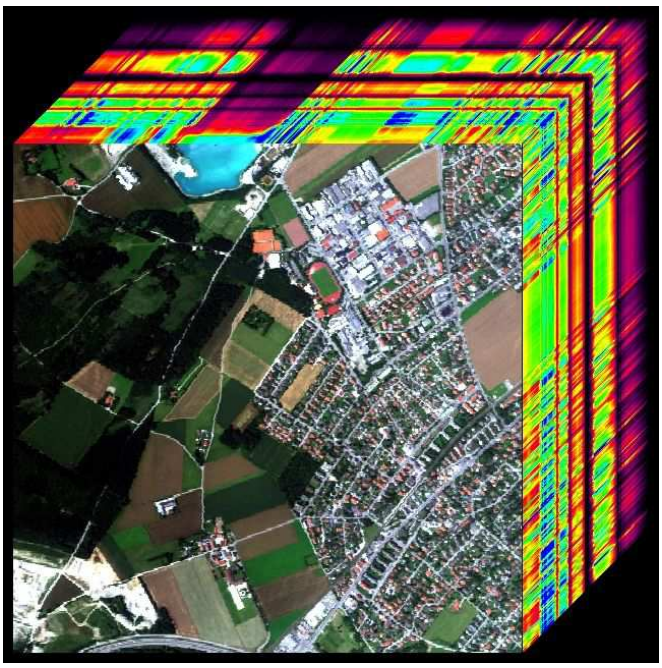
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ



Αξιοποίηση του Παράλληλου Προγραμματισμού σε γλώσσα C για την υλοποίηση αλγορίθμων για την Διαδικασία του Φασματικού Διαχωρισμού

Κέφαλος Δημήτριος

Επιβλέπουσα Καθηγήτρια: Βασιλεία Καραθανάση
Αναπλ. Καθηγήτρια ΕΜΠ

Αθήνα, Απρίλιος 2014

ΠΡΟΛΟΓΟΣ

Τα τελευταία χρόνια, η ανάπτυξη της τεχνολογίας και της επιστήμης, έχει καταστήσει δυνατή τη δημιουργία των υπερφασματικών δεκτών. Οι ίδιες οι ιδιότητες και τα πλεονεκτήματα των δεκτών αυτών καθιστούν τις παραδοσιακές μεθόδους της Τηλεπισκόπησης, αδύναμες στο να αξιοποιήσουν πλήρως την ποσότητα και την ποιότητα των υπερφασματικών δεδομένων. Έτσι, νέες μέθοδοι έχουν αναπτυχθεί και αναπτύσσονται συνεχώς για αυτόν ακριβώς τον σκοπό.

Το μεγάλο πλήθος των καναλιών των υπερφασματικών δεκτών και ο πολύ μεγάλος όγκος των δεδομένων που παράγουν, έχουν δημιουργήσει την ανάγκη ανάπτυξης αλγορίθμων και υλοποίησης προγραμμάτων για την επεξεργασία τους, ικανών να διαχειρίζονται και να αξιοποιούν το μεγάλο αυτό όγκο. Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη προγραμμάτων για την υλοποίηση του φασματικού διαχωρισμού των υπερφασματικών απεικονίσεων. Οι μέθοδοι του φασματικού διαχωρισμού οι οποίες επιλέχθηκαν να προγραμματιστούν έχουν αναπτυχθεί από τα μέλη του Εργαστηρίου Τηλεπισκόπησης του Ε.Μ.Π. Τα προγράμματα τα οποία αναπτύχθηκαν έχουν υλοποιηθεί σε γλώσσα προγραμματισμού C επειδή ως γλώσσα χαμηλού επιπέδου επιτρέπει την επεξεργασία μεγάλου όγκου δεδομένων. Παράλληλα η αξιοποίηση του παράλληλου προγραμματισμού σε μια χαμηλού επιπέδου γλώσσα προγραμματισμού όπως η C, επιτρέπει την βελτιστοποίηση του υπολογιστικού χρόνου των προγραμμάτων.

Για την εκπόνηση της εργασίας αυτής θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην πραγματοποίησή της. Αρχικά, οφείλω να ευχαριστήσω την Αναπληρώτρια Καθηγήτρια Βασιλεία Καραθανάση, για την ανάθεση αυτού του ενδιαφέροντος θέματος και τη βοήθεια της σε όλα τα στάδια υλοποίησής του. Στη συνέχεια, οφείλω να ευχαριστήσω τον Χρήστο Ιωσηφίδη, Ε.ΔΙ.Π., για την πολύτιμη βοήθεια του στις προκλήσεις του προγραμματισμού, και το Δρ. Πολυχρόνη Κολοκούση, για τη βοήθειά του, κυρίως κατά την διαδικασία εξοικείωσης μου με τον προγραμματισμό και όχι μόνο. Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην Δρ. Χαρούλα Ανδρέου, για την πολύτιμη βοήθεια της σε όλη τη διάρκεια της εκπόνησης της συγκεκριμένης διπλωματικής εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

ΠΕΡΙΛΗΨΗ

Στο επιστημονικό πεδίο της ψηφιακής τηλεπισκόπησης, τα δεδομένα των πολυφασματικών δεκτών υπάρχουν εδώ και αρκετά χρόνια. Με την ανάπτυξη όμως, της τεχνολογίας των δεκτών και την ραγδαία αύξηση των δυνατοτήτων των ηλεκτρονικών υπολογιστών, έγινε δυνατή η κατασκευή των υπερφασματικών δεκτών και η επεξεργασία των δεδομένων τους. Έτσι, προέκυψε η ανάγκη δημιουργίας μεθόδων και αλγορίθμων για την αξιοποίηση των πλεονεκτημάτων των δεκτών αυτών, που δεν ήταν δυνατό να αξιοποιήσουν οι υπάρχουσες μέθοδοι των πολυφασματικών δεδομένων.

Μία κατηγορία τέτοιων μεθόδων, αποτελούν αυτές που σκοπεύουν στην διαδικασία του φασματικού διαχωρισμού. Με τον όρο φασματικός διαχωρισμός μιας υπερφασματικής απεικόνισης εννοούμε την διαδικασία εύρεσης σε επίπεδο εικονοστοιχείου των υλικών / αντικειμένων που συνθέτουν την φασματική υπογραφή του εικονοστοιχείου. Η διαδικασία αυτή επιτυγχάνεται με τη χρήση μιας σειράς αλγορίθμων, που ο καθένας παρέχει τα δεδομένα που χρειάζεται ο επόμενος. Στην εργασία αυτή αξιοποιήθηκαν οι τεχνικές παράλληλου προγραμματισμού σε γλώσσα C, για τον προγραμματισμό των αλγορίθμων που αναπτύχθηκαν στο Εργαστήριο Τηλεπισκόπησης του Ε.Μ.Π. για την υλοποίηση του φασματικού διαχωρισμού.

Η διαδικασία του φασματικού διαχωρισμού ξεκινά με την εκτίμηση του φασματικού υπόχωρου του σήματος που είναι συνυφασμένος με το πλήθος των υλικών / αντικειμένων (καθαρών στόχων) που εμφανίζονται στην υπερφασματική απεικόνιση. Για την υλοποίηση της διαδικασίας αυτής χρειάζεται να μετατραπεί η απεικόνιση σε απεικόνιση λευκού θορύβου και να ταξινομηθούν τα κανάλια της ανάλογα με το ποσοστό της πληροφορίας που περιέχουν. Για αυτό το σκοπό υλοποιήθηκε αρχικά πρόγραμμα που αφορά στο μετασχηματισμό ελαχιστοποίησης του θορύβου (MNF). (Green et al. 1988) Ο αλγόριθμος εκτίμησης του φασματικού υπόχωρου του σήματος που προγραμματίστηκε σε αυτή την εργασία, ο οποίος βασίζεται στη μέθοδο Outlier Detection Method (ODM) (Andreou C. and Karathanassi V., 2012), υπολογίζει τα στατιστικά των MNF καναλιών, με στόχο να προσδιορίσει τον αριθμό των διαστάσεων που χρειάζονται για να περιγραφεί η πληροφορία της εικόνας.

Το αποτέλεσμα της μεθόδου εκτίμησης του φασματικού υπόχωρου του σήματος και το γεγονός ότι ο αριθμός των καθαρών στόχων ισούται με τον

αριθμό της φασματικής διάστασης που περιέχει την πληροφορία της εικόνας αυξανόμενη κατά ένα, αξιοποιεί στη συνέχεια η μέθοδος εξαγωγής των φασματικών υπογραφών των καθαρών στόχων. Στην εργασία αυτή υλοποιήθηκαν προγράμματα για δύο μεθόδους εξαγωγής καθαρών στόχων, για τη Simple Endmember Extraction (SEE) και την SEE - Enhanced (E-SEE). (Andreou C. and Karathanassi V. 2011)

Τέλος, υλοποιήθηκε πρόγραμμα εκτίμησης της αφθονίας του κάθε καθαρού στόχου, σε επίπεδο εικονοστοιχείου. Ο αλγόριθμος που υλοποιήθηκε, είναι ένας κλασσικός αλγόριθμος επίλυσης γραμμικών συστημάτων με ελαχιστοτετραγωνικές μεθόδους, ο οποίος χρησιμοποιεί ως εξισώσεις παρατήρησης τις φασματικές υπογραφές των καθαρών στόχων, για να εκτιμήσει τους συντελεστές αφθονίας, του κάθε καθαρού στόχου, σε κάθε εικονοστοιχείο.

ABSTRACT

In the scientific field of digital remote sensing, multispectral data have been appeared several years ago. With the development of sensor technology and the growth of computer capabilities, the development of hyperspectral sensors and the processing of their data became a challenge for the scientific community. The need of new methods and algorithms for the exploitation of hyperspectral datasets have been raised since the multispectral algorithms were unable to process them effectively. Among the new methods, methods that deal with the spectral unmixing have been developed.

Spectral unmixing is the procedure of finding the materials/objects that compose the spectral signature of each pixel in the image. This procedure is accomplished with the sequential use of a series of algorithms. In this diploma thesis, techniques for parallel processing in C programming language have been used and evaluated for the implementation of the spectral unmixing procedure. The spectral unmixing methods, programmed in this diploma thesis, have been developed by the members of the Remote Sensing Laboratory of the N.T.U.A.

The spectral unmixing procedure starts with the estimation of the signal dimensionality, which is closely related to the number of materials/objects (endmembers) appeared in the hyperspectral image. For the implementation of this procedure the image must be transformed into a white noise image, and its bands must be sorted according to the percentage of information they contain. For this reason, a program which applies the Minimum Noise Fraction Transformation (MNF) (Green et al. 1988) has been written. The method that estimates the number of endmembers, used in this diploma thesis, is the Outlier Detection Method (ODM) (Andreou C. and Karathanassi V., 2012). This method is based on the MNF band statistics, in order to define the dimensionality of the image.

The next step of the unmixing procedure is the extraction of the endmembers. The endmember extraction method uses the result of the ODM algorithm and the fact that the dimensionality of the hyperspectral image is equal to the number of its endmembers minus one. In this diploma thesis programs for two endmember extraction methods have been developed: the Simple Endmember Extraction Method (SEE), and the SEE – Enhanced method (E-SEE). (Andreou C. and Karathanassi V. 2011).

The last step of the unmixing procedure is the estimation of the abundance fractions. A pixel level abundance fraction estimation program has been developed based on the linear mixing model. The algorithm solves a linear system using least squares methods, in order to estimate the abundance fraction of each endmember in a pixel basis.

