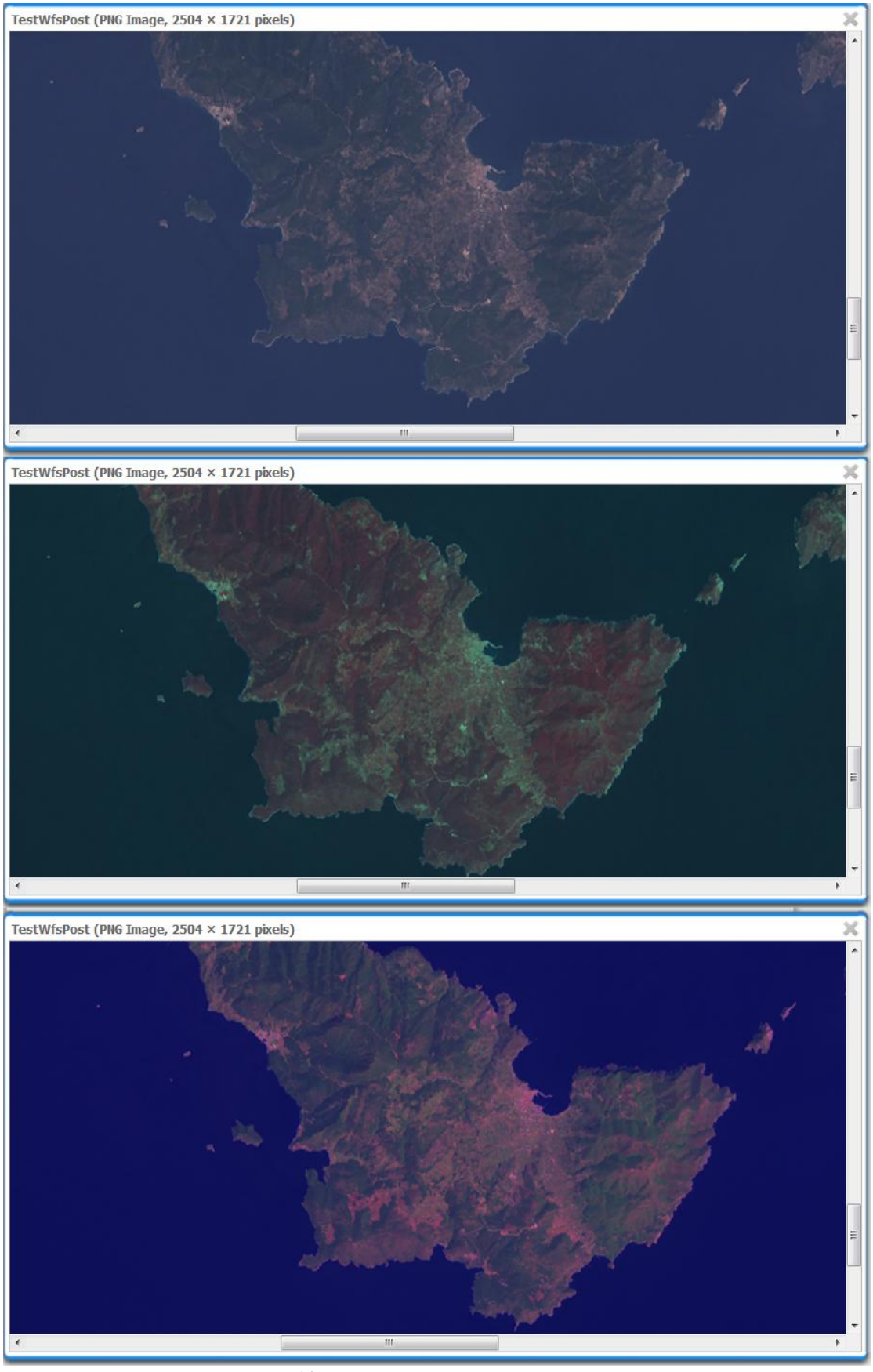
Εικόνα 4.7 : Γραφικό περιβάλλον της υπηρεσίας BandMerge

Παρακάτω φαίνονται τρία έγχρωμα σύνθετα ου δημιουργήθηκα με αυτή τη διαδικασία. Πρόκειται για τα τρία έγχρωμα σύνθετα 3-2-1(φυσικό), 4-3-2(υπέρυθρο), 5-4-1(εγγύς υπέρυθρο). Με βάση αυτά θα εξεταστεί κατά πόσο η διαδικασία αυτή παρουσιάζει τα χαρακτηριστικά που έχουν τα έγχρωμα σύνθετα και κατά πόσο εκπληρώνει το σκοπό της. Για να γίνουν αυτά παρουσιάζεται η αρχική εικόνα και έπειτα παρουσιάζεται η εικόνα εστιασμένη στον νησί της Αλοννήσου. Για το σκοπό της παρουσίασης επιλέχθηκε το τμήμα που βρίσκεται στο νότιο τμήμα του νησιού. Αυτό έγινε διότι το συγκεκριμένο τμήμα περιέχει περιοχές διάφορων κατηγοριών : αστικές περιοχές (το λιμάνι και τη Χώρα), δασικές εκτάσεις, θαμνώδεις εκτάσεις και το θαλάσσιο πάρκο. Με αυτόν τον τρόπο είναι εύκολο να εξαχθούν

συμπεράσματα για κάθε έγχρωμο σύνθετο και να αναλυθεί σε ποιο βαθμό η διαδικασία επιτυγχάνει το σκοπό της.



Εικόνα 4.8 Τα έγχρωμα σύνθετα 3-2-1, 4-3-2, 5-4-1



Εικόνα 4.9 : Το νότιο τμήμα της Αλοννήσου

Από τις παραπάνω εικόνες, παρατηρούμε ότι τα χαρακτηριστικά κάθε έγχρωμου σύνθετου είναι πιστά.

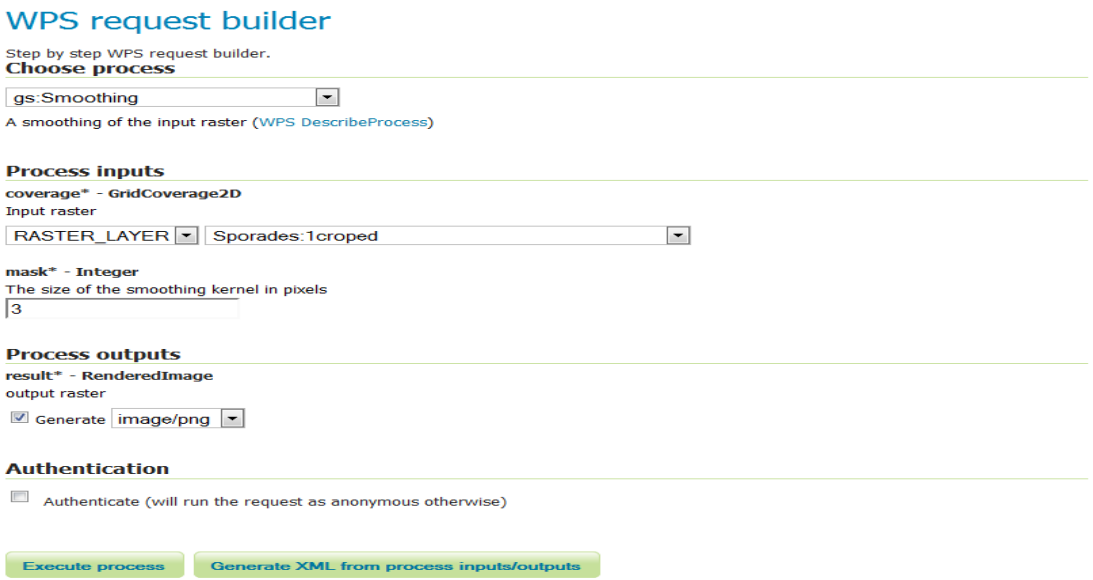
Το έγχρωμο σύνθετο 3-2-1 εμφανίζει τα αντικείμενα με το φυσικό τους χρώμα. Το έγχρωμο σύνθετο 4-3-2 εμφανίζει τη βλάστηση με κόκκινο χρώμα και υποδεικνύει τις αστικές περιοχές. Το έγχρωμο σύνθετο 5-4-1 εμφανίζει σε μεγάλο βαθμό τη βλάστηση και μάλιστα με πράσινο χρώμα. Επομένως τα έγχρωμα σύνθετα αυτά, ανταποκρίνονται με πιστότητα στο σκοπό που έχουν.

4.2.4 Φίλτρα – Ανίχνευση ακμών

Οι διαδικασίες εφαρμογής φίλτρων και συνέλιξης παίρνουν σαν όρισμα συνήθως μία εικόνα η οποία αποτελεί θεματικό επίπεδο του server και κάποιες άλλες παραμέτρους (μέγεθος μάσκας και κατεύθυνση φιλτραρίσματος). Τα γραφικά περιβάλλοντα κάθε διαδικασίας προσαρμόζονται ανάλογα με τα δεδομένα εισόδου. Τα δεδομένα εξόδου είναι η εικόνα μετά το πέρας της διαδικασίας συνέλιξης. Παρακάτω παρουσιάζονται αναλυτικά όλες οι διαδικασίες :

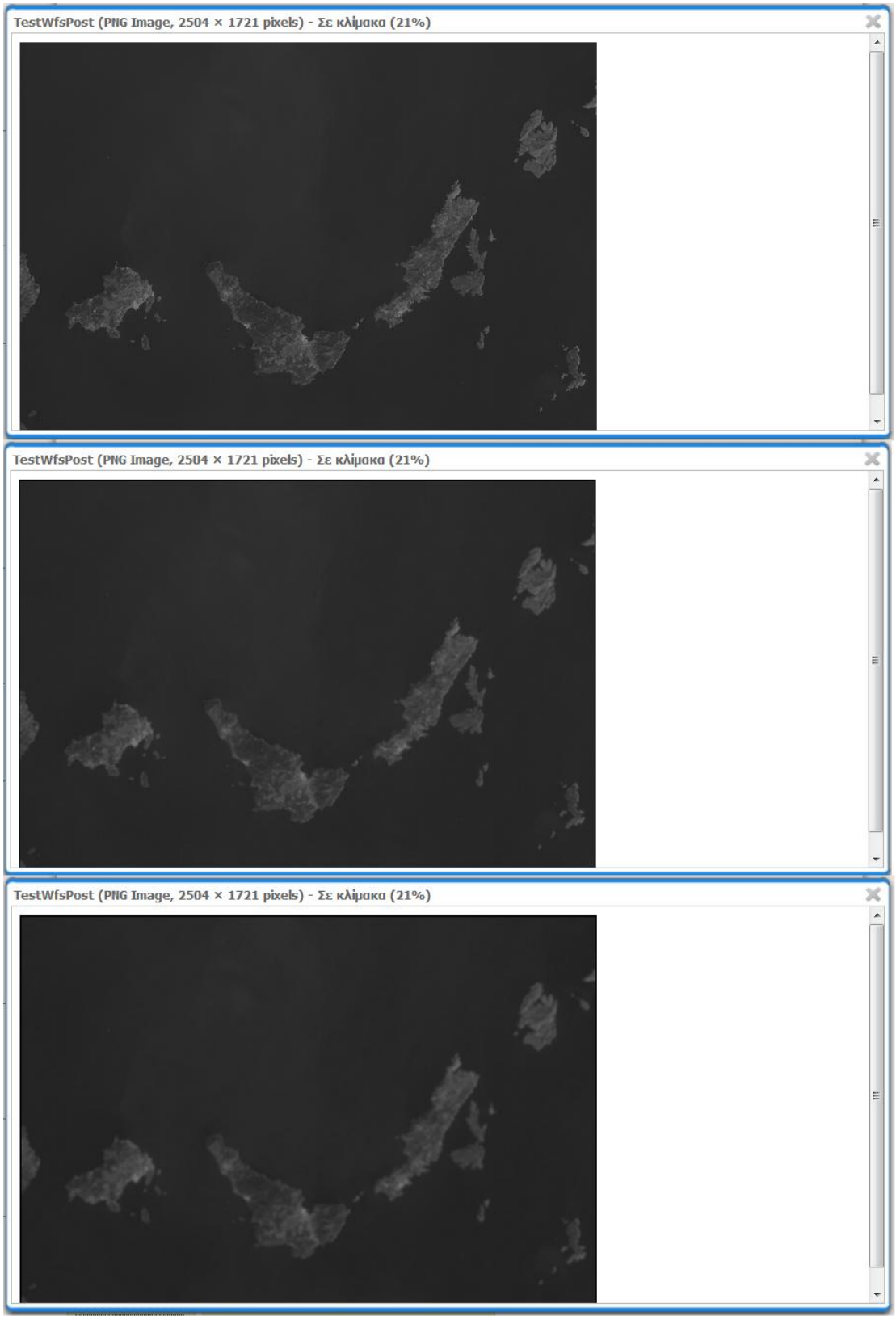
- Φίλτρο μέσης τιμής

Το φίλτρο εξομάλυνσης παίρνει σαν όρισμα το θεματικό επίπεδο της εικόνας και το μέγεθος της μάσκας. Πραγματοποιεί τη συνέλιξη του πίνακα της εικόνας και της μάσκας και δίνει την εικόνα από την οποία έχει αφαιρεθεί ο θόρυβος. Η λειτουργία είναι διαθέσιμη από την drop down list του request builder, με το όνομα gs: Smoothing. Παρακάτω φαίνεται το γραφικό περιβάλλον της λειτουργίας :



Εικόνα 4.10 : Το γραφικό περιβάλλον της λειτουργίας εξομάλυνσης

Παρακάτω φαίνονται τα αποτελέσματα της διαδικασίας για μέγεθος μάσκας 10x10 και 20x20 αντίστοιχα. Το παράδειγμα αφορά εικόνα του Landsat από την περιοχή των βόρειων Σποράδων :



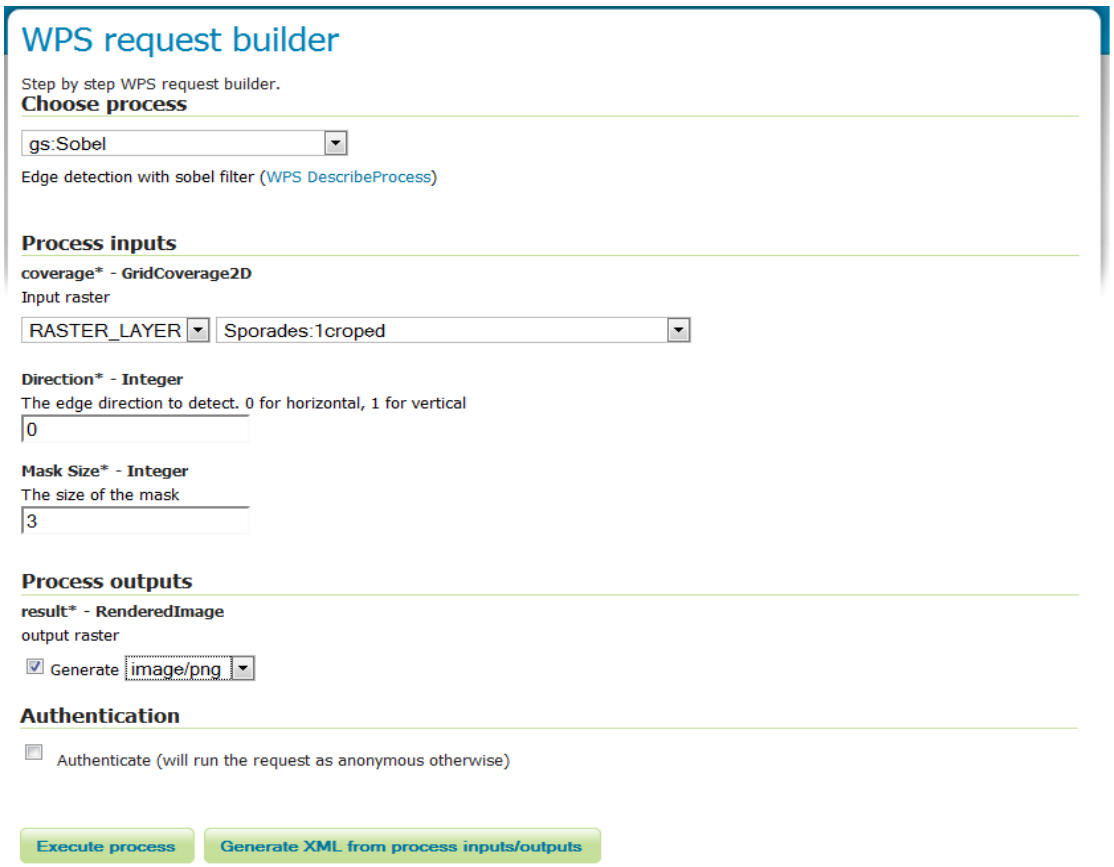
Εικόνα 4.11 : Αρχική εικόνα και φίλτρο μέσης τιμής με μέγεθος μάσκας 10x10 και 20x20

Από την παράθεση των εικόνων αυτών συμπεραίνουμε ότι στην εικόνα προστίθεται αρκετή θόλωση. Επίσης παρατηρείται καλή επίδοση στην απαλοιφή του γραμμικού θορύβου (στην αρχική εικόνα φαίνεται παράλληλος με την οριζόντια διάσταση της εικόνας). Τέλος παρατηρείται αδυναμία απαλοιφής του τυχαίου θορύβου (φαίνεται σαν κηλίδα στα βόρεια της εικόνας).