1. ΕΙΣΑΓΩΓΗ

Οι μέθοδοι της υπερφασματικής τηλεπισκόπησης αποτελούν ένα σημαντικό εργαλείο για την εξέταση και παρακολούθηση στην πορεία του χρόνου περιβαλλοντολογικών φαινομένων, είτε αυτά σχετίζονται με τις δραστηριότητες του ανθρώπου, είτε όχι. (Ανδρέου Χ. 2008) Το πρόβλημα του φασματικού διαχωρισμού (spectral unmixing) δηλαδή το τι υλικά ή/και αντικείμενα και με ποιο ποσοστό το κάθε ένα συνεισφέρουν στη διαμόρφωση της φασματικής τιμής ενός εικονοστοιχείου που καταγράφεται από έναν δέκτη, αποτελεί ένα πολύ σημαντικό κεφάλαιο της υπερφασματικής τηλεπισκόπησης, καθώς η επίλυση του, με αυτοματοποιημένες μεθόδους, μπορεί να έχει πολύ σημαντικές επιπτώσεις στον τρόπο που ερμηνεύονται τα υπερφασματικά δεδομένα και στην ποσότητα και την ποιότητα των πληροφοριών που μπορούν να δώσουν. Με τη βελτίωση και τελειοποίηση της διαδικασίας του φασματικού διαχωρισμού, μπορούν να αξιοποιηθούν σε μεγαλύτερο ποσοστό οι δυνατότητες των υπερφασματικών δεδομένων και να ξεπεραστούν αδυναμίες των πολυφασματικών δεδομένων.

Η εξαγωγή των καθαρών στόχων δηλαδή των υλικών/αντικειμένων που συμβάλλουν στη διαμόρφωση της φασματικής υπογραφής του κάθε εικονοστοιχείου, απασχολεί τα τελευταία χρόνια όλο και περισσότερους ερευνητές. Το ενδιαφέρον είναι μεγάλο καθώς η ανίχνευση καθαρών στόχων ανά εικονοστοιχείο αποτελεί το πρωταρχικό και πιο σημαντικό στάδιο από το οποίο εξαρτάται η ακρίβεια των μετέπειτα επεξεργασιών της υπερφασματικής απεικόνισης όπως είναι η ταξινόμηση και η ανίχνευση συγκεκριμένων αντικειμένων. Η εξαγωγή των καθαρών στόχων αλλά και η εκτίμηση του ποσοστού αφθονίας του κάθε καθαρού στόχου σε επίπεδο εικονοστοιχείου, αποτελούν αντικείμενο έρευνας των τελευταίων χρόνων καθώς επιτρέπουν την παραγωγή χαρτών αφθονίας του κάθε υλικού / αντικειμένου και την ποσοτικοποίηση των παραμέτρων που σχετίζονται με αυτά.

Η ίδια η φύση των υπερφασματικών δεδομένων ευνοεί, για τον προγραμματισμό των μεθόδων και των αλγορίθμων της επεξεργασίας τους, γλώσσες χαμηλού επιπέδου, όπως η C. Το μεγάλο πλεονέκτημα της συγκεκριμένης γλώσσας που είναι η ταχύτητα της, επιτρέπει την επεξεργασία υπερφασματικών δεδομένων, τα οποία λόγω του μεγάλου μεγέθους τους καθιστούν την χρήση γλωσσών πιο υψηλού επιπέδου, δύσκολη ή ακατάλληλη. Ειδικότερα στην περίπτωση της χρήσης τεχνικών παράλληλου προγραμματισμού σε C τα αποτελέσματα είναι ακόμα καλύτερα, ανάλογα βέβαια με τα τεχνικά χαρακτηριστικά του υπολογιστή.

Στην παρούσα διπλωματική εργασία προγραμματίστηκε μια σειρά αλγορίθμων με στόχο τον φασματικό διαχωρισμό. Οι περισσότεροι από τους αλγορίθμους αυτούς είναι προϊόντα έρευνας του Εργαστηρίου Τηλεπισκόπησης του Ε.Μ.Π. Η διαδικασία του φασματικού διαχωρισμού ξεκινά με την εκτίμηση του φασματικού υπόχωρου του σήματος. Η μέθοδος που υιοθετήθηκε για το στάδιο αυτό ήταν η Outlier Detection Method (ODM). Λόγω του μεγάλου αριθμού των καναλιών μια υπερφασματικής εικόνας, η μέθοδος ODM θεωρεί ως στατιστικό δεδομένο το θόρυβο και σαν ανωμαλία του δεδομένου αυτού την πληροφορία. Έτσι υπολογίζοντας και ταξινομώντας, με κατάλληλο τρόπο, τα στατιστικά μεγέθη των απεικονίσεων μπορεί να εκτιμήσει τον αριθμό των καθαρών στόχων.

Για να γίνει η εκτίμηση του φασματικού υπόχωρου του σήματος, σε πραγματικά υπερφασματικά δεδομένα, πρέπει να εκτιμηθεί αρχικά ο θόρυβος της απεικόνισης και στη συνέχεια να μετατραπεί σε λευκό θόρυβο. Αυτό επιτυγχάνεται με χρήση του μετασχηματισμού ελαχιστοποίησης του θορύβου (MNF). Ο αλγόριθμος αυτός, υλοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, χρησιμοποιώντας δύο μεθόδους εκτίμησης του θορύβου, τη μέθοδο του εγγύτερου γείτονα, και τη μέθοδο της πολλαπλής παλινδρόμησης.

Για τον εντοπισμό και την εξαγωγή των φασματικών υπογραφών των καθαρών στόχων, υλοποιήθηκε ο αλγόριθμος Simple Endmember Extraction (SEE) και ο Enhanced - Simple Endmember Extraction (E-SEE). Οι αλγόριθμοι αυτοί, παίρνοντας ως δεδομένο τον αριθμό των καθαρών στόχων της εικόνας, όπως τον εκτίμησε η μέθοδος ODM, εντοπίζουν τους καθαρούς στόχους αξιοποιώντας την ιδιότητα των καθαρών στόχων να βρίσκονται στις κορυφές του διαγράμματος διασποράς των πρώτων κυρίων συνιστωσών. Επιπλέον, ο αλγόριθμος SEE-E, χρησιμοποιεί μια επιπλέον παρατήρηση και τεχνική στροφής των αξόνων για να αποκαλύψει κρυμμένους καθαρούς στόχους οι οποίοι βρίσκονται μεν στις κορυφές του διαγράμματος διασποράς, αλλά είναι κοντά στη μέση τιμή και γι αυτό δεν αποκαλύπτονται.

Τη διαδικασία του φασματικού διαχωρισμού συμπληρώνει η μέθοδος εκτίμησης της αφθονίας. Η αλγόριθμος που υλοποιήθηκε προγραμματιστικά, παίρνει ως εξισώσεις παρατήρησης τις φασματικές υπογραφές των καθαρών στόχων και επιλύει ένα γραμμικό σύστημα για κάθε εικονοστοιχείο. Έτσι, προκύπτει μια εικόνα με τους συντελεστές αφθονίας για κάθε καθαρό στόχο, σε κάθε εικονοστοιχείο.

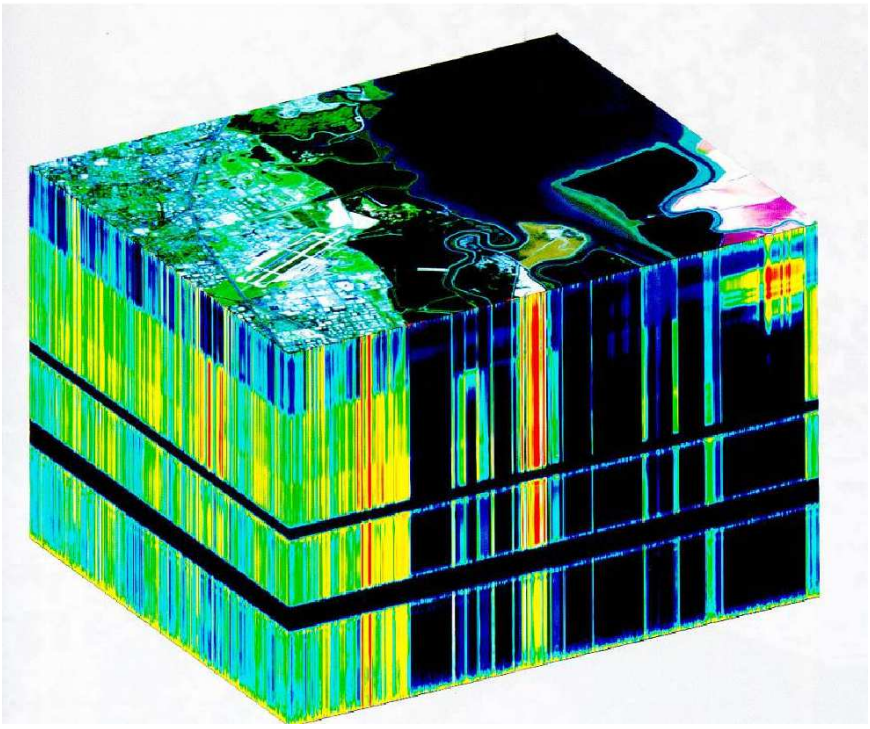
2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΤΗΣ ΥΠΕΡΦΑΣΜΑΤΙΚΗΣ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

2.1. Γενικά για την Υπερφασματική Τηλεπισκόπηση

Τηλεπισκόπηση καλείται η απόκτηση πληροφοριών εξ αποστάσεως –με δορυφορικό, αερομεταφερόμενο ή επίγειο αισθητήρα- για ένα αντικείμενο ή για το περιβάλλον του με βάση την ηλεκτρομαγνητική ακτινοβολία που αυτό ανακλά ή/και εκπέμπει (Goetz et al, 1985). Η ένταση και τα χαρακτηριστικά της καταγραφόμενης ηλεκτρομαγνητικής ακτινοβολίας εξαρτώνται από την ένταση και τα χαρακτηριστικά της πηγής της (στην περίπτωση των ηλεκτρο-οπτικών δεκτών πηγή είναι ο ήλιος), από το ανάγλυφο της γήινης επιφάνειας, από τη χημική σύσταση των αντικειμένων και τέλος, από το βαθμό απορρόφησης της ακτινοβολίας από την ατμόσφαιρα (Schott, 1997).

Στην Υπερφασματική Τηλεπισκόπηση, το ηλεκτρομαγνητικό φάσμα χωρίζεται σε εκατοντάδες στενές, παρακείμενες φασματικές ζώνες οι οποίες είναι επαρκείς ώστε να αποτυπώνονται με λεπτομέρεια όλες οι φασματικές υπογραφές των υλικών τα οποία απεικονίζονται σε μία εικόνα. Η Υπερφασματική Τηλεπισκόπηση συντελεί στην αναγνώριση επιφανειών διαφορετικής σύστασης υλικών λόγω της μοναδικής φασματικής υπογραφής την οποία αυτά διαθέτουν.

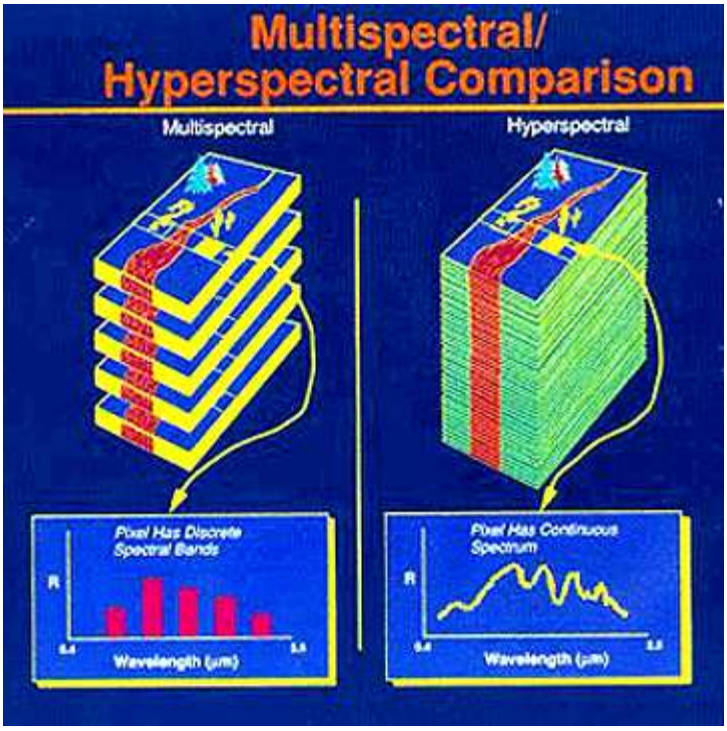
Τυπικά, μία υπερφασματική εικόνα έχει τη μορφή ενός κύβου όπου οι δύο διαστάσεις αφορούν στη χωρική πληροφορία και η τρίτη στη φασματική. Κατά συνέπεια κάθε εικονοστοιχείο μπορεί να θεωρηθεί ως διάνυσμα στήλης του οποίου οι επιμέρους τιμές αντιπροσωπεύουν το συντελεστή ανακλαστικότητας του εικονοστοιχείου σε συγκεκριμένα κανάλια (Keshava και Mustard, 2002, Manolakis et al., 2003).



Εικόνα 2.1. Κύβος υπερφασματικών δεδομένων
Πηγή : Wikipedia

Ένα κομμάτι της τηλεπισκόπησης αποτελεί η αξιοποίηση των πλεονεκτημάτων των υπερφασματικών δεκτών. Οι παθητικοί ηλεκτρο-οπτικοί δέκτες χωρίζονται σε πολυφασματικούς και υπερφασματικούς, με την κύρια διαφορά τους να είναι ο αριθμός των καναλιών της εικόνας, ο οποίος στους πολυφασματικούς δέκτες κυμαίνεται από λιγότερο από δέκα μέχρι μερικές δεκάδες κανάλια, ενώ στους υπερφασματικούς είναι της τάξης μερικών εκατοντάδων καναλιών.

Μια άλλη σημαντική διαφορά των πολυφασματικών και των υπερφασματικών δεκτών είναι ότι οι πρώτοι είναι ρυθμισμένοι να λαμβάνουν δεδομένα σε λίγα και «φαρδιά» κομμάτια του ηλεκτρομαγνητικού φάσματος ενώ οι δεύτεροι σε πολλά και λεπτά. Αυτό καθιστά τους πολυφασματικούς δέκτες ακατάλληλους, για την περιγραφή της φασματικής υπογραφής των αντικειμένων που απεικονίζουν. Η μεγάλη ραδιομετρική διακριτική ικανότητα των υπερφασματικών δεκτών, τους καθιστά ικανό εργαλείο για την αποτύπωση των φασματικών υπογραφών που απεικονίζουν, με σημαντική ακρίβεια.

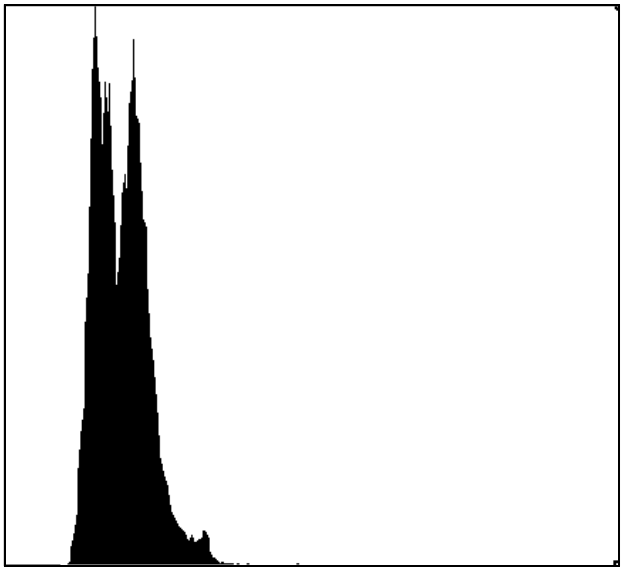


Εικόνα 2.2. Σύγκριση πολυφασματικών και υπερφασματικών δεκτών
Πηγή: Wikipedia

2.2. Γενικές έννοιες για τις Τηλεπισκοπικές Εικόνες

2.2.1. Ιστόγραμμα

Μια εικόνα αποθηκεύεται στο υπολογιστή ως τιμές των εικονοστοιχείων της σε καθένα από τα κανάλια της. Το ιστόγραμμα ενός καναλιού είναι ένα διάγραμμα με οριζόντιο άξονα τις τιμές ανακλαστικότητας που λαμβάνει ο δέκτης και κατακόρυφο τον αριθμό των εικονοστοιχείων που παίρνουν αυτή την τιμή. Το ιστόγραμμα είναι ένα πολύ χρήσιμο εργαλείο της τηλεπισκόπησης είτε αυτό χρησιμοποιείται μόνο για την ενίσχυση της εικόνας ώστε να είναι πιο εύκολη η ανίχνευση χαρακτηριστικών από τον χρήστη με μεθόδους παραδοσιακής φωτοερμηνείας, είτε για περαιτέρω ανάλυση και εξαγωγή συμπερασμάτων για υπολογιστικές μεθόδους τηλεπισκόπησης.



Εικόνα 2.3. Παράδειγμα ιστογράμματος εικόνας
Πηγή : ERMapper

2.2.2. Στατιστικά μεγέθη

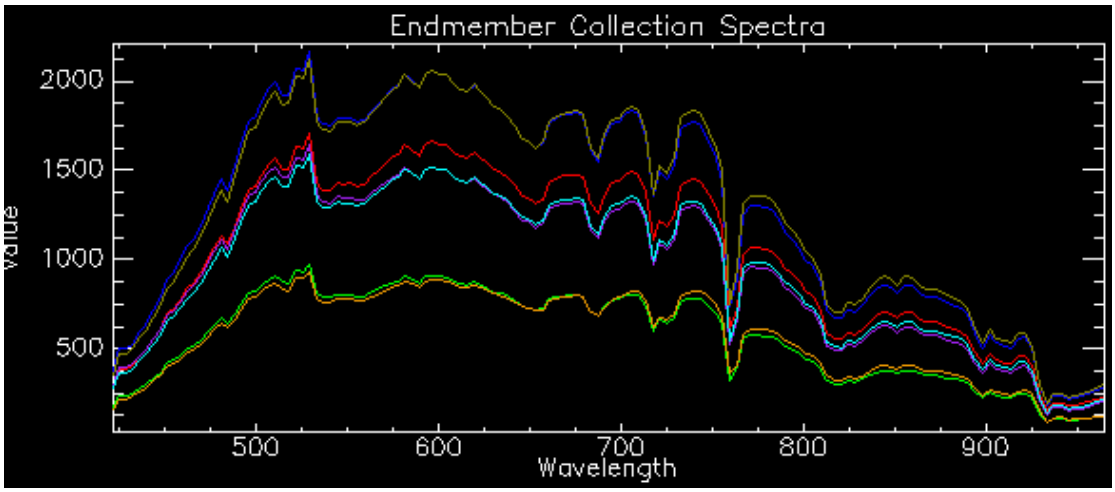
Στα πολυφασματικά, πόσο μάλλον και στα υπερφασματικά δεδομένα, χρησιμοποιούνται στατιστικές μέθοδοι για την εξαγωγή συμπερασμάτων και την υλοποίηση των αλγορίθμων. Η συσχέτιση των δύο αυτών επιστημονικών πεδίων (της τηλεπισκόπησης και της στατιστικής) γίνεται με τη θεώρηση ότι ένα εικονοστοιχείο είναι ένα στατιστικό μέγεθος και η τιμή του σε καθένα από τα κανάλια της εικόνας είναι μια παρατήρηση του μεγέθους αυτού. Με αυτή τη λογική εξάγονται οι πίνακες μεταβλητότητας-συμμεταβλητότητας των καναλιών και άλλα στατιστικά μεγέθη που χρησιμοποιούνται σε μεγάλο βαθμό στη ψηφιακή τηλεπισκόπηση και ακόμα περισσότερο στις μεθόδους επεξεργασίας υπερφασματικών δεδομένων.

2.2.3. Πρότυπο - Φασματική Υπογραφή

Αναγνώριση προτύπων μπορεί να οριστεί ως η κατηγοριοποίηση κάποιων δεδομένων (αντικειμένων, προτύπων) σε καθορισμένες και αναγνωρίσιμες τάξεις μέσω της εξαγωγής σημαντικών και μοναδιαίων χαρακτηριστικών ιδιοτήτων τους (Αργιαλάς 1998). Πρότυπο στην τηλεπισκόπηση μπορεί να οριστεί είτε ο τρόπος εμφάνισης ενός χαρακτηριστικού ή αντικειμένου στην κλασική φωτοερμηνεία είτε μια ή περισσότερες μαθηματικές περιγραφές ενός αντικειμένου στην ψηφιακή Τηλεπισκόπηση.

Στην υπερφασματική τηλεπισκόπηση, λόγω του πολύ μεγαλύτερου όγκου πληροφορίας, πλήθους και ιδιοτήτων των καναλιών, το πρότυπο για την

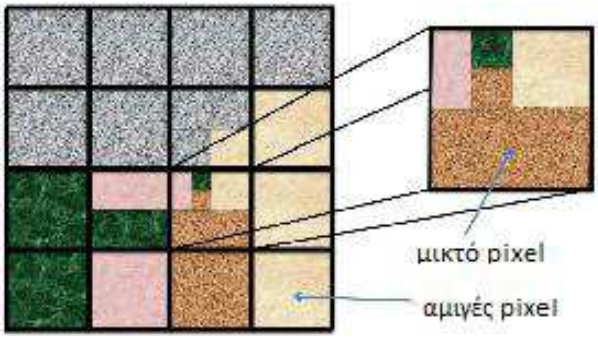
αναγνώριση αντικειμένων είναι η φασματική υπογραφή τους. Σαν φασματική υπογραφή μπορούμε να ορίσουμε το διάγραμμα των τιμών ανακλαστικότητας ενός αντικειμένου στα μήκη κύματος του οπτικού και υπέρυθρου τμήματος της ηλεκτρομαγνητικής ακτινοβολίας. Αυτό το αποτέλεσμα μπορούμε να το δούμε εάν έχουμε δεδομένα από υπερφασματικούς δέκτες, καθώς αυτοί μπορούν να περιγράψουν ένα αντικείμενο σε ευρύ και συνεχόμενο κομμάτι του ηλεκτρομαγνητικού φάσματος και με ικανή φασματική διακριτική ικανότητα, όπως αναφέρθηκε και παραπάνω.