Επομένως, συνάγεται το συμπέρασμα ότι το φίλτρο μέσης τιμής που αναπτύχθηκε μπορεί να ανταποκριθεί με επιτυχία στο σκοπό του.

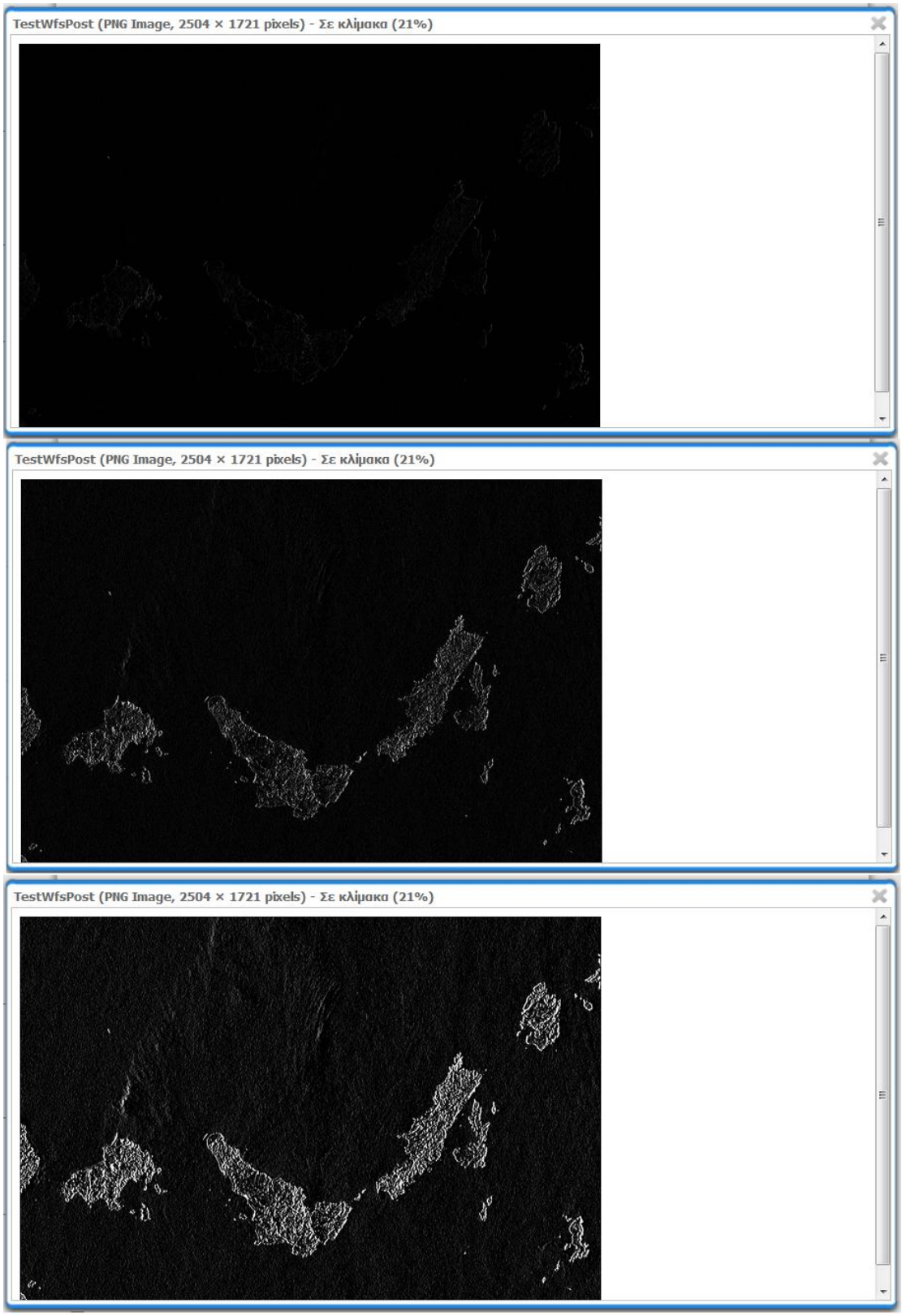
- Φίλτρο Sobel

Τα φίλτρα Sobel όπως περιγράφηκε στα προηγούμενα, μπορούν να χρησιμοποιηθούν για πολλές περιπτώσεις μάσκας. Στη συγκεκριμένη υλοποίηση, είναι δυνατό να παραχθεί η εικόνα με διαφορετικό μέγεθος μάσκας και με διαφορετική κατεύθυνση υπολογισμού των ακμών. Επίσης είναι δυνατό να υπολογισθεί το φίλτρο πρώτης παραγώγου με μάσκα Sobel (μόνο για μάσκα 3x3). Η διαδικασία βρίσκεται στην drop down list της υπηρεσίας WPS, υπό το όνομα gs : Sobel. Το γραφικό περιβάλλον του request builder μεταβάλλεται:



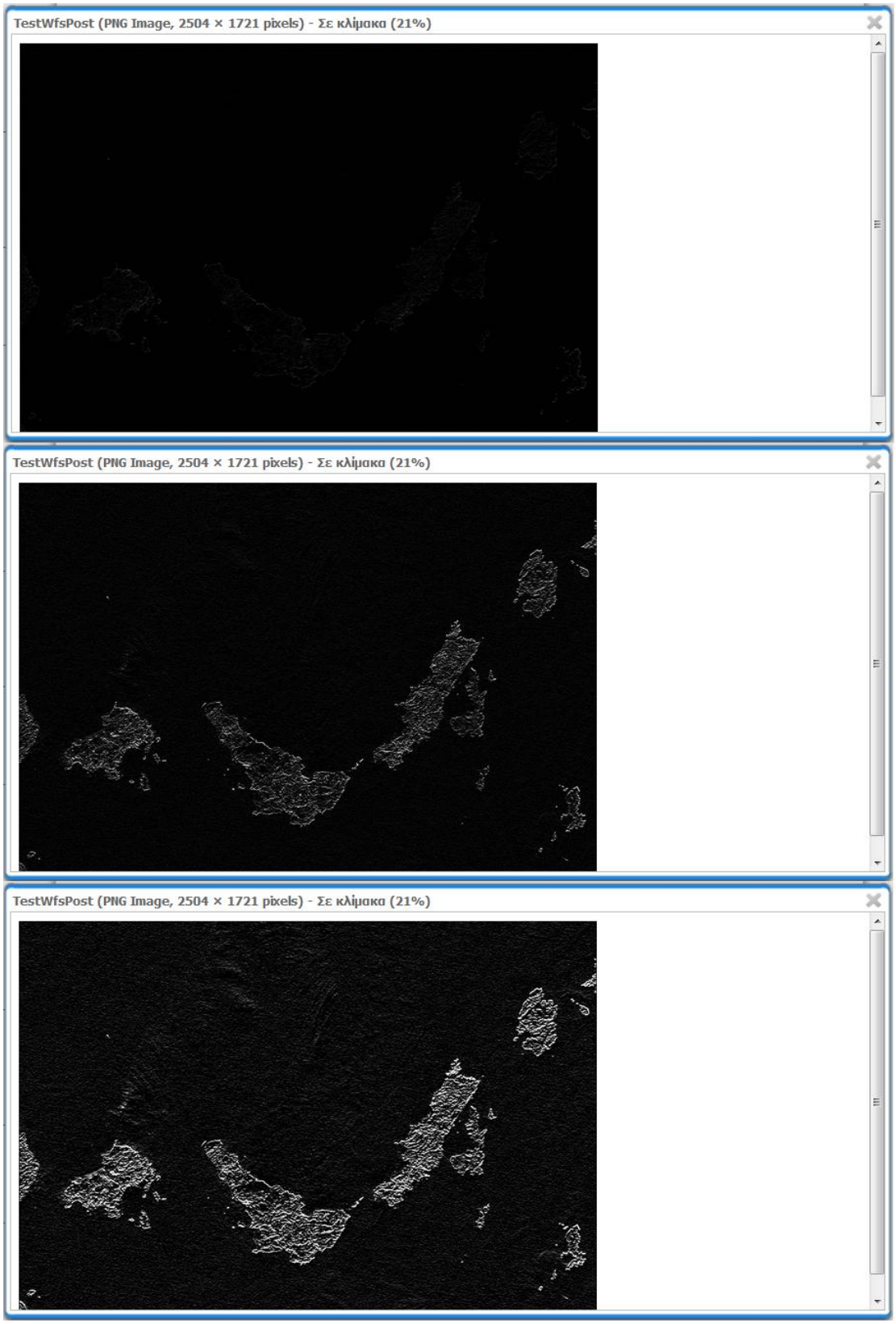
Εικόνα 4.12 : Το γραφικό περιβάλλον της υπηρεσίας Sobel

Παρακάτω εφαρμόζεται η διαδικασία κατά την οριζόντια διεύθυνση για διάφορα μεγέθη μάσκας. Η εικόνα αφορά το κανάλι 1 του Landsat από την περιοχή των βόρειων Σποράδων :



Εικόνα 4.13 : Εφαρμογή φίλτρου Sobel κατά την οριζόντια διεύθυνση για μέγεθος μάσκας 3x3, 5x5, 7x7 αντίστοιχα

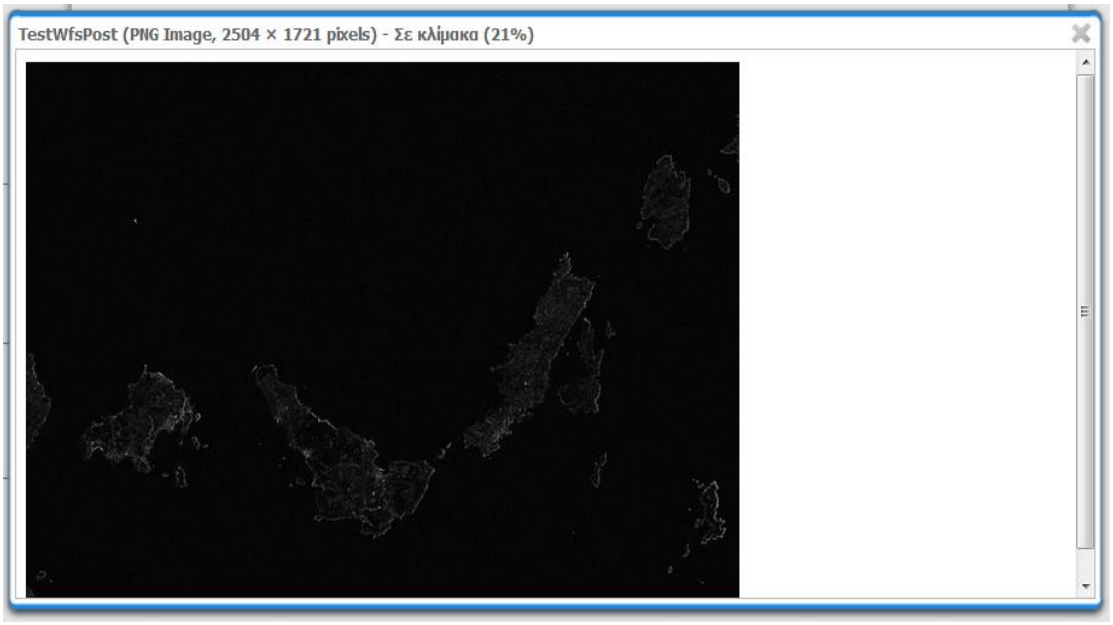
Παρακάτω εφαρμόζεται η διαδικασία κατά την κατακόρυφη διεύθυνση για διάφορα μεγέθη μάσκας. Η εικόνα αφορά το κανάλι 1 του Landsat από την περιοχή των βόρειων Σποράδων :



Εικόνα 4.14 : : Εφαρμογή φίλτρου Sobel κατά την κατακόρυφη διεύθυνση για μέγεθος μάσκας 3x3, 5x5, 7x7 αντίστοιχα

Από τις εικόνες που παρατέθηκαν, είναι εμφανής η διαφορά στην ανίχνευση της διεύθυνσης των ακμών. Επίσης, είναι εμφανής η ευαισθησία στο θόρυβο κατά τη μεγέθυνση της μάσκας. Τέλος, είναι εμφανής η ανίχνευση των αντίστοιχων συνιστωσών του θορύβου στο βόρειο μέρος της εικόνας.