Εικόνα 2.4. Παράδειγμα φασματικών υπογραφών
Πηγή : Ανδρέου Χ. Διπλωματική Εργασία, 2008

2.2.4. Καθαρός στόχος

Λόγω της παρουσίας διαφορετικών υλικών τα οποία συνθέτουν τις καλύψεις / χρήσεις γης, είναι δυνατόν σε ένα εικονοστοιχείο ακόμη και υψηλής χωρικής ανάλυσης να υπάρχουν φασματικές υπογραφές οι οποίες να ανήκουν σε περισσότερα από ένα υλικά. Τότε, το εικονοστοιχείο αυτό λέγεται “μικτό” (mixed pixel) σε αντίθεση με το “αμιγές” εικονοστοιχείο (pure pixel) στο οποίο περιέχεται η φασματική υπογραφή ενός μοναδικού υλικού (Jiang, 2002, Lentilucci, 2001) Οι καθαρές φασματικές υπογραφές της απεικόνισης οι οποίες ανήκουν σε ένα μοναδικό κάθε φορά υλικό/στόχο, καλούνται καθαροί στόχοι (endmembers) (Zare, 2008). Το σύνολο των μη επιθυμητών φασματικών υπογραφών καλείται υπόβαθρο (background) (Harsanyi and Chang,1994).

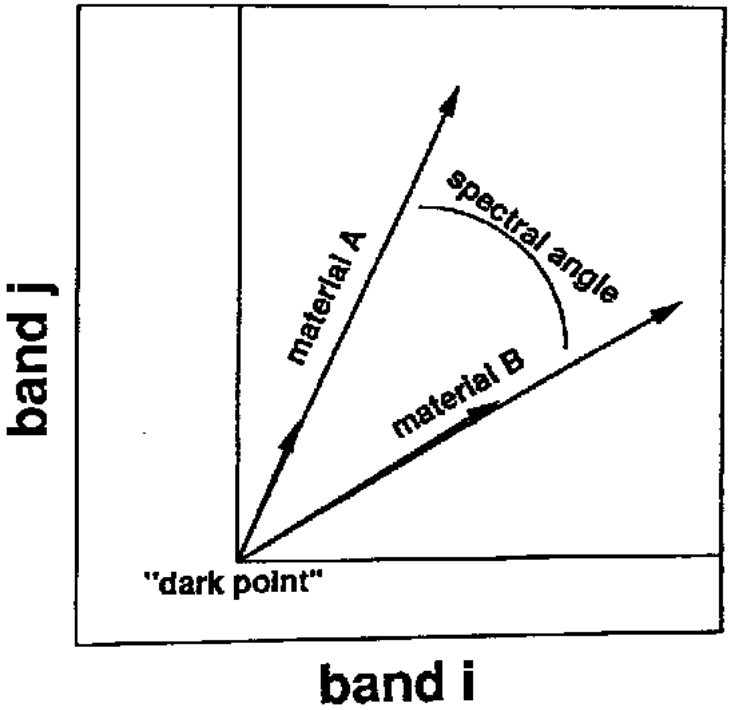


Εικόνα 2.5. Αμιγές και μικτό εικονοστοιχείο
Πηγή:www.orfeo-toolbox.org

Λόγω της υψηλής φασματικής ανάλυσης την οποία παρέχουν οι υπερφασματικοί αισθητήρες είναι δυνατή η ανίχνευση καθαρών στόχων οι οποίοι περιέχονται σε μικτά εικονοστοιχεία. Για να είναι επιτυχής η ανάλυση και η επεξεργασία των πληροφοριών οι οποίες περιέχονται σε μία υπερφασματική απεικόνιση, λόγω της ύπαρξης μικτών εικονοστοιχείων, είναι απαραίτητο να γίνει φασματικός διαχωρισμός (spectral unmixing problem) στην απεικόνιση, ο οποίος έχει ως αποτέλεσμα την αποσύνθεση των εικονοστοιχείων στους επιμέρους καθαρούς στόχους και στα αντίστοιχα ποσοστά συμμετοχής τους (Jiang, 2002). Έτσι, δουλεύοντας με τους όρους μικτό και αμιγές εικονοστοιχείο, καθώς και με τα ποσοστά συμμετοχής ή τους συντελεστές αφθονίας των καθαρών στόχων σε ένα εικονοστοιχείο, λέμε ότι δουλεύουμε σε επίπεδο υπό-εικονοστοιχείου.

2.2.5. Φασματικές γωνίες

Δύο θεματικές κατηγορίες διακρίνονται μεταξύ τους με διάφορους τρόπους, όπως η απόσταση των κέντρων των νεφών τους. Ένας άλλος τρόπος είναι ο διαχωρισμός με βάση την φασματική γωνία δυο κατηγοριών. Η φασματική γωνία είναι η γωνία που σχηματίζουν οι δυο κατηγορίες με το κέντρο του n-διάστατου χώρου. Η μέθοδος διαχωρισμού αυτή λαμβάνει υπόψη της την διεύθυνση του κάθε νέφους, και όχι την απόσταση από το κέντρο του χώρου.



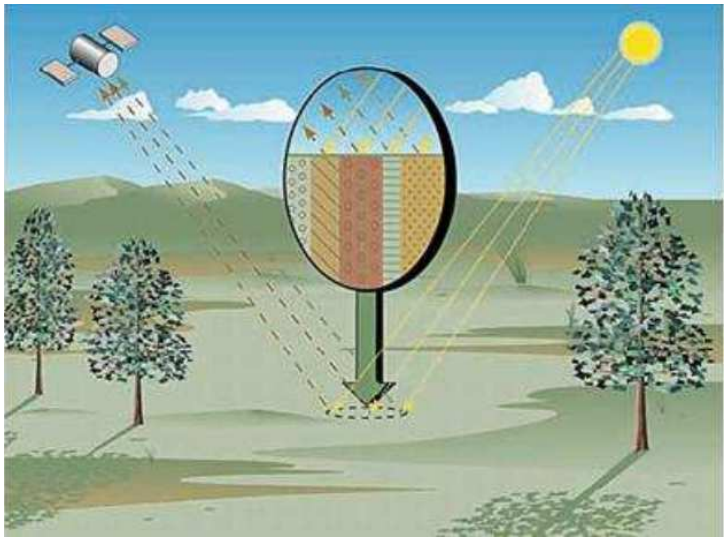
Εικόνα 2.6. Σχηματική απεικόνιση φασματικών εικόνων
Πηγή : Online ENVI Tutorial #14

2.2.6. Θόρυβος-λευκός θόρυβος

Μια δορυφορική εικόνα περιέχει δυο συνιστώσες, το σήμα που έλαβε ο δέκτης από το έδαφος και τον θόρυβο. Ο θόρυβος προέρχεται από διάφορες πηγές, όπως σφάλματα του δέκτη ή θόρυβο που εισάγει η ατμόσφαιρα. Ένας από τους σκοπούς της τηλεπισκόπησης είναι και η δημιουργία μεθόδων για την μείωση του θορύβου. Αυτό γίνεται με διάφορους τρόπους, όπως με τη χρήση φίλτρων ή μετασχηματισμών. Με τον όρο λευκό θόρυβο εννοούμε τον θόρυβο εκείνο που έχει μέση τιμή μηδέν, πεπερασμένη τυπική απόκλιση και η τιμή του θορύβου σε κάθε εικονοστοιχείο είναι ανεξάρτητη από τις τιμές των άλλων εικονοστοιχείων. Αν ο λευκός θόρυβος ακολουθεί την κανονική κατανομή ονομάζεται λευκός γκαουσιανός θόρυβος. (Green A. et al 1988)

2.2.7. Γραμμικό και μη-γραμμικό μοντέλο ανάμειξης

Υπάρχουν δύο είδη μοντέλων ανάμειξης των καθαρών στόχων σε κάθε εικόνα, το γραμμικό και το μη-γραμμικό. Σύμφωνα με το γραμμικό μοντέλο, το φως αλληλεπιδρά με ένα υλικό κάθε φορά. Έτσι, κάθε υλικό συνεισφέρει στην τιμή του αντίστοιχου εικονοστοιχείου, η οποία προκύπτει από το άθροισμα των συνεισφορών των υλικών.



Εικόνα 2.7. Γραμμικό μοντέλο ανάμειξης
Πηγή: Keshva and Mustard "Spectral Unmixing"

Το γραμμικό μοντέλο περιγράφεται από την σχέση:

$$r_i = \sum_{j=1}^{n} (e_{j,i} * a_j) + \theta$$

όπου r_i η τιμή ανακλαστικότητας του μικτού εικονοστοιχείου στο κανάλι i,

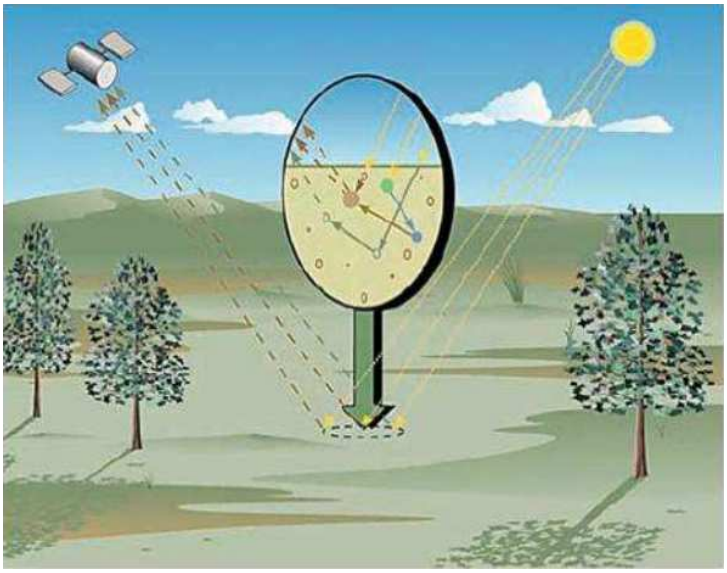
$e_{j,i}$ η τιμή ανακλαστικότητας του καθαρού στόχου j στο κανάλι i,

a_j το ποσοστό συμμετοχής του καθαρού στόχου j και

θ ο θόρυβος

Στις υπερφασματικές απεικονίσεις κάθε εικονοστοιχείο μπορεί να αναπαρασταθεί ως σημείο στο n-διάστατο χώρο, όπου n ο αριθμός των καναλιών, του οποίου οι συντεταγμένες δίνονται από τις διαφορετικές φασματικές τιμές σε κάθε κανάλι. Αν παραλείψουμε τον θόρυβο από την παραπάνω σχέση, οι γραμμικοί συνδυασμοί που προκύπτουν, σχηματίζουν ένα simplex, του οποίου οι κορυφές, αντιστοιχούν στους καθαρούς στόχους.

Το μη-γραμμικό μοντέλο θεωρεί πιθανές αλληλεπιδράσεις μεταξύ των υλικών και εισάγει μη γραμμικούς όρους στο σύστημα.



Εικόνα 2.8. Μη γραμμικό μοντέλο ανάμειξης
Πηγή: Keshva and Mustard "Spectral Unmixing"

Η σχέση που εκφράζει το μη-γραμμικό μοντέλο είναι η εξής:

$$r_i = f(e_{j,i}, a_j) + \theta$$

όπου r_i η τιμή ανακλαστικότητας του μικτού εικονοστοιχείου στο κανάλι i,

$f(e_{j,i}, a_j)$ μια μη γραμμική σχέση μεταξύ της τιμής ανακλαστικότητας του καθαρού στόχου j στο κανάλι i , $e_{j,i}$ και του ποσοστού συμμετοχής του καθαρού στόχου j, a_j και

θ ο θόρυβος

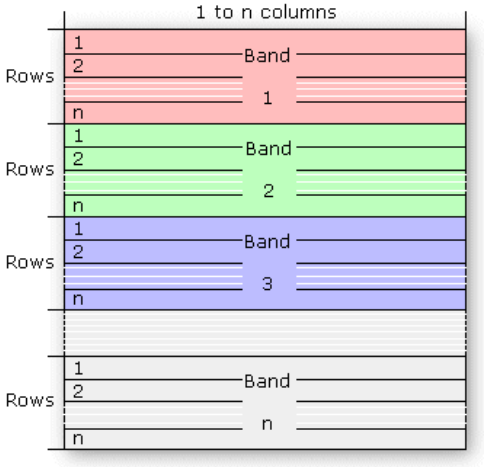
Παρότι και τα δύο μοντέλα βασίζονται σε ένα σκεπτικό που ισχύει στην πραγματικότητα και δίνουν ικανοποιητικά αποτελέσματα, ανάλογα με την χωρική διακριτική ικανότητα, το μη γραμμικό μοντέλο είναι ιδιαίτερα περίπλοκο και στην πράξη δεν χρησιμοποιείται όσο το γραμμικό.

2.3. Δομή των Δεδομένων της Ψηφιακής Τηλεπισκόπησης

Το κεφάλαιο αυτό παρέχει στον αναγνώστη απαραίτητες πληροφορίες ώστε να μπορέσει να κατανοήσει τον κώδικα του προγράμματος που θα αναλυθεί παρακάτω. Όπως είναι γνωστό, μια ψηφιακή εικόνα αποθηκεύεται στον υπολογιστή σαν ένας πινάκας ν γραμμών επί μ στηλών. Το μοντέλο αυτό όμως περιγράφει τον τρόπο αποθήκευσης μια εικόνας που περιέχει ένα

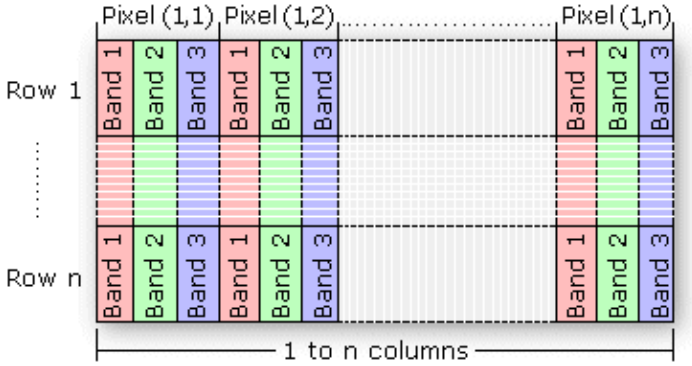
κανάλι. Για την εισαγωγή περισσότερων καναλιών υπάρχουν τρεις τρόποι οργάνωσης των δεδομένων: ο BSQ (band sequential), ο BIP (band interleaved by pixel) και ο συνηθέστερος BIL (band interleaved by line) οι οποίοι αναλύονται παρακάτω.

Οι εικόνες που χρησιμοποιούν την τυποποίηση BSQ είναι αποθηκευμένες σε ένα αρχείο που περιέχει ολόκληρη την εικόνα τον ένα κανάλι μετά το άλλο, όπως φαίνεται στην εικόνα.



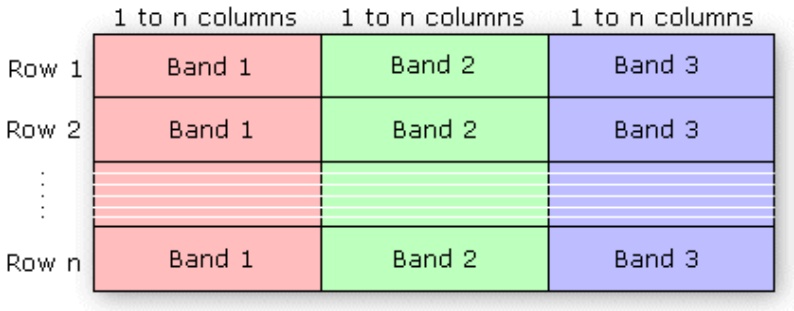
Εικόνα 2.9. Τυποποίηση αρχείου τύπου BSQ
Πηγή : esri.com

Οι εικόνες που αποθηκεύονται με την τυποποίηση BIP είναι οργανωμένες με τέτοιο τρόπο ώστε η τιμή του κάθε εικονοστοιχείου να περιγράφεται σε όλα τα κανάλια πριν περιγραφεί το επόμενο εικονοστοιχείο.



Εικόνα 2.10. Τυποποίηση αρχείου τύπου BIP
Πηγή : esri.com

Τέλος η τυποποίηση BIL σύμφωνα με την οποία λειτουργούν και τα προγράμματα αυτής της εργασίας, περιγράφει την εικόνα αποθηκεύοντας την κάθε σειρά της σε κάθε κανάλι πριν προχωρήσει στην επόμενη, όπως φαίνεται και στην παρακάτω εικόνα.



Εικόνα 2.11. Τυποποίηση αρχείου τύπου BIL
Πηγή : esri.com

Για να λειτουργήσουν όλοι αυτοί οι τρόποι οργάνωσης της εικόνας ώστε να μπορέσει η εικόνα να προβληθεί στην οθόνη του υπολογιστή ή να εκτυπωθεί, πρέπει να γνωρίζει με κάποιο τρόπο το αντίστοιχο πρόγραμμα, τον αριθμό των γραμμών, των στηλών και των καναλιών της εικόνας, καθώς και τον τύπο δεδομένων που είναι τα εικονοστοιχεία (ακέραιος, δεκαδικός, κ.α.). Αυτό επιτυγχάνεται είτε χειροκίνητα, δηλαδή δίνοντας ο χρήστης τα δεδομένα αυτά, είτε αυτόματα, δηλαδή διαβάζοντας τα από ένα κομμάτι του αρχείου της εικόνας που βρίσκεται στην αρχή, ή διαβάζοντάς τα από ένα συνοδευτικό αρχείο.

3. ΜΕΘΟΔΟΙ ΤΗΣ ΥΠΕΡΦΑΣΜΑΤΙΚΗΣ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

Στο κεφάλαιο αυτό αναλύεται η θεωρία που αφορά στις μεθόδους της υπερφασματικής τηλεπισκόπησης, που χρησιμοποιούνται κατά τη διαδικασία του φασματικού διαχωρισμού (unmixing) της εικόνας σε καθαρούς στόχους. Η βελτίωση και τελειοποίηση της διαδικασίας αυτής είναι ένας από τους κυριότερους στόχους της υπερφασματικής τηλεπισκόπησης.

3.1. Η Διαδικασία του Φασματικού Διαχωρισμού

Το ζήτημα του φασματικού διαχωρισμού των τηλεπισκοπικών απεικονίσεων είναι ένα από τα πρωταρχικά προβλήματα, που κλήθηκε να λύσει η τηλεπισκόπηση. Με την εξέλιξη της επιστήμης και της τεχνολογίας των δεκτών, καθώς και με την εμφάνιση των υπερφασματικών δεκτών, οι επιστήμονες βρέθηκαν ένα βήμα πιο κοντά στην επίλυσή του. Και αυτό γιατί το μεγάλο πλήθος των καναλιών των υπερφασματικών δεκτών, σε συνδυασμό με το ότι το κάθε κανάλι είναι όμορο με τα γειτονικά του, έπαιξαν σημαντικό ρόλο, καθώς έτσι ο ερευνητής έχει μια πιο πλήρη εικόνα των τιμών ανακλαστικότητας του κάθε εικονοστοιχείου σε ένα ευρύ κομμάτι του ηλεκτρομαγνητικού φάσματος.

Ο φασματικός διαχωρισμός αποτελείται από τρία κύρια στάδια: το στάδιο της εύρεσης του φασματικού υπόχωρου του σήματος, το στάδιο εξαγωγής των καθαρών στόχων και το στάδιο εκτίμησης της αφθονίας των καθαρών στόχων σε επίπεδο εικονοστοιχείου. Για κάθε ένα από τα στάδια αυτά έχουν αναπτυχθεί διάφορες μέθοδοι. Οι μέθοδοι που αναλύθηκαν και προγραμματίστηκαν για τον φασματικό διαχωρισμό στο πλαίσιο αυτής της διπλωματικής εργασίας, χρησιμοποιούν στατιστικές, γεωμετρικές και ελαχιστοτετραγωνικές προσεγγίσεις για κάθε ένα από τα στάδια του φασματικού διαχωρισμού αντίστοιχα. Καθεμία από τις μεθόδους που υλοποιήθηκαν προγραμματιστικά αποτελεί και ένα βήμα της διαδικασίας του φασματικού διαχωρισμού, καθώς καθεμιά ακολουθεί την προηγούμενη και συμπληρώνεται από την επόμενη.

Η διαδικασία του φασματικού διαχωρισμού, όπως υλοποιείται από τις μεθόδους αυτές, ξεκινά με την εκτίμηση του φασματικού υπόχωρου του σήματος που θεωρητικά είναι συνυφασμένος με το πλήθος των καθαρών

στόχων που υπάρχουν στην εικόνα. Όμως για να γίνει αυτό πρέπει ο θόρυβος της εικόνας να μετατραπεί σε λευκό γκαουσιανό θόρυβο, αν πρόκειται για πραγματικά δεδομένα, και να ταξινομηθούν τα κανάλια σύμφωνα με το ποσοστό της πληροφορίας που περιέχουν. Για αυτήν τη διαδικασία είναι υπεύθυνος ο αλγόριθμος MNF.

Ο MNF μετατρέπει την απεικόνιση, σε απεικόνιση με λευκό θόρυβο ώστε να είναι δυνατή η εκτίμηση του πλήθους των καθαρών στόχων. Ο αλγόριθμος εκτίμηση του πλήθους των καθαρών στόχων, ο ODM, χρησιμοποιώντας την μεταβλητότητα του κάθε καναλιού εκτιμά τον αριθμό των καθαρών στόχων. Σε περίπτωση που η αρχική εικόνα έχει ήδη λευκό θόρυβο, η εικόνα που πρέπει να διαβάσει ο αλγόριθμος ODM είναι η εικόνα των κυρίων συνιστωσών, καθώς δεν χρειάζεται να «λευκοποιηθεί» ο θόρυβος. Αυτό συμβαίνει συνήθως σε συνθετικές εικόνες.

Ο αριθμός των καθαρών στόχων ο οποίος απαιτείται για να περιγράψει ικανοποιητικά την εικόνα, καθορίζει και τη απαραίτητη φασματική διάσταση, την οποία θα επεξεργαστεί ο αλγόριθμος εξαγωγής καθαρών στόχων για την εξαγωγή των φασματικών υπογραφών των καθαρών στόχων,. Ο αλγόριθμος εξαγωγής καθαρών στόχων, αξιοποιώντας την ιδιότητα των καθαρών στόχων να βρίσκονται στις κορυφές του διαγράμματος διασποράς των πρώτων κυρίων συνιστωσών το οποίο αποτελεί μια χωρική διάταξη (simplex), επιλέγει υποψήφιους καθαρούς στόχους και με βάση τις φασματικές γωνίες των τελευταίων, καταλήγει στην εξαγωγή των φασματικών υπογραφών των καθαρών στόχων.