Το φίλτρο πρώτης παραγώγου με τη μάσκα Sobel ενδείκνυται για την επίλυση των παρακάτω προβλημάτων. Παρακάτω φαίνεται η υλοποίησή του για την ίδια εικόνα :



Εικόνα 4.15 : Φίλτρο πρώτης παραγώγου με μάσκα Sobel

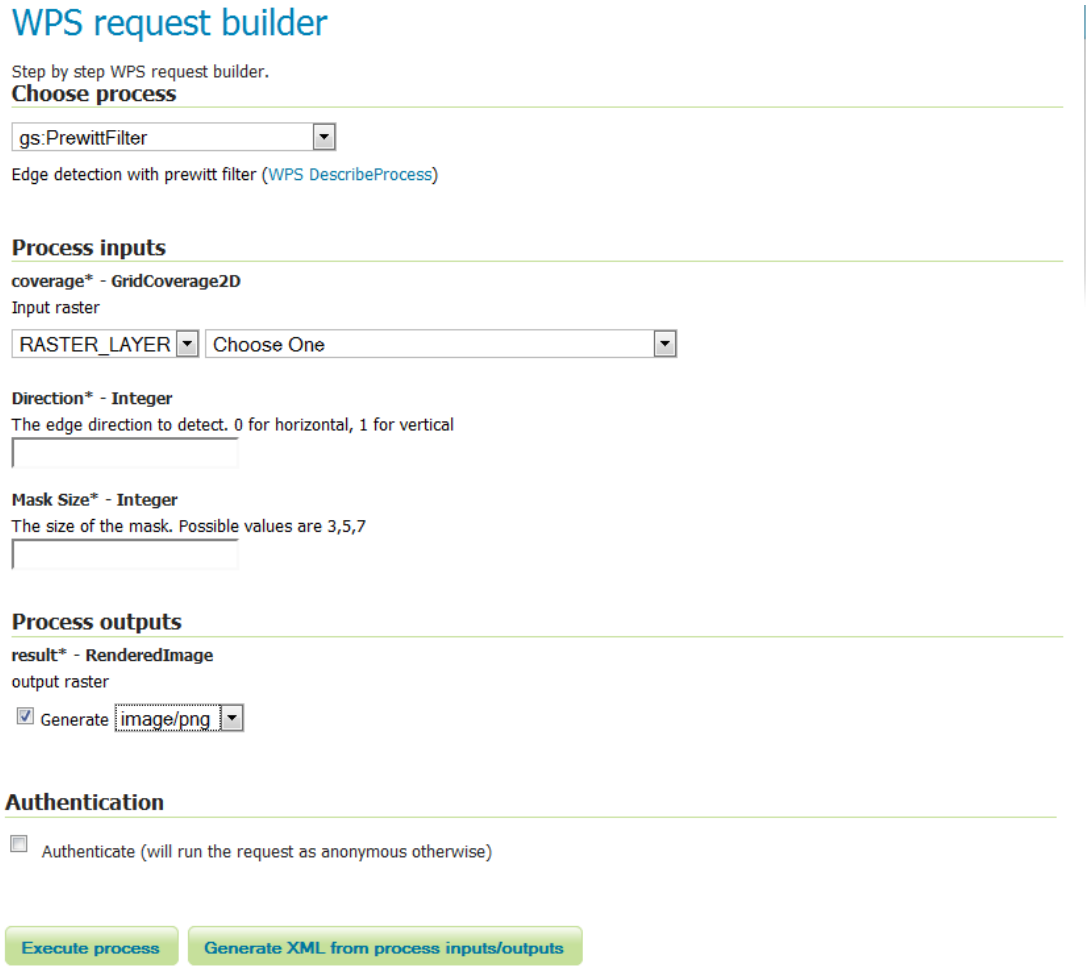
Στην παραπάνω εικόνα βλέπουμε όλα τα χαρακτηριστικά που αναμένονταν κατά την εφαρμογή φίλτρου πρώτης παραγώγου. Αρχικά είναι εμφανής η ανίχνευση των ακμών προς όλες τις κατευθύνσεις. Αυτό γίνεται διότι υπολογίζεται το μέτρο του διανύσματος της εφαπτομένης των ακμών με αυτήν τη μέθοδο. Επομένως το φίλτρο αυτό είναι ισοτροπικό.

Επίσης, είναι εμφανής η απαλοιφή του θορύβου στο βόρειο τμήμα της εικόνας. Αυτό γίνεται διότι το διανυσματικό άθροισμα των συνιστωσών του θορύβου (που ήταν ορατός στην προηγούμενη περίπτωση) έχει μικρό μέτρο και έτσι λαμβάνει μικρότερη τιμή χρώματος.

Τέλος από τα παραπάνω συμπεραίνουμε ότι η διαδικασία του φίλτρου Sobel, όπως αναπτύχθηκε είναι μία πιστή εφαρμογή της διαδικασίας και τα αποτελέσματά της ανταποκρίνονται σε αυτά που αναμένονταν. Επομένως αποτιμάται θετικά η ένταξή της στις επεξεργασίες WPS και είναι ουσιαστική μία επέκταση του geoserver προς αυτήν την κατεύθυνση.

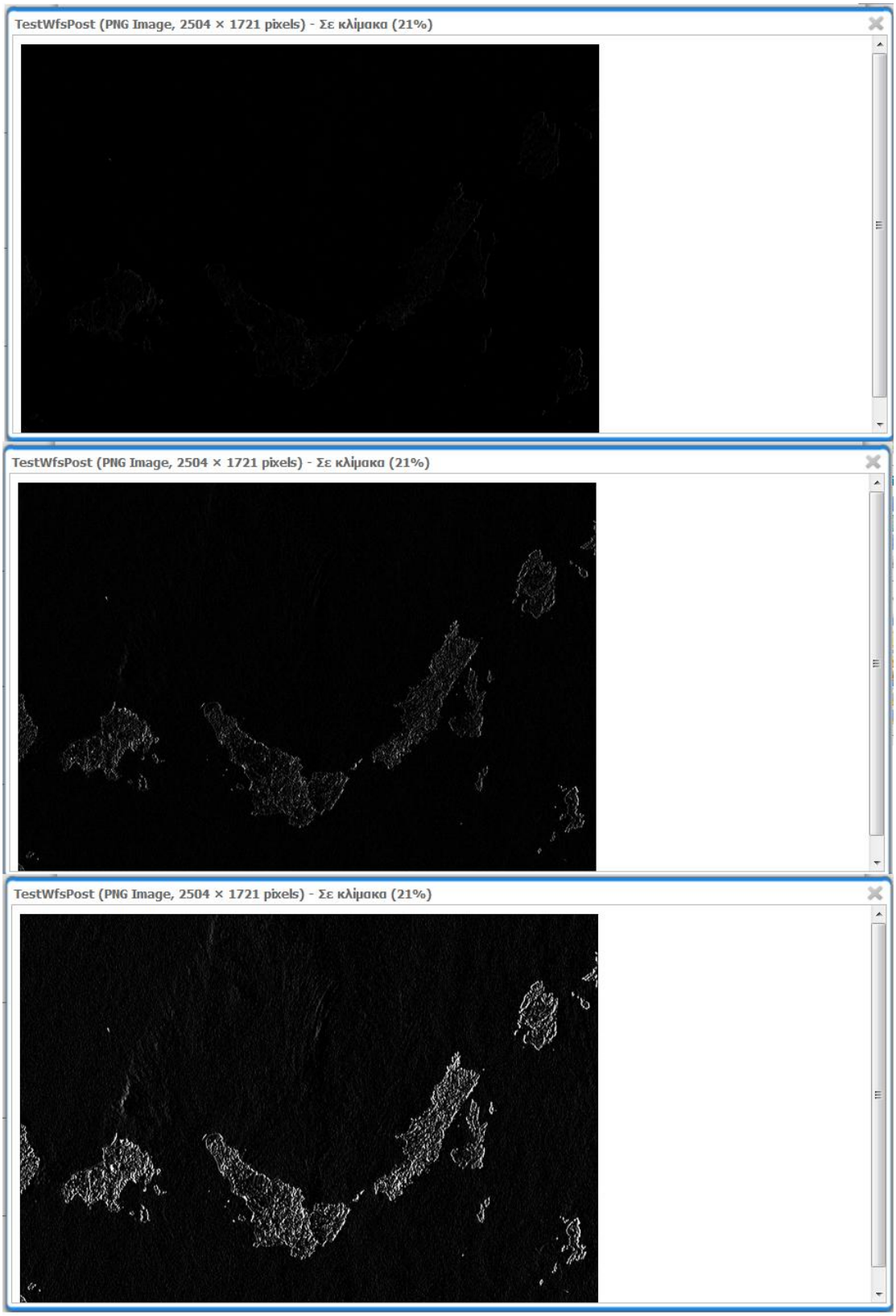
- Φίλτρο Prewitt

Το φίλτρο Prewitt είναι εναλλακτικό του Sobel, με τη διαφορά ότι δίνει μικρότερο βάρος στα γειτονικά pixel του κέντρου της μάσκας. Η διαδικασία που υλοποιήθηκε μπορεί να βρεθεί στον WPS request builder υπό την ονομασία gs : PrewittFilter. Η διαδικασία παίρνει σαν ορίσματα το θεματικό επίπεδο της εικόνας, το μέγεθος της μάσκας και την κατεύθυνση για την οποία ανιχνεύονται οι ακμές. Η γραφική διεπαφή της διαδικασίας είναι η παρακάτω:



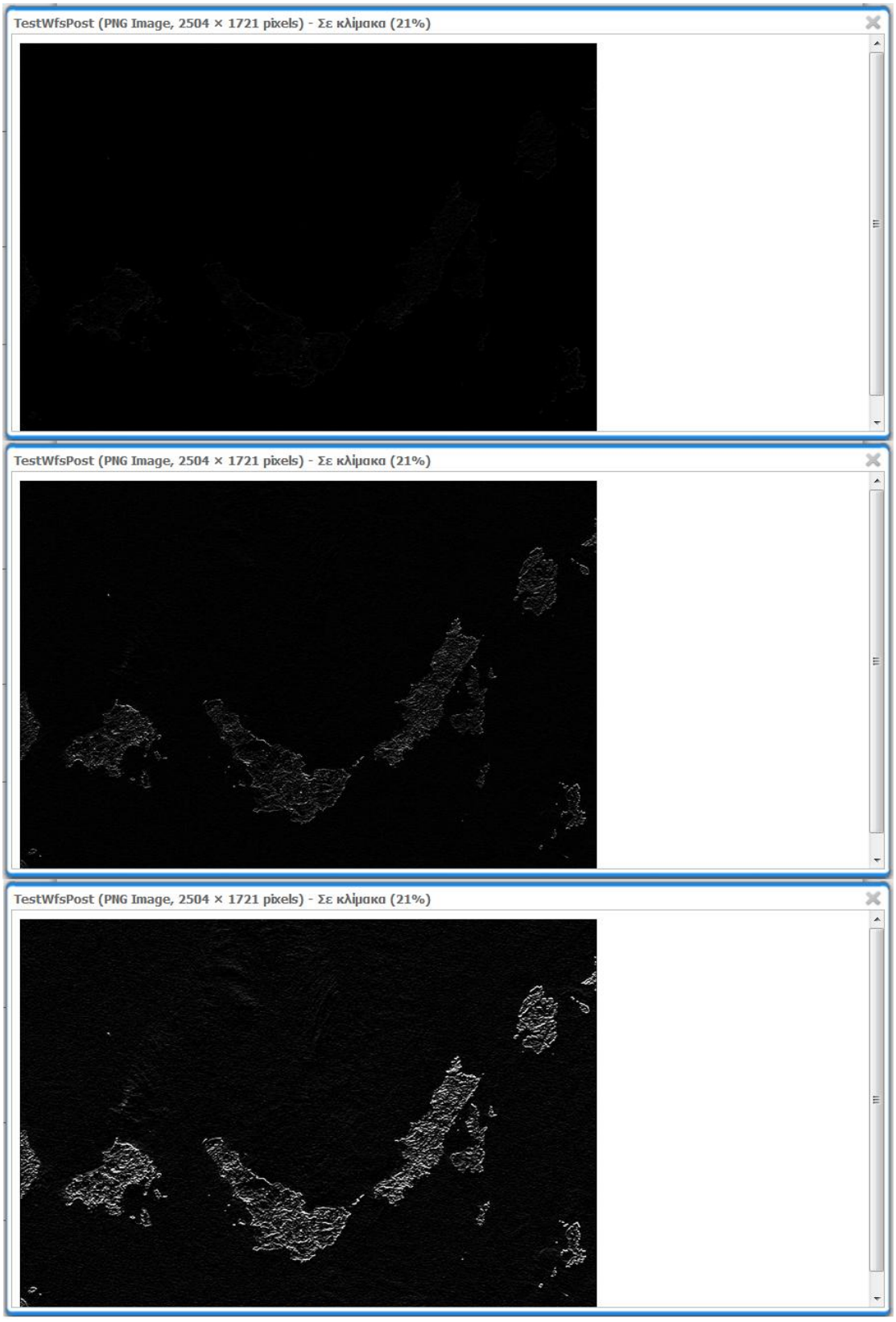
Εικόνα 4.16 : Γραφική διεπαφή της διαδικασίας φίλτρου Prewitt

Το αποτέλεσμα της διαδικασίας συνέλιξης είναι η εικόνα για την οποία έχουν υπολογιστεί οι ακμές. Το αποτέλεσμα της διαδικασίας για το κανάλι 1 του Landsat από την περιοχή των βόρειων Σποράδων φαίνεται παρακάτω. Δίνονται αρχικά οι εικόνες για φίλτρα κατά την οριζόντια διεύθυνση και ύστερα για φίλτρα κατά την κατακόρυφη διεύθυνση :



Εικόνα 4.17 : Φίλτρο Prewitt κατά την οριζόντια διεύθυνση με μάσκες 3x3, 5x5, 7x7

Κατά την κατακόρυφη διεύθυνση τα ίδια φίλτρα δίνουν :

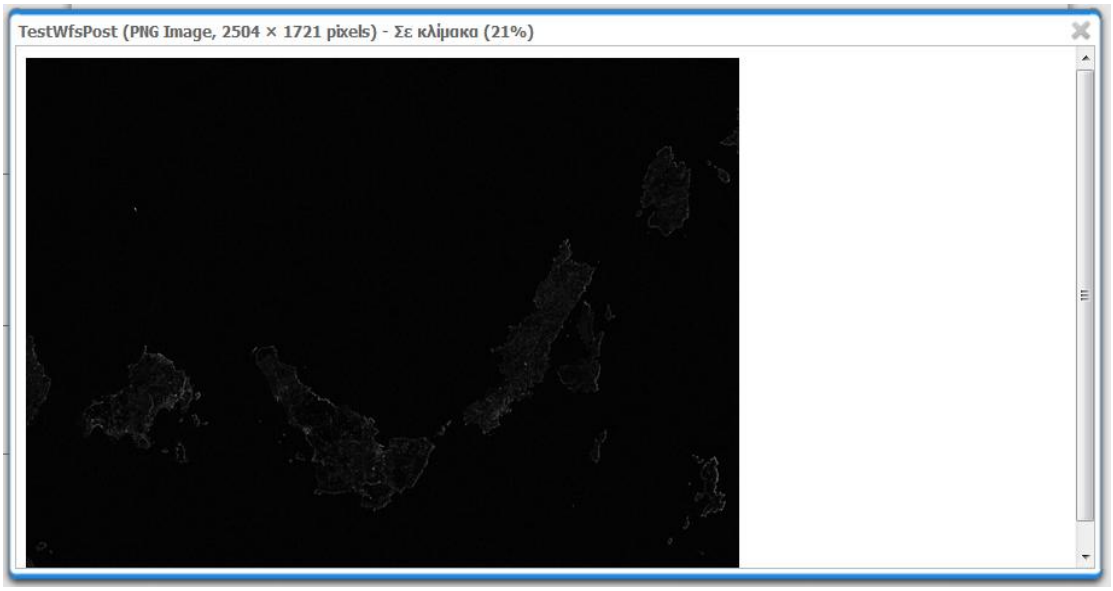


Εικόνα 4.18 : Φίλτρο Prewitt κατά την κατακόρυφη διεύθυνση με μάσκες 3x3, 5x5, 7x7

Στις παραπάνω εικόνες παρατηρείται εμφανώς η ανίχνευση της διεύθυνσης των ακμών. Επίσης παρατηρείται η αύξηση στην ευαισθησία στο θόρυβο κατά την αύξηση του μεγέθους της μάσκας. Τέλος, είναι εμφανής η ανίχνευση των αντίστοιχων συνιστωσών του θορύβου στο βόρειο μέρος της εικόνας.