Τέλος, ο αλγόριθμος εκτίμησης της αφθονίας των καθαρών στόχων, έχοντας ως δεδομένο τις φασματικές υπογραφές των καθαρών στόχων, θα παράγει τους χάρτες αφθονίας, για κάθε καθαρό στόχο. Οι χάρτες αυτοί θα προκύψουν από την επίλυση του γραμμικού μοντέλου για τους συγκεκριμένους καθαρούς στόχους. Έτσι, ο αλγόριθμος εκτίμησης της αφθονίας ολοκληρώνει τη διαδικασία του φασματικού διαχωρισμού, επιλύοντας του γραμμικό σύστημα που προκύπτει για κάθε εικονοστοιχείο.

3.2. Μέθοδος Εκτίμησης του φασματικού υπόχωρου του σήματος

Η εκτίμηση του πλήθους των πηγών του σήματος είναι ένα θεμελιώδες πρόβλημα της επιστήμης της επεξεργασίας σήματος. Στο επιστημονικό πεδίο

της υπερφασματικής τηλεπισκόπησης οι πηγές του σήματος έχουν μοναδικές φασματικές υπογραφές και αποτελούν τους καθαρούς στόχους. Η μεγάλη πλειοψηφία των μεθόδων εξαγωγής των καθαρών στόχων χρειάζονται σαν δεδομένο το πλήθος των στόχων αυτών, για να καθορίσουν το βέλτιστο σύνολο των φασματικών υπογραφών τους. Η σωστή εκτίμηση του αριθμού τους έχει μεγάλη επίδραση στην διαδικασία εξαγωγής τους, αλλά και στην διαδικασία “αποσύνθεσης” της εικόνας σε αυτούς.

Επίσης, η σωστή εκτίμηση του πλήθους των καθαρών στόχων δίνει την δυνατότητα της μείωσης των διαστάσεων της εικόνας, παράγοντας μια νέα εικόνα με ακριβώς όσα κανάλια χρειάζονται για να την περιγράψουν επαρκώς. Ως εκ τούτου, ο ακριβής προσδιορισμός του πλήθους των καθαρών στόχων βελτιώνει σημαντικά την ακρίβεια της φασματικής αποσύνθεσης της εικόνας και επιτρέπει την αναδημιουργία της με λιγότερες διαστάσεις, πράγμα που έχει σημαντικά πλεονεκτήματα στην επεξεργασία και αποθήκευση της.

Για τον προσδιορισμό του πλήθους των καθαρών στόχων χρησιμοποιήθηκε η μέθοδος ODM (Outlier Detection Method), με βασικό χαρακτηριστικό το ότι είναι πλήρως αυτόματη και μη-παραμετρική.

Ένα πρόβλημα που απασχολεί πολύ τους ερευνητές που ειδικεύονται στον τομέα της υπερφασματικής τηλεπισκόπησης είναι ο καθορισμός ενός κατωφλιού μεταξύ του σήματος και του θορύβου. Η μέθοδος ODM εισάγει μια νέα προσέγγιση, οπού θεωρεί τον θόρυβο, ως στατιστικό δεδομένο και το σήμα, ως ανωμαλία του δεδομένου αυτού. (Andreou C. and Karathanassi V. 2014) Ο καταλληλότερος, για την περίπτωση της υπερφασματικής τηλεπισκόπησης, ορισμός της στατιστικής ανωμαλίας είναι η παρατήρηση, η οποία διαφοροποιείται τόσο από τις υπόλοιπες που εγείρει υπόνοιες ότι δημιουργήθηκε από κάποιον άλλο μηχανισμό. (Hawkins D.M. 1980)

Σε αντίθεση με αντίστοιχες μεθόδους εκτίμησης του πλήθους των καθαρών στόχων, οι οποίες εστιάζουν στον υπόχωρο του σήματος, η ODM επικεντρώνεται στον υπόχωρο του θορύβου, αξιοποιώντας το γεγονός ότι ο γεωμετρικός τόπος του θορύβου, σε μια εικόνα που έχει υποστεί τον μετασχηματισμό ελαχιστοποίησης του θορύβου, είναι η υπερσφαίρα στις ν διαστάσεις, όπου ν ο αριθμός των καναλιών. Για τον εντοπισμό των ανωμαλιών χρησιμοποιείται η μέθοδος inter quartile range (IRQ), με χαρακτηριστικό το πλεονέκτημα της να μπορεί να χρησιμοποιηθεί όταν η κατανομή των δεδομένων είναι άγνωστη. (Andreou C. and Karathanassi V. 2014)

Για την χρήση της μεθόδου IQR είναι απαραίτητο να υπολογιστούν τα στατιστικά μεγέθη της εικόνας που είναι αποτέλεσμα του μετασχηματισμού ελαχιστοποίησης του θορύβου. Χρησιμοποιώντας τύπους της στατιστικής, υπολογίζεται η μέση τιμή και η τυπική απόκλιση του κάθε καναλιού. Οι τυπικές αποκλίσεις του καναλιών όμως έχουν μεγάλες αποκλίσεις μεταξύ τους πράγμα το οποίο θέτει ως προϋπόθεση την κανονικοποίησή τους πριν την περαιτέρω επεξεργασία τους. Επόμενο βήμα είναι η ταξινόμηση κατά φθίνουσα σειρά των κανονικοποιημένων πλέον τυπικών αποκλίσεων για να μπορεί να εφαρμοστεί η μέθοδος IRQ. Μετά την διαδικασία αυτή, μπορούμε να θεωρήσουμε ένα διάγραμμα με άξονα x τον αριθμό του καναλιού και άξονα y την τυπική απόκλιση του. Χρησιμοποιώντας το διάγραμμα αυτό υπολογίζονται οι ευκλείδειες αποστάσεις των σημείων του διαγράμματος με βάση τον τύπο:

$$ED_{i,i+1} = \sqrt{(s_i - s_{i+1}) + (i - (i+1)}$$

όπου, $ED_{i,i+1}$ η ευκλείδεια απόσταση των τυπικών αποκλίσεων των καναλιών i και i+1,

και s_i και s_{i+1}, οι τυπικές αποκλίσεις των καναλιών i και i+1 αντίστοιχα.

Αυτές οι ευκλείδειες αποστάσεις είναι τα στατιστικά δεδομένα στα οποία εφαρμόζεται η μέθοδος IQR. Η μέθοδος αυτή θεωρεί ότι τα δεδομένα που βρίσκονται πάνω από ένα συγκεκριμένο κατώφλι απέχουν πολύ περισσότερο από τις κεντρικές τιμές από το αναμενόμενο και χαρακτηρίζονται ως στατιστικές ανωμαλίες. Το κατώφλι αυτό προκύπτει από τον τύπο:

$$Q_3 + 1.5 \times IRQ$$

όπου, Q_3 συμβολίζεται η τιμή του στοιχείου με αριθμό στη διάταξη της φθίνουσας στύλης, ίσο με το 75% του συνόλου των v-1 μελών της στήλης (όπου v, ο αριθμός των καναλιών της υπερφασματικής απεικόνισης). Το μέγεθος IRQ υπολογίζεται από τον τύπο:

$$IRQ = Q_3 - Q_1$$

όπου, Q_1 είναι αντίστοιχα η τιμή του στοιχείου με αριθμό διάταξη της φθίνουσας στύλης, ίσο με το 25(v-1)/100.

Ένα πολύ σημαντικό πλεονέκτημα της μεθόδου είναι το ότι είναι μη-παραμετρική και ότι το κατώφλι που χρησιμοποιείται προκύπτει από τα ίδια τα δεδομένα. Έτσι καταλήγουμε σε ένα, διαφορετικό για κάθε εικόνα, κατώφλι το οποίο τελικά καθορίζει το τι είναι πληροφορία και τι θόρυβος, και συμπεραίνουμε ότι ο αριθμός των καναλιών μετά τον οποίο δεν περιέχεται καθόλου πληροφορία, αλλά μόνο θόρυβος, είναι και ο αριθμός των καθαρών στόχων της εικόνας.

Όπως αναφέρθηκε και παραπάνω, η μέθοδος εκτίμησης των καθαρών στόχων, χρησιμοποιεί ως δεδομένο την εικόνα λευκού θορύβου. Έτσι, πριν τη μέθοδο αυτή, πρέπει να προηγηθεί ο μετασχηματισμός ελαχιστοποίησης του θορύβου, για την επεξεργασία πραγματικών δεδομένων, ή ο μετασχηματισμός κυρίων συνιστωσών για την επεξεργασία συνθετικών δεδομένων, τα οποία όμως περιέχουν λευκό θόρυβο.

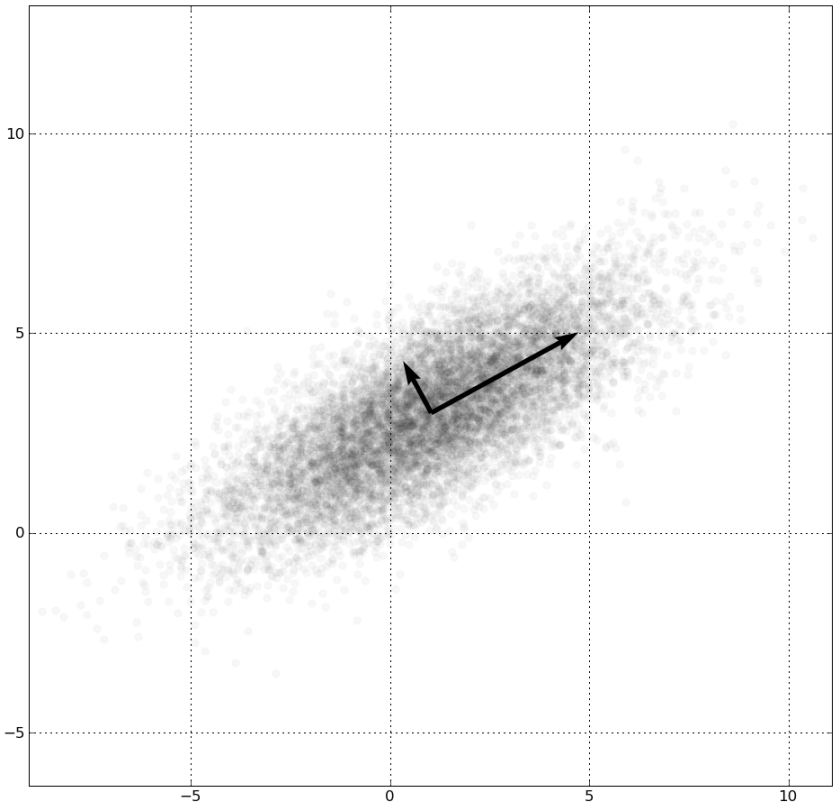
3.3. Ανάλυση Κυρίων Συνιστωσών

Η μέθοδος της ανάλυσης κυρίων συνιστωσών (Principal Components Analysis) αποτελεί μια διαδεδομένη μέθοδο της στατιστικής, για τη μείωση των διαστάσεων ενός δειγματικού χώρου, χωρίς απώλεια πληροφορίας. Η συγκεκριμένη μέθοδος χρησιμοποιεί έναν ορθογώνιο μετασχηματισμό, για να μετατρέψει ένα σύνολο παρατηρήσεων συσχετισμένων μεταβλητών σε ένα σύνολο ασυσχέτιστων μεταβλητών οι οποίες ονομάζονται κύριες συνιστώσες. Το πλήθος των συνιστωσών πρέπει να είναι μικρότερο η ίσο με το πλήθος των αρχικών μεταβλητών. Η πρώτη κύρια συνιστώσα ορίζεται στη διεύθυνση της μεγαλύτερης μεταβλητότητας, έτσι ώστε να περιγράφει μεγάλο ποσοστό πληροφορίας και κάθε επόμενη συνιστώσα στη διεύθυνση της αμέσως επόμενης μεγαλύτερης μεταβλητότητας, με την προϋπόθεση να είναι κάθετη στις προηγούμενες. Η καθετότητα είναι απαραίτητη για να εξασφαλιστεί ότι οι νέες μεταβλητές είναι ασυσχέτιστες.