Σε σύγκριση με το φίλτρο Sobel, παρατηρείται μία μείωση στο πάχος των ακμών. Επίσης, παρατηρείται ένα κενό στην ανίχνευση εσωτερικών ακμών (π.χ. εσωτερικό των νησιών) και μικρότερου μεγέθους ακμών.

Τα παραπάνω προβλήματα αναμένεται να αντιμετωπιστούν με την εφαρμογή του φίλτρου πρώτης παραγώγου με μάσκα Prewitt. Το φίλτρο αυτό φαίνεται στην επόμενη εικόνα :



Εικόνα 4.19 : Φίλτρο Prewitt πρώτης παραγώγου

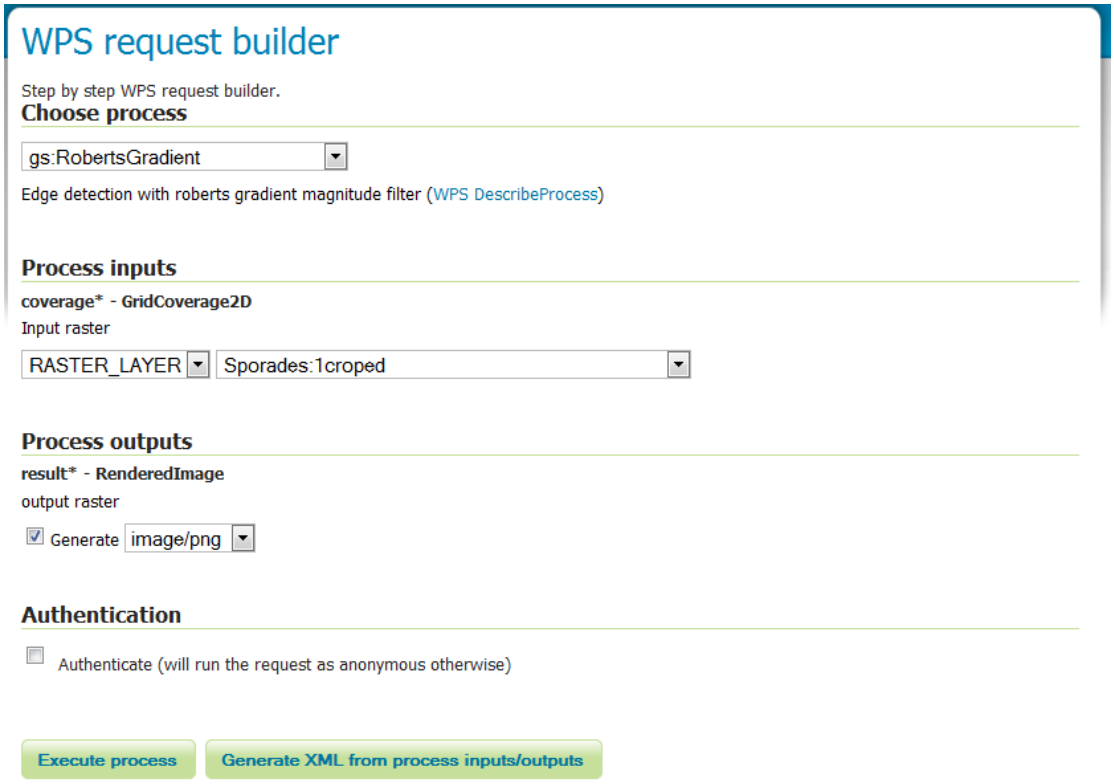
Στην παραπάνω εικόνα βλέπουμε όλα τα χαρακτηριστικά που αναμένονταν κατά την εφαρμογή φίλτρου πρώτης παραγώγου. Αρχικά είναι εμφανής η ανίχνευση των ακμών προς όλες τις κατευθύνσεις. Αυτό γίνεται διότι υπολογίζεται το μέτρο του διανύσματος της εφαπτομένης των ακμών με αυτήν τη μέθοδο. Επομένως το φίλτρο αυτό είναι ισοτροπικό.

Επίσης, είναι εμφανής η απαλοιφή του θορύβου στο βόρειο τμήμα της εικόνας. Αυτό γίνεται διότι το διανυσματικό άθροισμα των συνιστωσών του θορύβου (που ήταν ορατός στην προηγούμενη περίπτωση) έχει μικρό μέτρο και έτσι λαμβάνει μικρότερη τιμή χρώματος.

Τέλος, η υλοποιημένη διαδικασία σύμφωνα με τα παραπάνω, ανταποκρίνεται στα αναμενόμενα αποτελέσματα και ικανοποιεί το σκοπό για τον οποίο υλοποιείται. Επομένως η διαδικασία επέκτασης του geoserver με αυτήν αποτιμάται θετικά.

- Φίλτρο Roberts

Το φίλτρο Roberts είναι εναλλακτικό των δύο προηγούμενων, με τη διαφορά ότι χρησιμοποιεί μάσκα 2x2 . Το φίλτρο Roberts, είναι ευαίσθητο στο θόρυβο και λόγω του μικρού μεγέθους του είναι βέλτιστο για ασπρόμαυρες εικόνες. Για αυτούς τους λόγους, υλοποιήθηκε στη συγκεκριμένη εφαρμογή μόνο το φίλτρο πρώτης παραγώγου. Η διαδικασία είναι διαθέσιμη στον WPS request builder υπό την ονομασία gs: RobertsGradient. Η διαδικασία δέχεται σαν μοναδικό όρισμα το θεματικό επίπεδο της εικόνας και δίνει σαν αποτέλεσμα την υπολογισμένη εικόνα. Παρακάτω φαίνεται το γραφικό περιβάλλον της διαδικασίας :



Εικόνα 4.20 : Γραφικό περιβάλλον της διαδικασίας RobertsGradient

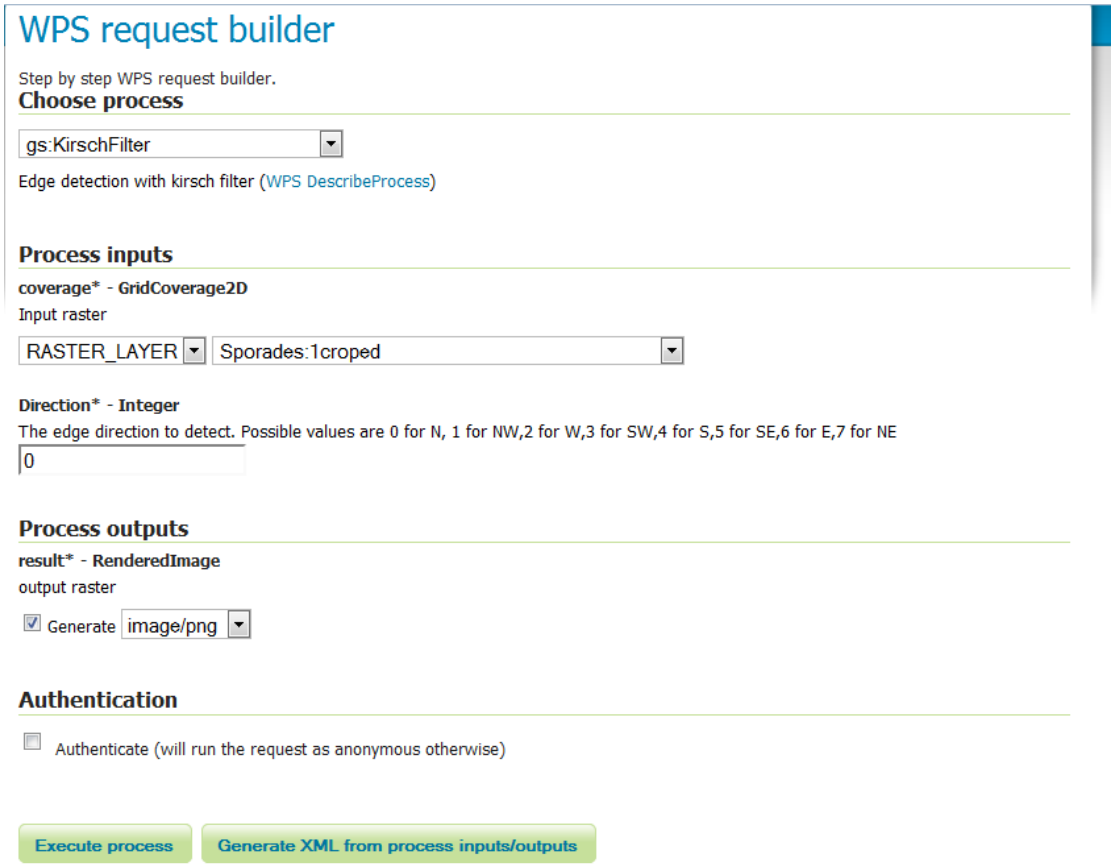
Η εκτέλεση της διαδικασίας για την εικόνα των προηγούμενων παραδειγμάτων δίνει την παρακάτω εικόνα :

Εικόνα 4.21 : Εφαρμογή φίλτρου πρώτης παραγώγου με μάσκα Roberts

Από το αποτέλεσμα της διαδικασίας παρατηρούμε πολύ φτωχή ανίχνευση ακμών. Αυτό συμβαίνει διότι η εικόνα της οποίας υπολογίζεται το φίλτρο Roberts έχει βάθος χρώματος 8. Το φίλτρο Roberts δεν μπορεί να αντιληφθεί καλά τις διαφορές στην φωτεινότητα (λόγω μικρής μάσκας και μικρού βάρους που δίνεται στα pixel), επομένως δεν δίνει αντίστοιχα σωστές εκτιμήσεις του τόνου της τελικής εικόνας.

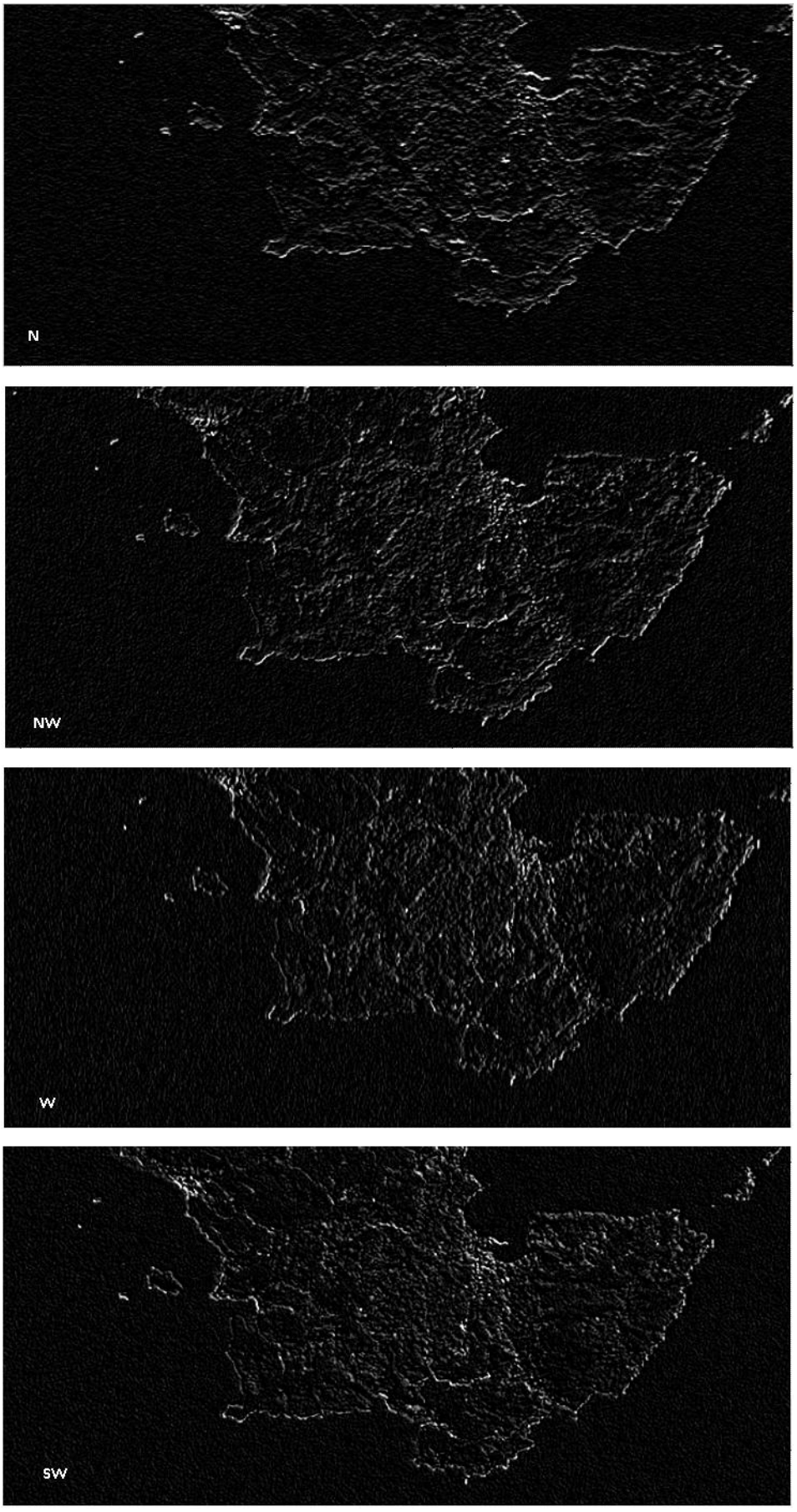
- Φίλτρο Kirsch

Το φίλτρο Kirsch είναι μία ειδική περίπτωση φίλτρου, καθώς είναι μη γραμμικό . Χρησιμοποιείται κυρίως για την ανίχνευση ακμών σε διαφορετικές διευθύνσεις, οι οποίες απέχουν μεταξύ τους κατά 45 μοίρες. Η κλάση που υλοποιεί τη διαδικασία αυτή παίρνει σαν όρισμα το θεματικό επίπεδο μίας εικόνας και την διεύθυνση για την οποία επιθυμείται η ανίχνευση των ακμών. Το αποτέλεσμα της διαδικασίας είναι μία εικόνα για την οποία έχουν υπολογιστεί οι ακμές στην κατεύθυνση που επιθυμείται. Η διαδικασία είναι προσβάσιμη μέσω του WPS request builder και έχει την ονομασία kirschFilter. Το γραφικό περιβάλλον της διαδικασίας φαίνεται παρακάτω :