Το γεγονός ότι όσο πιο μικρός είναι ο αριθμός της συνιστώσας τόσο μεγαλύτερη μεταβλητότητα έχει και τόσο πιο πολλή πληροφορία περιέχει οδηγεί στη μικρή μεταβλητότητα, άρα και χρησιμότητα των τελευταίων συνιστωσών. Αυτό καθιστά τη μέθοδο των κυρίων συνιστωσών χρήσιμη στη

μείωση των διαστάσεων του δειγματικού χώρου, χωρίς, πιθανότατα, την μείωση της πληροφορίας.

Εφαρμογή της ανάλυσης κυρίων συνιστωσών υπάρχει και στην τηλεπισκόπηση, όπου ο δειγματικός χώρος είναι η εικόνα, μεταβλητές είναι τα κανάλια της εικόνας και παρατηρήσεις είναι οι τιμές των εικονοστοιχείων σε κάθε κανάλι. Στο επιστημονικό πεδίο της τηλεπισκόπησης, η μέθοδος των κυρίων συνιστωσών χρησιμεύει στη συμπύκνωση της πληροφορίας της εικόνας στις πρώτες κύριες συνιστώσες και στη μείωση του αριθμού των καναλιών της εικόνας χωρίς ταυτόχρονη μείωση της πληροφορίας που περιέχει. Αυτό επιτυγχάνεται λόγω του ότι, εφόσον οι κύριες συνιστώσες είναι στην διεύθυνση της μεγαλύτερης μεταβλητότητας, μπορούν να περιγράψουν το σύνολο σχεδόν της πληροφορίας σε μικρότερο αριθμό καναλιών.



Εικόνα 3.1. Σχηματική απεικόνιση των αξόνων των κυρίων συνιστωσών
Πηγή : Wikipedia

Το πρώτο βήμα για την υλοποίηση της μεθόδου των κυρίων συνιστωσών είναι ο υπολογισμός των στατιστικών της εικόνας. Για το στάδιο αυτό χρησιμοποιούνται οι τύποι της στατιστικής για τον υπολογισμό της μέσης τιμής και της μεταβλητότητας. Έστω, λοιπόν Χ το σύνολο των εικονοστοιχείων

της εικόνας, σε p αριθμό καναλιών και n αριθμό εικονοστοιχείων σε κάθε κανάλι. Τότε:

$$\overline{X_i} = \frac{\sum_{j=1}^{n} X_j}{n} \text{ για } i = 1,2,...,p$$

Όπου, $\overline{X_i}$ συμβολίζεται η μέση τιμή των εικονοστοιχείων καναλιού i,

X_j , το κάθε εικονοστοιχείο του καναλιού και

n, το πλήθος των εικονοστοιχείων που περιέχει κάθε κανάλι.

Έτσι, αφού έχει υπολογιστεί η μέση τιμή του κάθε καναλιού, μπορεί να υπολογιστεί ο πίνακας μεταβλητότητας-συμμεταβλητότητας μεγέθους p × p για όλη την εικόνα, σύμφωνα με τον τύπο:

$$Cov(X_i, X_j) = \frac{\sum_{k=1}^{n} (X_{ik} - \overline{X_i})(X_{ik} - \overline{X_j})}{n}$$

Όπου, $Cov(X_i, X_j)$ είναι η συμμεταβλητότητα των καναλιών X_i και X_j

X_{ik} , η τιμή του κάθε εικονοστοιχείου στο κανάλι X_i

και n, το πλήθος των εικονοστοιχείων που περιέχει κάθε κανάλι.

Στην συνέχεια πρέπει να υπολογιστούν οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα μεταβλητότητας, γιατί ο πίνακας των ιδιοδιανυσμάτων αποτελεί τον πίνακα στροφής του συστήματος, ώστε οι νέοι άξονες της εικόνας να είναι κάθετοι μεταξύ τους και να βρίσκονται στην διεύθυνση της μέγιστης μεταβλητότητας. Πριν γίνει ο πολλαπλασιασμός της εικόνας με τον πίνακα των ιδιοδιανυσμάτων, πρέπει να ταξινομηθούν τα ιδιοδιανύσματα σε φθίνουσα σειρά με βάση την τιμή της αντίστοιχης ιδιοτιμής του καθενός, ώστε οι κύριες συνιστώσες να είναι ταξινομημένες σε φθίνουσα σειρά, με βάση την τιμή της μεταβλητότητας της καθεμιάς. Ο πολλαπλασιασμός θα γίνει με βάση τον παρακάτω τύπο:

$$Y_{v\mu} = \sum_{i=0}^{p} (X_{vi} * E_{\mu i})$$

όπου, $Y_{vμ}$ συμβολίζεται η τιμή του v–ιστού εικονοστοιχείου της μ-οστής συνιστώσας της μετασχηματισμένης εικόνας,

X_{vi}, η τιμή του v –οστού εικονοστοιχείου της αρχικής εικόνας στο i-οστό κανάλι της,

$E_{μi}$ είναι η i-οστή τιμή του μ-οστού ιδιοδιανύσματος, και

p το πλήθος των καναλιών.

Η διαδικασία όμως αυτή οδηγεί στην στροφή των αξόνων της εικόνας στην διεύθυνση της μέγιστης μεταβλητότητας, χωρίς να μετατοπίζει το κέντρο των αξόνων στο κέντρο των δεδομένων. Για να επιτευχθεί αυτό, πρέπει να αφαιρεθεί, από την τιμή του κάθε εικονοστοιχείου, η μέση τιμή του καναλιού στο οποίο ανήκει, σύμφωνα με τον τύπο:

$$Y_{v\mu} = \sum_{i=0}^{p} [(X_{vi} - \overline{X_i}) * E_{\mu i}]$$

όπου, $Y_{vμ}$ συμβολίζεται η τιμή του v–ιστού εικονοστοιχείου της μ-οστής συνιστώσας της μετασχηματισμένης εικόνας,

X_{vi}, η τιμή του v –οστού εικονοστοιχείου της αρχικής εικόνας στο i-οστό κανάλι της,

$\overline{X_i}$, η μέση τιμή του καναλιού i της αρχικής εικόνας,

$E_{μi}$ είναι η i-οστή τιμή του μ-οστού ιδιοδιανύσματος, και

p το πλήθος των καναλιών.

3.4. Ο Μετασχηματισμός Ελαχιστοποίησης του Θορύβου (MNF)

Ο μετασχηματισμός ελαχιστοποίησης του θορύβου (Minimum Noise Fraction (MNF)) (Green et al., 1988) είναι μια πολύ σημαντική μέθοδος ελαχιστοποίησης του θορύβου και μείωσης των διαστάσεων μιας τηλεπισκοπικής εικόνας. Ο μετασχηματισμός αυτός μετατρέπει τον θόρυβο της εικόνας σε λευκό γκαουσιανό θόρυβο, με μέση τιμή ίση με το μηδέν και τυπική απόκλιση ίση με ένα και επιτυγχάνει την προβολή των χρήσιμων πληροφοριών σε ένα μικρότερο σύνολο καναλιών, με αποτέλεσμα τη μείωση

των αποθηκευτικών απαιτήσεων και των υπολογιστικών πόρων που χρειάζονται για την επεξεργασία της εικόνας.

Σημαντικό μειονέκτημα του μετασχηματισμού ελαχιστοποίησης του θορύβου, είναι το γεγονός ότι κατά το μετασχηματισμό αλλάζει η φυσική σημασία των αρχικών δεδομένων. Τα κανάλια του νέου –μικρότερων διαστάσεων- υπόχωρου δεν αντιστοιχούν στα επιμέρους αρχικά κανάλια, αλλά σε γραμμικούς συνδυασμούς τους, γεγονός που μπορεί να περιπλέξει και δυσκολέψει την ερμηνεία των αποτελεσμάτων.

Ο μετασχηματισμός ελαχιστοποίησης του θορύβου δημιουργήθηκε από την ανάγκη να υπάρχει μια μέθοδος η οποία να δίνει κάθε φορά ως αποτέλεσμα, μια εικόνα που να περιέχει το μεγαλύτερο κομμάτι της πληροφορίας στα πρώτα κανάλια, ενώ τα επόμενα περιέχουν κυρίως θόρυβο. Αυτό επιτυγχάνεται μέσω της «στροφής» της εικόνας ως προς τον λόγο σήμα προς θόρυβο (signal to noise ratio (SNR)).

Αν υποθέσουμε ότι η αρχική εικόνα έχει P πλήθος καναλιών και N πλήθος εικονοστοιχείων. Η εικόνα αυτή θα είναι ένα πολυδιάστατο μέγεθος και συμβολίζεται ως:

$$Z_i(X) \quad \text{με } i = 1,2,...,p$$

Όπου το X συμβολίζει την φασματική τιμή του κάθε εικονοστοιχείου. Θεωρούμε ότι:

$$Z(X) = S(X) + N(X)$$

όπου, S(X) συμβολίζεται η συνάρτηση του σήματος και
N(X) η συνάρτηση του θορύβου.

Εφόσον το σήμα και ο θόρυβος είναι ασυσχέτιστα μεγέθη θα ισχύει:

$$Cov\{Z(X)\} = Cov\{S(X)\} + Cov\{N(X)\}$$

όπου, Cov{Z(X)} ο πίνακας μεταβλητότητας της εικόνας,
Cov{S(X)} ο πίνακας μεταβλητότητας του σήματος και
Cov{N(X)} ο πίνακας μεταβλητότητας του θορύβου.

Στις περισσότερες περιπτώσεις οι πίνακες μεταβλητότητας του θορύβου και του σήματος είναι άγνωστα μεγέθη. Έτσι με μεθόδους που αναλύονται σε παρακάτω κεφάλαιο, εκτιμάται ο θόρυβος. Η εκάστοτε μέθοδος εκτίμησης του θορύβου σχηματίζει μια νέα εικόνα την οποία θεωρούμε εικόνα του θορύβου, από την οποία υπολογίζονται τα στατιστικά μεγέθη της με χρήση των τύπων στατιστικής:

$$\overline{X_i} = \frac{\sum_{j=1}^{n} X_j}{n} \text{ για } i = 1,2,...,p$$

Όπου, $\overline{X_i}$ συμβολίζεται η μέση τιμή των εικονοστοιχείων καναλιού i,

X_j , το κάθε εικονοστοιχείο του καναλιού και

n, το πλήθος των εικονοστοιχείων που περιέχει κάθε κανάλι του θορύβου.

$$Cov(X_i, X_j) = \frac{\sum_{k=1}^{n} (X_{ik} - \overline{X_i})(X_{ik} - \overline{X_j})}{n}$$

Όπου, Cov(X_i, X_j) είναι η συμμεταβλητότητα των καναλιών X_i και X_j του θορύβου,

X_{ik} , η τιμή του κάθε εικονοστοιχείου στο κανάλι X_i, και

n, το πλήθος των εικονοστοιχείων που περιέχει κάθε κανάλι.

Χρησιμοποιώντας τη μέθοδο της διαγωνοποίησης πίνακα παίρνουμε τις ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα μεταβλητότητας του θορύβου, αξιοποιώντας την τεχνική «eigendecomposition”.