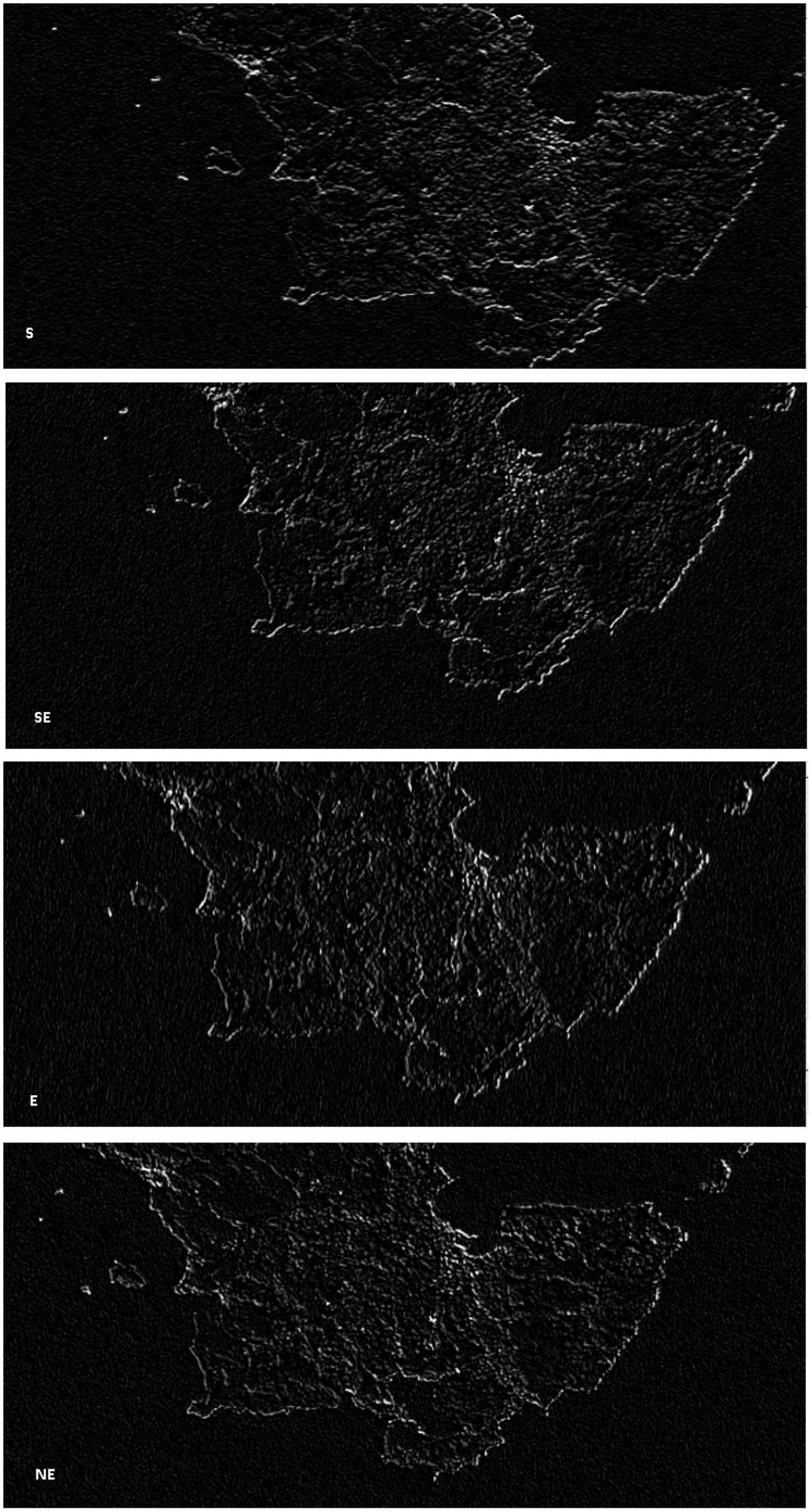


Εικόνα 4.22 : Το γραφικό περιβάλλον της διαδικασίας φίλτρου Kirsch

Η εφαρμογή του συγκεκριμένου φίλτρου για όλες τις διευθύνσεις (N, NW, W, SW, S, SE, E, NE) φαίνεται στην επόμενη εικόνα.



Εικόνα 4.23 : Φίλτρο kirsch για τις αναγραφόμενες ενδείξεις



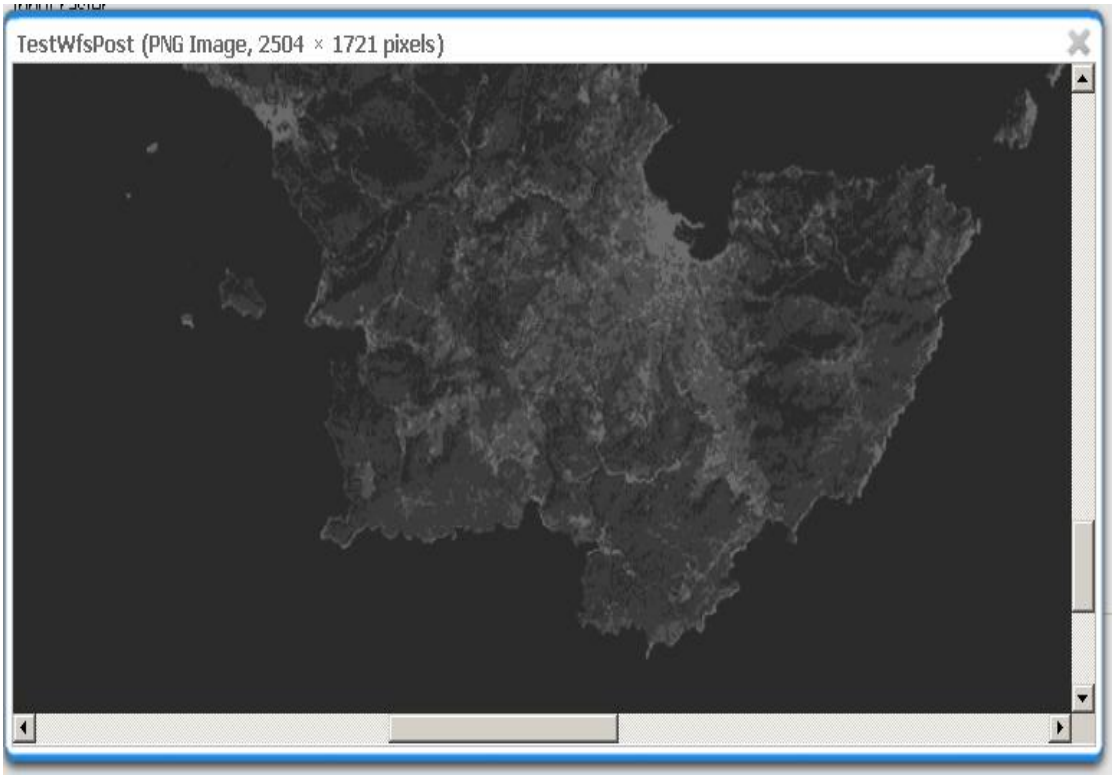
Εικόνα 4.24 : Φίλτρο kirsch για τις αναγραφόμενες ενδείξεις

Από τη σύγκριση των παρατιθέμενων εικόνων διαπιστώνουμε αρχικά ότι τα νότια, νοτιοανατολικά και νοτιοδυτικά φίλτρα έχουν μεγαλύτερη ευαισθησία στον κρουστικό θόρυβο. Αυτό μας δίνει και μία εικόνα της διάταξης των συστηματικών σφαλμάτων του συγκεκριμένου συστήματος.

Επίσης είναι σαφής ο διαχωρισμός των κατευθύνσεων των ακμών κατά περίπτωση. Παρατηρούμε ότι οι εικόνες αντίθετων κατευθύνσεων(π.χ. SE και NW), εμφανίζουν ενισχυμένες παρόμοιες ακμές . Το τελευταίο είναι ένα μέτρο της ορθότητας της διαδικασίας.

4.2.5 Αλγόριθμος K - means

Ο αλγόριθμος K – means δέχεται σαν όρισμα το θεματικό επίπεδο μίας εικόνας, τον αριθμό των κλάσεων στις οποίες θέλουμε να κατατμήσουμε την εικόνα, τον αριθμό των μέγιστων επαναλήψεων (σε περίπτωση μη σύγκλισης) και τον παράγοντα σύγκλισης. Η αρχικοποίηση των κέντρων γίνεται με τυχαία επιλογή στην εικόνα. Η διαδικασία επιστρέφει την κατατετμημένη εικόνα. Η κλάση που υλοποιεί τη διαδικασία αυτή είναι διαθέσιμη στον WPS request builder υπό την ονομασία, gs: Kmeans. Παρακάτω παρουσιάζεται η υλοποίηση του αλγορίθμου για τέσσερις κλάσεις, 10 επαναλήψεις και παράγοντα σύγκλισης 50%. Επίσης η εικόνα αφορά το κανάλι 3 του Landsat από την νότια περιοχή της Αλοννήσου :



Εικόνα 4.25 : Ταξινόμηση με τον αλγόριθμο K – means για 4 κλάσεις, 10 επαναλήψεις και παράγοντα σύγκλισης 1

Παρατηρούμε όπως ήταν αναμενόμενο, ότι η εξαγόμενη εικόνα απεικονίζει τις κλάσεις με διαφορετικό τόνο του γκρι. Παρόλα αυτά το αποτέλεσμα είναι ευδιάκριτο και οι κλάσεις ξεχωρίζουν μεταξύ τους.

Όσον αφορά την πιστότητα των κλάσεων, συγκρίνουμε το αποτέλεσμα της διαδικασίας με τα έγχρωμα σύνθετα που δημιουργήθηκαν στο προηγούμενο κεφάλαιο. Παρατηρείται ότι δημιουργείται αρκετά αντιπροσωπευτική ταξινομημένη εικόνα :

- Η κλάση που απεικονίζεται ως η δεύτερη πιο σκούρα κλάση έχει αναλάβει να αντιπροσωπεύσει τους θαμνώδεις λόφους όπου στο έγχρωμο σύνθετο 4-3-2 παρουσιάζονται ως κόκκινοι τόνοι.

- Η σκουρότερη κλάση αντιπροσωπεύει τα μέρη με αρδευόμενη βλάστηση, καθώς στο έγχρωμο σύνθετο 5-4-1 αυτά τα μέρη εμφανίζονται με σκούρο πράσινο χρώμα

- Η φωτεινότερη κλάση αντιπροσωπεύει τις αστικές περιοχές. Αυτές οι περιοχές φαίνονται στο έγχρωμο σύνθετο 4-3-2 ως λευκές, καθώς έχουν έντονη ανακλαστικότητα σε όλα τα κανάλια. Επίσης παρατηρούνται κάποια σφάλματα στην κατηγορία αυτή, καθώς συμπεριλαμβάνει κάποιες βραχώδεις περιοχές

- Τέλος η δεύτερη πιο φωτεινή κλάση απεικονίζει με πιστό τρόπο το γυμνό έδαφος. Αυτές οι περιοχές απεικονίζονται ως καφέ στο έγχρωμο σύνθετο 3-2-1.

Από τα παραπάνω συμπεραίνουμε ότι η παρούσα υλοποίηση αποτελεί είναι αξιόπιστη και μπορεί να χρησιμοποιηθεί για οποιαδήποτε εφαρμογή κατάτμησης. Η επέκταση του geoserver προς αυτήν την κατεύθυνση είναι σκόπιμη και ευνοεί την ανάπτυξη τηλεπισκοπικών μεθόδων σε GIS ανοιχτού κώδικα.

4.2.6 Pansharpening

Ο αλγόριθμος Pansharpening στην παρούσα υλοποίηση βρίσκεται στην dropdown list του WPS request builder υπό την ονομασία gs: IHSPansharpening. Όπως περιγράφηκε στα προηγούμενα, ο αλγόριθμος αυτός παίρνει σαν όρισμα τρείς εικόνες από διαφορετικά κανάλια του ίδιου δέκτη και μέσω της μετατροπής του συστήματος απεικόνισής τους από το σύστημα RGB στο σύστημα χρώματος IHS, δίνει ως αποτέλεσμα μία εικόνα η οποία προσομοιάζει στο αντίστοιχο έγχρωμο σύνθετο, αλλά παρέχει καλύτερη

ανάλυση. Το αποτέλεσμα για την ίδια περιοχή της Αλοννήσου για τα κανάλια 3,2,1 του Landsat είναι το παρακάτω :

Εικόνα 4.26 : Pansharpening στο Νότιο τμήμα της Αλοννήσου

Στην παραπάνω εικόνα συναντώνται όλα τα τυπικά χαρακτηριστικά μίας εικόνας που έχει προκύψει από τον αλγόριθμο Pansharpening.

Αρχικά η εικόνα συγκρινόμενη με το αντίστοιχο έγχρωμο σύνθετο (3-2-1) παρουσιάζει αισθητά καλύτερη χωρική ανάλυση. Στην πραγματικότητα η εικόνα αυτή έχει το μισό μέγεθος pixel από τα αντίστοιχα έγχρωμα σύνθετα.

Όσον αφορά τα χρώματα, παρατηρείται μία αρκετά πιστή αναπαράσταση και οι τόνοι που χρησιμοποιούνται είναι αρκετά κοντά στους αντίστοιχους φυσικούς τόνους. Επίσης, η αντίθεση των χρωμάτων είναι αυξημένη κάτι που σε συνδυασμό με την αυξημένη χωρική ανάλυση οδηγεί σε καλύτερη αναγνώριση λεπτομερειών (παρατηρείται σαφώς το οδικό δίκτυο).

Τα παραπάνω οδηγούν στο συμπέρασμα ότι η υλοποίηση του αλγορίθμου είναι αξιόπιστη και ανταποκρίνεται στο σκοπό ανάπτυξης του. Η ενσωμάτωση της διαδικασίας στον Geoserver μαζί με τις προηγούμενες υλοποιημένες διαδικασίες αποτιμάται με θετικό πρόσημο, καθώς συμβάλλει στη μετατροπή του σε σταθμό τηλεπισκοπικής επεξεργασίας.

5. Συμπεράσματα και προοπτικές εξέλιξης

Στο κεφάλαιο αυτό θα επιχειρηθεί μία συνολική αξιολόγηση της εργασίας που αναπτύχθηκε και θα επιχειρηθεί η χάραξη κάποιων κατευθύνσεων για το μέλλον της εφαρμογής. Επίσης θα επιχειρηθεί μία γενικότερη αξιολόγηση του ελεύθερου γεωχωρικού λογισμικού που χρησιμοποιήθηκε και θα περιγραφούν πλεονεκτήματα και μειονεκτήματα.

Στην πρώτη ενότητα θα αναλυθούν τα πλεονεκτήματα και μειονεκτήματα της εφαρμογής ως σύνολο και θα περιγραφούν οι δυνατότητες και οι αδυναμίες των τηλεπισκοπικών μεθόδων που αναπτύχθηκαν. Στην επόμενη ενότητα θα περιγραφούν οι προοπτικές εξέλιξης που διαφαίνονται στο χώρο του γεωχωρικού λογισμικού ανοιχτού κώδικα και η πιθανή επέκταση της παρούσας εφαρμογής προκειμένου να συναντήσει τις αλλαγές αυτές.

5.1 Η εφαρμογή ως σύνολο : πλεονεκτήματα και μειονεκτήματα

Στην υποενότητα αυτή θα περιγραφούν τα πλεονεκτήματα και μειονεκτήματα της εφαρμογής ως σύνολο και θα εξειδικευτούν σε κάθε διαδικασία :

Τα πλεονεκτήματα της εφαρμογής ως σύνολο είναι :

- Αναπτύχθηκε σε γλώσσα Java και το μεγαλύτερο μέρος των διαδικασιών χρησιμοποιεί τη βιβλιοθήκη JAI για την υλοποίηση. Αυτό εγγυάται τη βέλτιστη υλοποίηση στον τομέα της ταχύτητας, της αποσφαλμάτωσης και της έκτασης του κώδικα
- Λόγω του παραπάνω γεγονότος η εφαρμογή μπορεί να εγκατασταθεί και να τρέξει σε όλες τις πλατφόρμες και συστήματα (π.χ. κινητά τηλέφωνα με λογισμικό Android)
- Η εφαρμογή μπορεί να λειτουργήσει είτε ως server χωρικών δεδομένων και υπηρεσιών είτε ως εφαρμογή desktop δηλαδή να τρέχει τοπικά από το χρήστη
- Λόγω του γεγονότος ότι εδράζεται στον Geoserver η εφαρμογή έχει στη διάθεσή της ένα έτοιμο βελτιστοποιημένο σύστημα βάσης δεδομένων για την αποθήκευση και ανάκτηση των γεωχωρικών στοιχείων.
- Αποτέλεσμα του παραπάνω είναι η διάθεση στο χρήστη ενός βολικού γραφικού περιβάλλοντος εισαγωγής γεωχωρικών στοιχείων
- Υπάρχουν κλάσεις οι οποίες μετατρέπουν κάθε γεωχωρικό στοιχείο (ψηφιογραφικό και διανυσματικό) σε αντικείμενο της μνήμης και θεματικό επίπεδο αντίστοιχα. Με αυτόν τον τρόπο είναι δυνατός ο χειρισμός αντικειμένων πολλαπλών θεματικών επιπέδων και η εξαγωγή αποτελεσμάτων από ένα συνδυασμό αυτών.

- Η εφαρμογή αποτελείται από γεωχωρικό λογισμικό ανοιχτού κώδικα. Αυτό είναι καθοριστικής σημασίας για την δυνατότητα ανάγνωσης και τροποποίησης του κώδικα, καθώς και για την επέκτασή του.
- Υπάγεται στην άδεια GPL (Gnu *General Public License)* η οποία γράφτηκε αρχικά από τον Ρίτσαρντ Στόλλμαν για το εγχείρημα GNU και προστατεύει το μεγαλύτερο ποσοστό του ελεύθερου λογισμικού που υπάρχει μέχρι σήμερα.
- Λόγω του ελεύθερου χαρακτήρα του και της ευκολίας στο χειρισμό χωρικών δεδομένων η παρούσα εφαρμογή φιλοδοξεί να αποτελέσει πηγή γνώσης για την επιστήμη της τηλεπισκόπησης και τις μεθόδους της

Τα μειονεκτήματα της εφαρμογής είναι :

- Το μεγαλύτερο πρόβλημα με αυτήν την εφαρμογή είναι η απουσία κάποιου αξιόπιστου client ο οποίος να χειρίζεται τον geoserver και να ενσωματώνει διεπαφές ακόμα πιο φιλικές προς το χρήστη
- Οι διαδικασίες που αναπτύχθηκαν, απαιτούν βελτιστοποίηση ώστε να ελαχιστοποιηθεί ο χρόνος επεξεργασίας και να υποστηριχθούν περισσότερα format εισαγόμενων εικόνων
- Κάποιες επεξεργασίες που απαιτούν την εισαγωγή πολλών εικόνων και την αρχικοποίηση πολλών πινάκων (π.χ. Pansharpening) εξαντλούν τη μνήμη της εικονικής μηχανής της Java με αποτέλεσμα να μπλοκάρονται οι επεξεργασίες

Κάθε διαδικασία παρουσιάζει τα παρακάτω πλεονεκτήματα και μειονεκτήματα:

Τα πλεονεκτήματα για το κατώφλι είναι :

- Υλοποιείται με τη σύνταξη απλών αρχείων SLD και είναι απλό στην εφαρμογή για κάθε θεματικό επίπεδο
- Απαιτεί ελάχιστο χρόνο επεξεργασίας και λίγη υπολογιστική ισχύ
- Εφαρμόζεται για κάθε μέγεθος αρχείου

Τα μειονεκτήματα για το κατώφλι είναι :

- Απαιτεί τη γνώση της γλώσσας SLD για τη σύνταξη του αντίστοιχου αρχείου
- Η σύνταξη του αρχείου δεν συνοδεύεται από μία άμεση προεπισκόπηση του αποτελέσματος, αντιθέτως πρέπει να αποθηκευτεί το αρχείο και ύστερα να προβληθεί η εικόνα με το κατώφλι.

Τα πλεονεκτήματα για τα έγχρωμα σύνθετα είναι :

- Αποτελούν κόμβο επεξεργασίας εικόνας στη βιβλιοθήκη JAI και ως εκ τούτου είναι βελτιστοποιημένη διαδικασία
- Δίνεται η δυνατότητα δημιουργίας οποιουδήποτε έγχρωμου σύνθετου από οποιοδήποτε τρία κανάλια κάποιας πολυφασματικής εικόνας
- Τα χρώματα και η απόδοση των λεπτομερειών, όπως αναλύθηκε και στο προηγούμενο κεφάλαιο είναι πιστά και η εξαγωγή συμπερασμάτων είναι μία εύκολη υπόθεση

Τα μειονεκτήματα για τα έγχρωμα σύνθετα είναι :

- Η διαδικασία απαιτεί σχετικά μεγάλη υπολογιστική ισχύ και αρχικοποιεί αρκετά μεγάλο ποσό μνήμης
- Τα εξαγόμενο της διαδικασίας παραδίδεται μόνο σε format PNG ή JPEG

Τα πλεονεκτήματα για τα φίλτρα είναι :

- Τα αποτελέσματα που παρουσιάζουν σύμφωνα με την ανώτερη ενότητα, είναι πιστά και βοηθούν στην εξαγωγή συμπερασμάτων
- Αποτελούν κόμβο στην επεξεργασία εικόνας της βιβλιοθήκης JAI
- Έχουν πολλές επιλογές εφαρμογής ως προς κάποια κατεύθυνση ή με διαφορετικό μέγεθος μάσκας

Τα μειονεκτήματα για τα φίλτρα είναι :

- Για εικόνες που απαιτούν μεγάλη μνήμη, κάποιες φορές παρουσιάζονται σφάλματα, καθώς εξαντλείται η μνήμη της εικόνικής μηχανής

Τα πλεονεκτήματα για τον αλγόριθμο K - means είναι :

- Έχει πολλές παραμέτρους ρύθμισης από το χρήστη (αριθμός μέγιστων επαναλήψεων, παράγοντας σύγκλισης) και η διαδικασία της ταξινόμησης προσαρμόζεται στις ανάγκες του χρήστη
- Όπως περιγράφηκε παραπάνω, παρουσιάζει πιστότητα στις εμφανιζόμενες κλάσεις και μπορεί να χρησιμοποιηθεί για την εξαγωγή συμπερασμάτων

- Απαιτεί σχετικά λίγα υπολογιστικά βήματα και σχετικά μικρή αρχικοποίηση μνήμης, κάτι που συνεπάγεται ταχύτητα στους υπολογισμούς

Τα μειονεκτήματα για τον αλγόριθμο K - means είναι :

- Δεν αποτελεί κόμβο επεξεργασίας εικόνας της βιβλιοθήκης JAI, επομένως χρήζει περαιτέρω επεξεργασίας και βελτιστοποίησης
- Δεν υποστηρίζει κάποια format εικόνων εισαγωγής (συγκεκριμένα αυτά που έχουν κάποιο ασυνήθιστο ColorModel και SampleModel) επομένως παρουσιάζει κάποια σφάλματα στην περίπτωση εισαγωγής τέτοιων εικόνων

Τα πλεονεκτήματα για τον αλγόριθμο Pansharpening είναι :

- Τα χρώματα παρουσιάζουν μεγάλη πιστότητα και προσεγγίζουν τα πραγματικά
- Η εικόνα που παράγεται παρότι διπλάσιας ανάλυσης δεν είναι διπλάσια σε μέγεθος, λόγω της εσωτερικής συμπίεσης
- Η χρήση διάφορων κόμβων επεξεργασίας εικόνας της βιβλιοθήκης JAI, εγγυάται ότι το αποτέλεσμα θα παραχθεί ανεξάρτητα από τον τύπο των δεδομένων εισαγωγής

Τα μειονεκτήματα για τον αλγόριθμο Pansharpening είναι :

- Και εδώ λόγω της εισαγωγής πολλών εικόνων και της αρχικοποίησης πολλών πινάκων, η διαδικασία κάποιες φορές εξαντλεί την μνήμη της εικονικής μηχανής της Java

5.2 Προοπτικές εξέλιξης – επέκτασης

Οι προοπτικές εξέλιξης και επέκτασης των συγκεκριμένων επεξεργασιών είναι πολλές. Ο Geoserver είναι ένα σχετικά καινούργιο project και έχει πολύ δρόμο να διανύσει ακόμα. Σαν αρχικός στόχος θα πρέπει να τίθεται η εξάλειψη των προαναφερμένων μειονεκτημάτων. Επίσης διαφάνηκε από τα προηγούμενα ότι ο geoserver μπορεί να επιτελέσει το ρόλο ενός server επεξεργασίας τηλεπισκοπικών δεδομένων και μεθόδων. Κάποιες προτάσεις προς την επίτευξη αυτών των στόχων είναι :

- Αρχικά θα πρέπει να εξαλειφθεί το λειτουργικό πρόβλημα που αναφέρθηκε στο προηγούμενο κεφάλαιο και έχει να κάνει με την ορθή δρομολόγηση των WPS POST request και την ορθή δημιουργία απάντησης από τον Geoserver
- Μετά τη επίλυση του παραπάνω προβλήματος, θα πρέπει να αναπτυχθεί ένας client σε κάποια γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη web εφαρμογών (π.χ. JavaScript, HTML, PHP). Αυτό θα δώσει τη δυνατότητα ανάπτυξης εξειδικευμένων client-side εφαρμογών, οι οποίες θα μπορούν να χρησιμοποιηθούν για ευρύτερες εφαρμογές.
- Με την ανάπτυξη της συνεργασίας client-side και server-side εφαρμογών θα μπορούσε να κατασκευαστεί μία ιστοσελίδα η οποία θα περιελάμβανε όλες τις τυπικές επεξεργασίες τηλεπισκόπησης
- Ανάλογες διαδικασίες που μπορούν να αναπτυχθούν με τη συνεργασίας server και client είναι οι διαδικασίες της επιβλεπόμενης ταξινόμησης σε δύο στάδια (εκπαίδευση και ταξινόμηση), καθώς και εφαρμογές νευρωνικών δικτύων δύο σταδίων
- Αποσφαλμάτωση και βελτιστοποίηση των ήδη δημιουργημένων διαδικασιών ώστε να γίνει η συγκεκριμένη εφαρμογή ένας αξιόπιστος τηλεπισκοπικός σταθμός
- Με τη χρήση εικονικής μηχανή 64 bit τα σφάλματα μνήμης πρακτικά μηδενίζονται και δίνονται δυνατότητες επεξεργασίας σε μεγαλύτερες εικόνες και σετ δεδομένων. Επομένως συνίσταται η χρήση τέτοιων εικονικών μηχανών
- Η μετατροπή της εφαρμογής σε τοπική εφαρμογή για χρήση σε άλλες συσκευές (π.χ. σε λογισμικό android)

Βιβλιογραφία

1. Αργιαλάς, Δ., 1999, **Φωτοερμηνεία-Τηλεπισκόπηση.**
2. Αργιαλάς, Δ., 1998, **Ψηφιακή Τηλεπισκόπηση.**
3. D. P. Argialas , S. Krishnamurthy , **DETECTION OF LINES AND CIRCLES IN MAPS AND ENGINEERING DRAWINGS**
4. Deng_Meixia, 2000, **AN INTEROPERABLE WEB-BASED CLASSIFICATION SERVICE FOR REMOTE SENSING DATA**
5. Rana Abdul Rahman Lateef, **Expansion and Implementation of a 3x3 Sobel and Prewitt Edge Detection Filter to a 5x5 Dimension Filter**
6. Ελένη Δ. Τσίγκα , 2005, **Εφαρμογή διαχείρισης και παροχής γεωκαθορισμένης πολυμεσικής πληροφορίας, με χρήση ανοικτών προτύπων και εργαλείων.**
7. Dr. H. B. Kekre , 2010, **Image Segmentation using Extended Edge Operator for Mammographic Images**
8. Rafael Santos, **Java Advanced Imaging API: A Tutorial**
9. Nataliia Kussul , Andrii Shelestov , Serhiy Skakun, 2008, **Grid system for flood extent extraction from satellite images**
10. Κοντόπουλος Γεώργιος , 2010 , **Ανάπτυξη διαδικτυακών εφαρμογών GIS με λογισμικό ανοιχτού κώδικα (Geoserver).**
11. Πανάγου Μαρία , 2008 , **Δημοσιοποίηση σεισμολογικών δεδομένων στον παγκόσμιο ιστό**
12. William B. Thompson, Gregory W. Thoenen, Ronald G. Moore, Thomas C. Henderson, 1998 , **EXTRACTION OF MICRO-TERRAIN FEATURES**
13. Pierre Soille , 2002 , **Advances in the Analysis of Topographic Features on Discrete Images**

Εγχειρίδια

14. Refsnes Data, 1999-2014, **HTML Tutorial**, http://www.w3schools.com/html/default.asp
15. Refsnes Data, 1999-2014, **JavaScript Tutorial,** http://www.w3schools.com/js/default.asp
16. Refsnes Data, 1999-2014, **CSS Tutorial,** http://www.w3schools.com/css/default.asp
17. Refsnes Data, 1999-2014, **XML Tutorial,** http://www.w3schools.com/xml/default.asp
18. Refsnes Data, 1999-2014, **Jquery Tutorial,** http://www.w3schools.com/jquery/default.asp
19. Refsnes Data, 1999-2014, **Ajax Tutorial**, http://www.w3schools.com/ajax/default.asp
20. GeoServer Team, **GeoServer Documentation,** http://docs.geoserver.org/
21. OpenLayers Community, 2008, **OpenLayers Documentation,** http://docs.openlayers.org/

22. OpenLayers Community, 2008, **OpenLayers Examples,** http://openlayers.org/dev/examples/
23. Boundless Spatial, **Introduction to OpenLayers,** http://workshops.boundlessgeo.com/openlayers-intro/
24. Open Geospatial Consortium, 1994-2014, **OGC Standards,** http://www.opengeospatial.org/standards/is
25. Lawrence H. Rodrigues , 2001 , **Rendering Images in JAI,** http://www.informit.com/articles/article.aspx?p=23668&seqNum=1
26. JAI Team, **Programming in Java Advanced Imaging,** http://iihm.imag.fr/Docs/java/jai1_0guide/Programming-environ.doc.html
27. Geosolutions Team , **GeoServer training** , http://geoserver.geo-solutions.it/edu/en/wps/rendering_tx.html
28. OpenGeo, **Spatial processing and analysis ,** http://suite.opengeo.org/opengeo-docs/processing/index.html
29. Rafael Santos , **Java Image Processing Cookbook ,** http://www.lac.inpe.br/JIPCookbook/index.jsp
30. OSGeo, **Εγχειρίδιο Γρήγορης Εκκίνησης GeoServer ,** http://live.osgeo.org/el/quickstart/geoserver_quickstart.html

Παράρτημα

Στο παράρτημα που ακολουθεί, παρατίθεται ο κώδικα που χρησιμοποιήθηκε για κάθε τηλεπισκοπική μέθοδο

1. Ιστόγραμμα εικόνας

```
@DescribeProcess(title="Histogram", description="Display the histogram analytics for the image")
public class MyHisto implements GeoServerProcess{

	@DescribeResult(name="result", description="output analytics")
	public String execute(
			@DescribeParameter(name = "coverage", description = "Input raster") GridCoverage2D coverage
			)throws ProcessException{

	//read the image
	RenderedImage image = coverage.getRenderedImage();
	if (image.getSampleModel().getDataType() == DataBuffer.TYPE_BYTE)
	return doHistogram(image,256);
	// Is it a 16-bit image?
	else if ((image.getSampleModel().getDataType() == DataBuffer.TYPE_SHORT) ||
	(image.getSampleModel().getDataType() == DataBuffer.TYPE_USHORT))
	return doHistogram(image,65536);
	else // OK, bail out with some information for the user.
	{
   String str = null;
	switch(image.getSampleModel().getDataType())
	{
	case DataBuffer.TYPE_INT: str = "Integer"; break;
	case DataBuffer.TYPE_FLOAT: str="Float"; break;
	case DataBuffer.TYPE_DOUBLE: str="Double"; break;
	case DataBuffer.TYPE_UNDEFINED: str="Undefined"; break;
	default: str="Other -- weird!"; break;
	}
	throw new ProcessException("The type of"+str+"Data buffer is unsupported");
```

```
        }
        }

        // Auxiliary method to create the histogram and display information about it.
        private static String doHistogram(RenderedImage image,int nbins)
        {
        // Create the histogram.
        ParameterBlock pb = new ParameterBlock();
        pb.addSource(image);
        pb.add(null); // The ROI -- all image.
        pb.add(1); pb.add(1); // Sampling -- each and every pixel.
        pb.add(new int[]{nbins}); // Bins.
        pb.add(new double[]{0}); pb.add(new double[]{nbins}); // Range for inclusion.
        RenderedOp dummyImage = JAI.create("histogram", pb);
        // Get the histogram from the RenderedOp.
        Histogram histogram = (Histogram)dummyImage.getProperty("histogram");
        // Print some histogram stats.
        // For each band...
        String str1 = null;
        switch(histogram.getNumBands()){
        case 3:
        str1= "Histogram with "+nbins+" bins:\n"+"Band "+1+":"+" Mean: "+histogram.getMean()[0]+" StdDev: "+histogram.getStandardDeviation()[0]+" Entropy: "+histogram.getEntropy()[0]+" Total Count: "+histogram.getTotals()[0]+"\n"
                +"Band "+2+":"+" Mean: "+histogram.getMean()[1]+" StdDev: "+histogram.getStandardDeviation()[1]+" Entropy: "+histogram.getEntropy()[1]+" Total Count: "+histogram.getTotals()[1]+"\n"
                +"Band "+3+":"+" Mean: "+histogram.getMean()[2]+" StdDev: "+histogram.getStandardDeviation()[2]+" Entropy: "+histogram.getEntropy()[2]+" Total Count: "+histogram.getTotals()[2];break;
        case 2:
        str1="Histogram with "+nbins+" bins:\n"+"Band "+1+":"+" Mean: "+histogram.getMean()[0]+" StdDev: "+histogram.getStandardDeviation()[0]+" Entropy: "+histogram.getEntropy()[0]+" Total Count: "+histogram.getTotals()[0]+"\n"
                    +"Band "+2+":"+" Mean: "+histogram.getMean()[1]+" StdDev: "+histogram.getStandardDeviation()[1]+" Entropy: "+histogram.getEntropy()[1]+" Total Count: "+histogram.getTotals()[1];break;
        case 1:
        str1= "Histogram with "+nbins+" bins:\n"+"Band "+1+":"+" Mean: "+histogram.getMean()[0]+" StdDev:
```

```
"+histogram.getStandardDeviation()[0]+"                    Entropy:
"+histogram.getEntropy()[0]+" Total Count: "+histogram.getTotals()[0];break;
        }
        return str1;
        }

}
```

2. Έγχρωμα σύνθετα

```
@DescribeProcess(title="ColourComposite",      description="Merge      three
imagery bands to create a colour composite")
public class BandMerge implements GeoServerProcess {

        @DescribeResult(name="result", description="Composite raster")
        public RenderedImage execute(
                        @DescribeParameter(name = "RedBand", description =
"Input raster") GridCoverage2D rcoverage,
                        @DescribeParameter(name = "GreenBand", description =
"Input raster") GridCoverage2D gcoverage,
                        @DescribeParameter(name = "BlueBand", description =
"Input raster") GridCoverage2D bcoverage
                        )throws ProcessException{
        //public static void main(String[] args) {

                // First we open the input images. We assume that each band is in a
separate file.
                RenderedImage rendRed =  //JAI.create("fileload", args[0]);
                                rcoverage.getRenderedImage();
                RenderedImage rendGreen =  //JAI.create("fileload", args[1]);
                gcoverage.getRenderedImage();
                RenderedImage rendBlue =  //JAI.create("fileload", args[2]);
                bcoverage.getRenderedImage();

                //fill the parameter block with source images
                ParameterBlock pb = new ParameterBlock();
                pb.setSource(rendRed,0);
                pb.setSource(rendGreen,1);
                pb.setSource(rendBlue,2);
                // JAI rendering node !!!!!!!!
                RenderedImage result = JAI.create("bandmerge",pb,null);
                //JAI.create( "filestore",result,"composite.png","PNG");
                return result;
        }
```

3. Φίλτρα (ενδεικτικά δίνεται η περίπτωση του φίλτρου Sobel)

```
@DescribeProcess(title="Sobel", description="Edge detection with sobel filter")
public class Sobel  implements GeoServerProcess {

        @DescribeResult(name="result", description="output raster")
        public RenderedImage execute(
                        @DescribeParameter(name = "coverage", description = "Input raster") GridCoverage2D coverage,
                        @DescribeParameter(name = "Direction", description = "The edge direction to detect. 0 for horizontal, 1 for vertical") Integer direction,
                        @DescribeParameter(name = "Mask Size", description = "The size of the mask") Integer size
                        )throws ProcessException {
//public static void main (String[] args ){

                RenderedImage re = coverage.getRenderedImage() ;
                //RenderedImage re = JAI.create("fileload", args[0]);

                float [] kernelMatrix3v = { -1, -2, -1,
                                            0,  0,  0,
                                            1,  2,  1 };

                float [] kernelMatrix3h ={-1,0,1,
                                          -2,0,2,
                                          -1,0,1};

                float [] kernelMatrix5v = { 2, 3 , 4 , 3 , 2,
                                            1, 2 , 3 , 2 , 1,
                                            0, 0 , 0 , 0 , 0,
                                           -1,-2, -3 ,-2 , -1,
                                           -2,-3, -4 ,-3 , -2};

                float [] kernelMatrix5h = {2 , 1 , 0 ,-1, -2,
                                           3 , 2 , 0 ,-2, -3,
                                           4 , 3 , 0 ,-3, -4,
                                           3 , 2 , 0 ,-2, -3,
                                           2 , 1 , 0 ,-1, -2};

                float [] kernelMatrix7v = { 3, 4, 5, 6, 5, 4, 3,
                                            2, 3, 4, 5, 4, 3, 2,
                                            1, 2, 3, 4, 3, 2, 1,
                                            0, 0, 0, 0, 0, 0, 0,
```

```
                                                    -1,-2,-3,-4,-3,-2,-1,
                                                    -2,-3,-4,-5,-4,-3,-2,
                                                    -3,-4,-5,-6,-5,-4,-3};

                float [] kernelMatrix7h = {3, 2, 1, 0, -1, -2, -3,
                                                    4, 3, 2, 0, -2, -3, -4,
                                                    5, 4, 3, 0, -3, -4, -5,
                                                    6, 5, 4, 0, -4, -5, -6,
                                                    5, 4, 3, 0, -3, -4, -5,
                                                    4, 3, 2, 0, -2, -3, -4,
                                                    3, 2, 1, 0, -1, -2, -3};

                float [] kernelMatrix9v = { 4, 5, 6, 7, 8, 7, 6, 5, 4,
                                            3, 4, 5, 6, 7, 6, 5, 4, 3,
                                            2, 3, 4, 5, 6, 5, 4, 3, 2,
                                            1, 2, 3, 4, 5, 4, 3, 2, 1,
                                                    0, 0, 0, 0, 0, 0, 0, 0, 0,
                                                    -1,-2,-3,-4,-5,-4,-3,-2,-
1,
                                                    -2,-3,-4,-5,-6,-5,-4,-3,-
2,
                                                    -3,-4,-5,-6,-7,-6,-5,-4,-
3,
                                                    -4,-5,-6,-7,-8,-7,-6,-5,-
4 };
                float [] kernelMatrix9h = {4, 3, 2, 1, 0, -1, -2, -3, -4,
                                            5, 4, 3, 2, 0, -2, -3, -4, -5,
                                            6, 5, 4, 3, 0, -3, -4, -5, -6,
                                            7, 6, 5, 4, 0, -4, -5, -6, -7,
                                            8, 7, 6, 5, 0, -5, -6, -7, -8,
                                            7, 6, 5, 4, 0, -4, -5, -6, -7,
                                            6, 5, 4, 3, 0, -3, -4, -5, -6,
                                            5, 4, 3, 2, 0, -2, -3, -4, -5,
                                            4, 3, 2, 1, 0, -1, -2, -3, -4};

                if (direction == 0){
                        if (size==3){
                                KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix3h);
                                return JAI.create("convolve", re, kernel);
                        }
                        else if (size==5){
```

```
                    KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix5h);
                    return JAI.create("convolve", re, kernel);
                }
                else if (size==7){
                    KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix7h);
                    return JAI.create("convolve", re, kernel);
                }
                else if (size==9){
                    KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix9h);
                    return JAI.create("convolve", re, kernel);
                }
                else{
                    throw new ProcessException("Invalid mask size.
Must be one of 3,5,7,9");
                }
            }
                else if(direction == 1){
                    if (size==3){
                        KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix3v);
                        return JAI.create("convolve", re, kernel);
                    }
                    else if (size==5){
                        KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix5v);
                        return JAI.create("convolve", re, kernel);
                    }
                    else if (size==7){
                        KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix7v);
                        return JAI.create("convolve", re, kernel);
                    }
                    else if (size==9){
                        KernelJAI kernel = new
KernelJAI(size,size,kernelMatrix9v);
                        return JAI.create("convolve", re, kernel);
                    }
                    else{
                        throw new ProcessException("Invalid mask
size. Must be one of 3,5,7,9");
                    }
                }
                else {
```

```
                throw new ProcessException("Invalid direction. Must be either 0 for horizontal or 1 for vertical");
            }

            //JAI.create("filestore",output,"Sobeled3v.png","PNG");


        }

}
```

4. Φίλτρα πρώτης παραγώγου (ενδεικτικά δίνεται με μάσκα Sobel)

```
@DescribeProcess(title="GradientMagnitude", description="Edge detection with gradient magnitude filter")
public class SobelGradient implements GeoServerProcess{

        @DescribeResult(name="result", description="output raster")
        public RenderedImage execute(
                        @DescribeParameter(name = "coverage", description = "Input raster") GridCoverage2D coverage
                        )throws ProcessException {

        //import image
        RenderedImage ri = coverage.getRenderedImage();

        //construct the masks
        KernelJAI sh = KernelJAI.GRADIENT_MASK_SOBEL_HORIZONTAL;
        KernelJAI sv = KernelJAI.GRADIENT_MASK_SOBEL_VERTICAL;

        RenderedImage gm = JAI.create("gradientmagnitude", ri, sh, sv);
        return gm;
        }
}
```

5. Αλγόριθμος K - Means

a. Κυρίως αλγόριθμος

```
public class KMeansImageClustering {

    // The output (clustered) image.
    private TiledImage pOutput;
    // The input image dimensions.
    private int width,height,numBands;
    // Some clustering parameters.
    private int maxIterations,numClusters;
    // The iteration counter will be global so we can get its value on the
    // middle of the clustering process.
    private int iteration;
   // A metric of clustering "quality".
    private double sumOfDistances;
    // In each iteration of the algorithm we will calculate the sum of squared
distances betw
    // pixels and cluster centers. If the difference between this sum over two
consecutive it
    // is smaller than epsilon we will assume the algorithm isn't converging
anymore and stop
    private double epsilon;
    // Flags and counters.
    private boolean finished = false;
    private long position;
    // The cluster centers.
    private float[][] clusterCenters;
    // The cluster assignment counter.
    private int[] clusterAssignmentCount;
    // A big array with all the input data and a small one for a pixel.
    private int[] inputData,aPixel;
    // A big array with the output data (cluster indexes).
    private short[][] outputData;

    /**
 69    * The constructor for the class, which sets the input image, the number of
 70    * desired clusters, the maximum number of iterations, a value that will be
 71    * used to decide whether the convergence has stopped and the initial
 72    * clusters centers initialization mode. It also allocates needed memory.
 73    * @param pInput the input planar image.
 74    * @param numClusters the desired number of clusters.
 75    * @param maxIterations the maximum number of iterations.
```

```
76       * @param epsilon a small value used to verify if clustering has converged.
77    * @param initMode the initialization mode: 'R' for random values, 'S' for
78    *        evenly spaced values, 'D' for randomly sampled data values.
79    */
   public KMeansImageClustering(PlanarImage pInput,int numClusters,int maxIterations,
                                double epsilon,char initMode)
     {
     // Get the image dimensions.
     width = pInput.getWidth();
     height = pInput.getHeight();
     numBands = pInput.getSampleModel().getNumBands();
     // Create the output image based on the input image's dimensions.
     /*
                  SampleModel                    sampleModel                    =
RasterFactory.createPixelInterleavedSampleModel(DataBuffer.
                  // Create a compatible ColorModel.
                  ColorModel                     colorModel                     =
PlanarImage.createColorModel(sampleModel);
     pOutput                                =                                  new
TiledImage(pInput.getMinX(),pInput.getMinY(),width,height,
                           pInput.getMinX(),pInput.getMinY(),
                           sampleModel,colorModel); */
    // Get some clustering parameters.
    this.numClusters = numClusters;
    this.maxIterations = maxIterations;
    this.epsilon = epsilon;
    iteration = 0;
     // We need arrays to store the clusters' centers and assignment counts.
     clusterCenters = new float[numClusters][numBands];
     clusterAssignmentCount = new int[numClusters];
     // Gets the raster for the input image.
    Raster raster = pInput.getData();
    // Gets the whole image data on memory. Get memory for a single pixel too.
    inputData = new int[width*height*numBands];
    aPixel = new int[numBands];
    // Gets memory for the output data (cluster indexes).
    outputData = new short[width][height];
    raster.getPixels(raster.getMinX(),raster.getMinY(),width,height,inputData);
    // Initialize the clusters centers, depending on the initialization mode.
    // Evenly spaced values (over main diagonal).
    if ((initMode == 's') || (initMode == 'S'))
      {
      // Discover the extrema value for the image data.
      double[] extrema = getExtrema(pInput);
```

```
      double delta = (extrema[1]-extrema[0])/(numClusters-1);
      for(int cluster=0;cluster<numClusters;cluster++)
        {
        for(int band=0;band<numBands;band++)
          clusterCenters[cluster][band] = (float)extrema[0];
        extrema[0] += delta;
        }
      }
    // Randomly sampled data values.
    else if ((initMode == 'd') || (initMode == 'D'))
      {
      // Create a Iterator to get the samples.
      RandomIter iterator = RandomIterFactory.create(pInput,null);
      for(int cluster=0;cluster<numClusters;cluster++)
        {
        int rx = (int)(Math.random()*width);
        int ry = (int)(Math.random()*height);
        iterator.getPixel(rx,ry,aPixel);
        for(int band=0;band<numBands;band++)
          clusterCenters[cluster][band] = aPixel[band];
        }
      }
    // Random values, using the maximum of the data values as limit.
    else // if ((initMode == 'r') || (initMode == 'R'))
      {
      // Discover the maximum value for the image data.
      double[] extrema = getExtrema(pInput);
      for(int cluster=0;cluster<numClusters;cluster++)
        for(int band=0;band<numBands;band++)
          clusterCenters[cluster][band] = (float)(extrema[1]*Math.random());
      }
    }

  /**
147     * This method returns a two-element array with the minimum and
maximum
148   * values found in a PlanarImage.
149   * @param image the input PlanarImage.
150     * @return an array with two positions: the first is the minima of the
image
151   *         and the second is the maxima.
152   */
   private double[] getExtrema(PlanarImage image)
     {
     ParameterBlock pbMaxMin = new ParameterBlock();
     pbMaxMin.addSource(image);
```

```
    RenderedOp extremaOp = JAI.create("extrema", pbMaxMin);
    // Must get the extrema of all bands !
    double[] allMins = (double[])extremaOp.getProperty("minimum");
    double[] allMaxs = (double[])extremaOp.getProperty("maximum");
    double minValue = allMins[0];
    double maxValue = allMaxs[0];
    for(int v=1;v<allMins.length;v++)
      {
      if (allMins[v] < minValue) minValue = allMins[v];
      if (allMaxs[v] > maxValue) maxValue = allMaxs[v];
      }
    return new double[] {minValue,maxValue};
    }

  /**
172   * This method performs the bulk of the processing. It runs the classic
173   * K-Means clustering algorithm:
174   * 1 - Scan the image and calculate the assignment vector.
175   * 2 - Scan the assignment vector and recalculate the cluster centers.
176   * 3 - Calculate statistics and repeat from 1 if needed.
177   */
   public void run()
    {
    double lastSumOfDistances=0;
    iterations: // Label for main loop.
      for(iteration=0;iteration<maxIterations;iteration++)
        {
        // 0 - Clean the cluster assignment vector.
        Arrays.fill(clusterAssignmentCount,0);
        // 1 - Scan the image and calculate the assignment vector.
        for(int h=0;h<height;h++)
         for(int w=0;w<width;w++)
            {
            // Where does the individual pixel data start ?
            int index = (h*width+w)*numBands;
            // Gets the class for this pixel.
            for(int band=0;band<numBands;band++) aPixel[band] =
inputData[index+band];
            short aClass = getClassFor(aPixel);
            // Update the position index.
            position += numBands + numClusters*numBands;
            outputData[w][h] = aClass;
            clusterAssignmentCount[aClass]++;
            }
        // 2 - Scan the assignment vector and recalculate the cluster centers.
        for(int cluster=0;cluster<numClusters;cluster++)
```

```
          Arrays.fill(clusterCenters[cluster],0f);
        // Update the position index.
        position += numClusters;
        for(int h=0;h<height;h++)
          for(int w=0;w<width;w++)
            {
            int theCluster = outputData[w][h];
            for(int band=0;band<numBands;band++)
              {
              int index = (h*width+w)*numBands;
              clusterCenters[theCluster][band] += inputData[index+band];
              }
            // Update the position index.
            position += numBands;
            }
        // 2a - Recalculate the centers.
        for(int cluster=0;cluster<numClusters;cluster++)
          if (clusterAssignmentCount[cluster] > 0)
            for(int band=0;band<numBands;band++)
              clusterCenters[cluster][band]/=clusterAssignmentCount[cluster];
        // Update the position index.
        position += numClusters*numBands;
        // 3 - Calculate statistics and repeat from 1 if needed.
        sumOfDistances = 0;
        for(int h=0;h<height;h++)
          for(int w=0;w<width;w++)
            {
            // To which class does this pixel belong ?
            short pixelsClass = outputData[w][h];
            // Calculate the distance between this pixel's values and its
            // assigned cluster center values.
            double distance = 0;
            int index = (h*width+w)*numBands;
            for(int band=0;band<numBands;band++)
             {
              double e1 = inputData[index+band];
              double e2 = clusterCenters[pixelsClass][band];
              double diff = (e1-e2)*(e1-e2);
              distance += diff;
              }
            distance = Math.sqrt(distance);
           sumOfDistances += distance;
            // Update the position index.
            position += numBands;
            }
        // Is it converging ?
```

```
        if (iteration > 0)
          if  (Math.abs(lastSumOfDistances-sumOfDistances)  <  epsilon)  break
iterations;
        lastSumOfDistances = sumOfDistances;
        } // end of the iterations loop.
    finished = true;
    // Means that all calculations are done, too.
    position = getSize();
    }

  /**
258   * This auxiliary method gets the class for a pixel vector.
259   * @return the class (cluster index) for a pixel.
260   */
  private short getClassFor(int[] pixel)
    {
    // Let's compare this pixel data with all the cluster centers.
    float closestSoFar = Float.MAX_VALUE;
    short classSoFar = 0;
    for(short cluster=0;cluster<numClusters;cluster++)
      {
      // Calculate the (quick) distance from this pixel to that cluster center.
      float distance = 0f;
      for(int band=0;band<numBands;band++)
        distance += Math.abs(clusterCenters[cluster][band]-pixel[band]);
     if (distance < closestSoFar)
        {
        closestSoFar = distance;
        classSoFar = cluster;
        }
      }
    return classSoFar;
    }

  /**
282   * This method returns the output image. It can return an image which is
283   * being processed while this method run, so results are not guaranteed
284   * to be final (and can be very strange if the image is large).
285   * @return the (possibly temporary) clustering results.
286
 * @throws Exception */
  public RenderedImage getOutput() throws ProcessException
  {
  // We must dump the crossed contents of the cluster centers and the
  // cluster indexes into the resulting image.
  // Create a SampleModel for the output data.
```

```
  //SampleModel sampleModel =

//RasterFactory.createBandedSampleModel(DataBuffer.TYPE_INT,width,heig
ht,numBands);
  // Create a WritableRaster using that sample model.
  //WritableRaster raster =
RasterFactory.createWritableRaster(sampleModel,new Point(0,0));
  // A pixel array will contain all bands for a specific x,y.
  //int[] pixelArray = new int[numBands];
  // For all pixels in the image...
  int [] data =new int[width*height*numBands];// Image data array.
  int count = 0;// Temporary counter.
  for(int h=0;h<height;h++)
    for(int w=0;w<width;w++)
      {
      // Get the class (cluster center) for that pixel.
      short aClass = outputData[w][h];
      // Fill the array with the cluster center.
      for(int band=0;band<numBands;band++) {
         data[count+band]= (int)clusterCenters[aClass][band];
         //pixelArray[band]= (int)clusterCenters[aClass][band];
      }
      count += numBands;
      }
  SampleModel sm =
RasterFactory.createPixelInterleavedSampleModel(DataBuffer.TYPE_INT,wid
th
         // Create a compatible ColorModel.

         // Create a WritableRaster.
         WritableRaster ra =
RasterFactory.createInterleavedRaster(DataBuffer.TYPE_INT, sm.getWidt
         ra.setPixels(0,0,ra.getWidth(),ra.getHeight(), data);
 // Set the raster on the output image.
         ColorModel cm = PlanarImage.createColorModel(sm);
  //check compatibility and replace if needed
         if(!JDKWorkarounds.areCompatibleDataModels(sm,cm)) {
                throw new ProcessException("Incompatible Color Model");
         }
         //
  TiledImage src = new TiledImage(ra.getMinX(), ra.getMinY(),
ra.getWidth(), ra.getHeight(),
  src.setData(ra);

    ParameterBlock pbConvert = new ParameterBlock();
    pbConvert.addSource(src);
```

```
    pbConvert.add(DataBuffer.TYPE_BYTE);
    RenderedImage bOut = JAI.create("format", pbConvert);

    return bOut;
  }


  /**
316   * This method returns the estimated size (steps) for this task.
317   * The value is, of course, an approximation, just so we will be able to
318   * give the user a feedback on the processing time. In this case, the
319   * value is calculated as the number of loops in the run() method.
320   */
  public long getSize()
    {
    // Return the estimated size for this task:
    return (long)maxIterations* // The maximum number of iterations times
      (
       (width*height*(numBands+numClusters*numBands))+ // Step 1 of run()
       (numClusters+width*height*numBands)+ // Step 2 of method run()
       (numClusters*numBands)+  // Step 2a of method run()
       (width*height*numBands) // Step 3 of the method run()
      );
    }

  /**
334   * This method returns a measure of the progress of the algorithm.
335   */
  public long getPosition()
    {
    return position;
    }

  /**
342   * This method returns true if the clustering has finished.
343   */
  public boolean isFinished()
    {
    return finished;
    }

  /**
350   * This method returns some information about the task (its progress).
351   * @return some information about the task
352   */
  public String getInfo()
```

```
    {
    if (!finished) return "Iteration "+(iteration+1)+" of "+maxIterations;
    else return "Clustering converged.";
    }
```

b. Εισαγωγή τιμών παραμέτρων στη διαδικασία

```
@DescribeProcess(title="Kmeans segmentation", description="A Kmean algorithm segments the input raster")
public class Kmeans implements GeoServerProcess{

        @DescribeResult(name="result", description="output raster")
        public RenderedImage execute(
                        @DescribeParameter(name = "coverage", description = "Input raster") GridCoverage2D coverage,
                        @DescribeParameter(name = "number of clusters", description = "The number of the clusters to be computed") Integer clu,
                        @DescribeParameter(name = "maximum iterations", description = "the maximum iterations to be made") Integer iter,
                        @DescribeParameter(name = "epsilon", description = "a small value used to verify if clustering has converged") Double ep
                        )throws ProcessException  {
                //get the rendered image
                RenderedImage re = coverage.getRenderedImage();
                //final int clusters = clu;
                //final int iterations = iter;
                //final double epsilon = ep;
                //wrap as planar image
                PlanarImage plim = PlanarImage.wrapRenderedImage(re);
                if(plim==null){
                        throw new ProcessException("Wrap Rendered Image Didn't worked");
                }
                //instantantiate the class
                KMeansImageClustering kmns = new KMeansImageClustering(plim,clu,iter,ep,'R');
                //run it!!
                kmns.run();
                //get the output as rendered image
                if(kmns.isFinished()== true){
            final RenderedImage rimage=kmns.getOutput();
            if(rimage!=null){
       return  rimage;
            }
            else
```

```
            {
            throw new ProcessException("null Image");
            }
            }
                else
                {
                        throw new ProcessException("something went wrong...");
                }

}
}
```

6. Αλγόριθμος Pansharpening

```
//@DescribeProcess(title="PanSharpeningAlgorithm", description="Use higher resolution panchromatic image to display RGB")
public class IHSPanSharpening {/*implements GeoServerProcess {

        @DescribeResult(name="result", description="Pansharpened raster")
        public RenderedImage execute(
                        @DescribeParameter(name = "RedBand", description = "Input raster") GridCoverage2D rcoverage,
                        @DescribeParameter(name = "GreenBand", description = "Input raster") GridCoverage2D gcoverage,
                        @DescribeParameter(name = "BlueBand", description = "Input raster") GridCoverage2D bcoverage,
                        @DescribeParameter(name = "PanchromaticBand", description = "Input raster") GridCoverage2D pancoverage
                        )throws ProcessException, FileNotFoundException {*/

public static void main(String[] args){
                // First we open the input images. We assume that each band is in a separate file.
                RenderedImage rendRed = JAI.create("fileload", args[0]);//rcoverage.getRenderedImage();
                RenderedImage rendGreen = JAI.create("fileload", args[1]);//gcoverage.getRenderedImage();
                RenderedImage rendBlue = JAI.create("fileload", args[2]);//bcoverage.getRenderedImage();
                RenderedImage rendPan = JAI.create("fileload", args[3]);//pancoverage.getRenderedImage();
                //wrap as PlanarImage
                PlanarImage iRed = PlanarImage.wrapRenderedImage(rendRed);
```

```
                    PlanarImage                    iGreen                    =
PlanarImage.wrapRenderedImage(rendGreen);
                    PlanarImage                    iBlue                     =
PlanarImage.wrapRenderedImage(rendBlue);
                    // Let's also load the pan image to use it as the new I band.
                    PlanarImage                    panImage                  =
PlanarImage.wrapRenderedImage(rendPan);
                    // We need to combine those bands into a single RGB image.
                    ParameterBlock pb = new ParameterBlock();
                    pb.addSource(iRed);
                    pb.addSource(iGreen);
                    pb.addSource(iBlue);
                    PlanarImage rgbImage = JAI.create("bandmerge", pb);
                    // Let's scale it since the pan image has the double of the
resolution.
                    pb = new ParameterBlock();
                    pb.addSource(rgbImage);
                    // Calculate the scale from the images' dimensions.
                    float scaleX = (1f*panImage.getWidth()/iRed.getWidth());
                    float scaleY = (1f*panImage.getHeight()/iRed.getHeight());
                    pb.add(scaleX);
                    pb.add(scaleY);
                    rgbImage = JAI.create("scale",pb);
                    // Now we can convert it to the IHS color space.
                    IHSColorSpace ihs = IHSColorSpace.getInstance();
                    ColorModel IHSColorModel =
                      new ComponentColorModel(ihs,
                                    new int []{8,8,8},
                                    false,false,
                                    Transparency.OPAQUE,
                                    DataBuffer.TYPE_BYTE);
                    pb = new ParameterBlock();
                    pb.addSource(rgbImage);
                    pb.add(IHSColorModel);
                    RenderedImage imageIHS  = JAI.create("colorconvert", pb);
                    // The image is in the IHS color space. Let's separate the I, H
and S band.
                    PlanarImage[] IHSBands = new PlanarImage[3];
                    for(int band=0;band<3;band++)
                      {
                      pb = new ParameterBlock();
                      pb.addSource(imageIHS);
                      pb.add(new int[]{band});
                      IHSBands[band] = JAI.create("bandselect",pb);
                      }
                    // Now we can compose the new IHS image.
```

```
                    // We must pass an instance of RenderingHint with the IHS
color model.
                    ImageLayout imageLayout = new ImageLayout();
                    imageLayout.setColorModel(IHSColorModel);

imageLayout.setSampleModel(imageIHS.getSampleModel());
                    RenderingHints rendHints = new
RenderingHints(JAI.KEY_IMAGE_LAYOUT,imageLayout);
                    pb = new ParameterBlock();
                    pb.addSource(panImage);
                    pb.addSource(IHSBands[1]);
                    pb.addSource(IHSBands[2]);
                    RenderedImage panSharpenedIHSImage =
JAI.create("bandmerge", pb, rendHints);
                    // Now we convert this image back to the RGB color space.
                    pb = new ParameterBlock();
                    pb.addSource(panSharpenedIHSImage);
                    pb.add(rgbImage.getColorModel()); // RGB color model
                    RenderedImage finalImage = JAI.create("colorconvert", pb);

                    //encode it!!!!!!
                    /*
                    TIFFEncodeParam tep = new TIFFEncodeParam();
                    tep.setWriteTiled(true);
                    tep.setTileSize(128,128); */

//JAI.create("filestore",finalImage,"pansharpened.png","PNG");
                    /*
                    String outputFile = null;
                    outputFile = finalImage;
                    FileOutputStream stream =
                            new FileOutputStream(outputFile);*/

                    //return finalImage;


                ParameterBlock pbConvert = new ParameterBlock();
                pbConvert.addSource(finalImage);
                pbConvert.add(DataBuffer.TYPE_BYTE);
                RenderedImage bOut = JAI.create("format", pbConvert);
                /*
                            int[] iArray = null;
                              int[]    dstpixels    =    bOut.getData().getPixels(0,
0,20,20, iArray);
                                    for(int i = 0; i < 20; i++) {
```

```
                                System.out.println(dstpixels[i]     +     "     "     +
dstpixels[i+1]+" "+ dstpixels[i+2]);}*/
                    //JAI.create("filestore",bOut,"pansharpened.png","PNG");

System.out.println(bOut.getHeight()*bOut.getWidth()*bOut.getSampleModel(
).getNumBands());
            }

}
```