Το επόμενο βήμα είναι η στροφή της εικόνας στους άξονες που ορίζουν τα ιδιοδιανύσματα του πίνακα μεταβλητότητας του θορύβου. Για τον σκοπό αυτό πρέπει η τιμή του κάθε εικονοστοιχείου της νέας εικόνας να ισούται με το άθροισμα των γινομένων της τιμής του ίδιου εικονοστοιχείου στην αρχική εικόνα σε κάθε κανάλι με την αντίστοιχη τιμή του αντίστοιχου ιδιοδιανύσματος. Για παράδειγμα η τιμή του v–ιστού εικονοστοιχείου της μ-οστής συνιστώσας ισούται με το άθροισμα των γινομένων της τιμής του v –οστού εικονοστοιχείου

της αρχικής εικόνας επί το αντίστοιχο στοιχείο του μ-οστού ιδιοδιανύσματος. Η πράξη αυτή περιγράφεται μαθηματικά παρακάτω.

$$Y_{v\mu} = \sum_{i=0}^{p} [(X_{vi} - \overline{X_i}) * E_{\mu i}]$$

όπου, $Y_{v\mu}$ συμβολίζεται η τιμή του ν–ιστού εικονοστοιχείου της μ-οστής συνιστώσας της μετασχηματισμένης εικόνας,

X_{vi}, η τιμή του ν –οστού εικονοστοιχείου της αρχικής εικόνας στο i-οστό κανάλι της,

$\overline{X_i}$, η μέση τιμή του καναλιού i της αρχικής εικόνας,

$E_{\mu i}$ είναι η i-οστή τιμή του μ-οστού ιδιοδιανύσματος, και

p το πλήθος των καναλιών.

Λόγω του ότι η εικόνα που θα προκύψει μετά την στροφή πρέπει να έχει μέση τιμή μηδέν, για να συμπίπτουν οι μέσες τιμές με την αρχή των μετασχηματισμένων αξόνων, αφαιρείται από την τιμή κάθε εικονοστοιχείου, η μέση τιμή του καναλιού στο οποίο ανήκει.

Στη συνέχεια η τιμή του κάθε εικονοστοιχείου διαιρείται με την τιμή της τυπικής απόκλισης του θορύβου στο αντίστοιχο κανάλι της αρχικής εικόνας του θορύβου, σύμφωνα με τον τύπο:

$$Z_i = \frac{Y_i}{S_i}$$

όπου, Z_i συμβολίζεται η τιμή του κάθε εικονοστοιχείου του καναλιού i της εικόνας που περιέχει λευκό γκαουσιανό θόρυβο,

Y_i είναι η τιμή του κάθε εικονοστοιχείου της μετασχηματισμένης εικόνας και

S_i η τυπική απόκλιση θορύβου του καναλιού i της αρχικής εικόνας του θορύβου.

Έτσι, η μετασχηματισμένη εικόνα που προκύπτει περιέχει θόρυβο με μηδενική μέση τιμή και ίση τυπική απόκλιση ίση με τη μονάδα σε όλα τα κανάλια, περιέχει δηλαδή λευκό θόρυβο. Επομένως, αν θεωρήσουμε ένα σύστημα με

διάσταση p, τη διάσταση της υπερφασματικής απεικόνισης και άξονες τα κανάλια της νέας εικόνας, ο γεωμετρικός τόπος του θορύβου θα είναι μια υπερσφαίρα με ακτίνα ίση με την κανονικοποιημένη τυπική απόκλιση του θορύβου.

Για να αποκτήσει η νέα εικόνα την ιδιότητα της σταδιακής μείωσης της πληροφορίας με την αύξηση του αριθμού του καναλιού, πρέπει τα κανάλια αυτά να ταξινομηθούν με βάση τον λόγο σήμα προς θόρυβο (SNR). Όμως, εφόσον ο θόρυβος είναι ίδιος σε κάθε κανάλι, λόγω του προηγούμενου μετασχηματισμού, αρκεί να ταξινομηθούν τα κανάλια με βάση τη διασπορά που εμφανίζουν. Έτσι αρκεί ο μετασχηματισμός της εικόνας με βάση την μέθοδο των κυρίων συνιστωσών (PCA).

Πιο συγκεκριμένα, χρησιμοποιώντας τις σχέσεις της στατιστικής που αναπτύχθηκαν παραπάνω, υπολογίζονται τα στατιστικά μεγέθη της εικόνας, όπως η μέση τιμή κάθε καναλιού και ο πίνακας μεταβλητότητας-συμμεταβλητότητας της εικόνας.

Στην συνέχεια υπολογίζονται τα χαρακτηριστικά μεγέθη του πίνακα αυτού, έτσι ώστε να έχουμε τον πίνακα των ιδιοδιανυσμάτων του πίνακα μεταβλητότητας της μετασχηματισμένης εικόνας, ο οποίος είναι και ο πίνακας στροφής των αξόνων της εικόνας.

Έτσι λοιπόν η τελική εικόνα του μετασχηματισμού MNF, υπολογίζεται σύμφωνα με την παρακάτω σχέση:

$$X_{(b,r*c)} = \left(Y_{(b,r*c)} - M_{(b,r*c)}\right) * E_{(b,b)}$$

όπου, Χ συμβολίζεται ο πίνακας από τον οποίο αποτελείται η νέα εικόνα με μέγεθος bands x rows*columns,

Υ αποτελεί τον πίνακα της εικόνας λευκού θορύβου που προέκυψε από τον προηγούμενο μετασχηματισμό,

Μ συμβολίζει τον πίνακα με τις μέσες τιμές κάθε καναλιού της εικόνας λευκού θορύβου, και ο πίνακας

Ε αναφέρεται στον πίνακα των ιδιοδιανυσμάτων του πίνακα μεταβλητότητας της εικόνας που περιέχει λευκό θόρυβο.

Έτσι, η Μέθοδος Ελαχιστοποίησης του Θορύβου δίνει μια εικόνα στην οποία ο θόρυβος είναι λευκός και γκαουσιανός, δηλαδή έχει μέση τιμή ίση με το μηδέν, τυπική απόκλιση ίση με ένα και ακολουθεί την κανονική κατανομή. Επίσης, το μεγαλύτερο κομμάτι της πληροφορίας της εικόνας περιέχεται στα πρώτα κανάλια της, χάρη στο δεύτερο μετασχηματισμό, που αναλύθηκε παραπάνω. Αυτές τις ιδιότητες αξιοποιεί η μέθοδος εκτίμησης του αριθμού των καθαρών στόχων, για να διαχωρίσει την πληροφορία από τον θόρυβο.

3.5. Μέθοδοι Εκτίμησης του Θορύβου

3.5.1. Nearest Neighbor Distance

Πολλές μέθοδοι της υπερφασματικής τηλεπισκόπησης προϋποθέτουν την γνώση του πίνακα μεταβλητότητας του θορύβου προκειμένου να μετασχηματίσουν την εικόνα σε εικόνα λευκού θορύβου. Έτσι προκύπτει η ανάγκη ύπαρξης και εφαρμογής μεθόδων εκτίμησης του θορύβου.

Όλες αυτές οι μέθοδοι εφαρμόζονται σε κάθε εικόνα πριν την επεξεργασία της, γιατί για κάθε εικόνα και του ίδιου ακόμα δέκτη ο πίνακας μεταβλητότητας του θορύβου είναι διαφορετικός, καθώς δεν προκύπτει μόνο από συστηματικά σφάλματα του δέκτη αλλά και από άλλους παράγοντες που είναι διαφορετικοί σε κάθε περιοχή και χρονική στιγμή, όπως οι επιδράσεις της ατμόσφαιρας.

Η μέθοδος Nearest Neighbor Distance (NND) είναι ένας τρόπος εκτίμησης του πίνακα μεταβλητότητας του θορύβου με μοναδικό δεδομένο την ίδια την εικόνα. Η μέθοδος στηρίζεται στην παραδοχή ότι τα γειτονικά εικονοστοιχεία μιας εικόνας «θα έπρεπε» να έχουν περίπου ίσες τιμές. Εφαρμόζοντας ένα χωρικό φίλτρο στην εικόνα, το οποίο μας δίνει τη μέση τιμή των διαφορών του δεξιά και του επάνω εικονοστοιχείου για κάθε εικονοστοιχείο της απεικόνισης, ο αλγόριθμος εξάγει τις τιμές εικονοστοιχείων που είναι θόρυβος. Από αυτές εξάγεται και ο πίνακας συμμεταβλητότητας του θορύβου.

Ουσιαστικά, η μέθοδος NND εφαρμόζει ένα χαμηλοδιαβατό χωρικό φίλτρο σε όλη την εικόνα, πριν διαιρέσει το αποτέλεσμα του διά δύο, για να εκτιμήσει το θόρυβο. Ο πίνακας συνέλιξης του συγκεκριμένου φίλτρου είναι ο παρακάτω:

-1	
2	-1

Από την θεωρία των χωρικών φίλτρων προκύπτει επίσης ότι το συγκεκριμένο φίλτρο, αλλά και γενικότερα τα χαμηλοδιαβατά χωρικά φίλτρα χρησιμοποιούνται για την εκτίμηση του τυχαίου θορύβου.

3.5.2. Multiple Regression Theory-Based Method

Η θεωρία της πολλαπλής παλινδρόμησης χρησιμοποιείται συχνά στην υπερφασματική τηλεπισκόπηση, καθώς η δομή και η ποσότητα των δεδομένων το επιτρέπει. Πιο συγκεκριμένα, οι τιμές των γειτονικών καναλιών μιας υπερφασματικής εικόνας είναι συσχετισμένες μεταβλητές, πράγμα που επιτρέπει τον υπολογισμό της τιμής του κάθε εικονοστοιχείου ενός συγκεκριμένου καναλιού βάσει των τιμών που έχει το εικονοστοιχείο αυτό στα υπόλοιπα κανάλια. Την ιδιότητα αυτή των υπερφασματικών εικόνων αξιοποιεί η μέθοδος της παλινδρόμησης για την εκτίμηση του θορύβου σε μια εικόνα, καθώς εκτιμά σύμφωνα με την παραπάνω περιγραφή την τιμή του κάθε εικονοστοιχείου και την αφαιρεί από την πραγματική του τιμή, ώστε να υπολογίσει τον θόρυβο.

Η εκτίμηση του θορύβου με τη μέθοδο της παλινδρόμησης γίνεται χρησιμοποιώντας τον τύπο:

$$\hat{n}_i = z_i - Z_{\ i} * \hat{b}_i$$

όπου, $\hat{n}_i$ είναι ένας πίνακας διαστάσεων 1 επί τον αριθμό των εικονοστοιχείων N του καναλιού i, που συμβολίζει τον θόρυβο σε αυτό το κανάλι,

z_i είναι ένας πίνακας ίδιων διαστάσεων που περιέχει τις τιμές των εικονοστοιχείων του καναλιού i της αρχικής εικόνας,

$Z_{\ i}$ είναι ένας πίνακας διαστάσεων N επί το πλήθος των καναλιών της εικόνας μειωμένο κατά ένα, που περιέχει όλες τις τιμές των εικονοστοιχείων της εικόνας εκτός από τις τιμές του καναλιού i, και

$\hat{b}_i$ είναι ένας πίνακας με διαστάσεις, όσα τα κανάλια μείον 1, επί 1, ο οποίος περιέχει τους συντελεστές παλινδρόμησης που αφορούν κάθε κανάλι.i.

Το διάνυσμα των συντελεστών παλινδρόμησης $\hat{b}_i$ υπολογίζεται από τον τύπο:

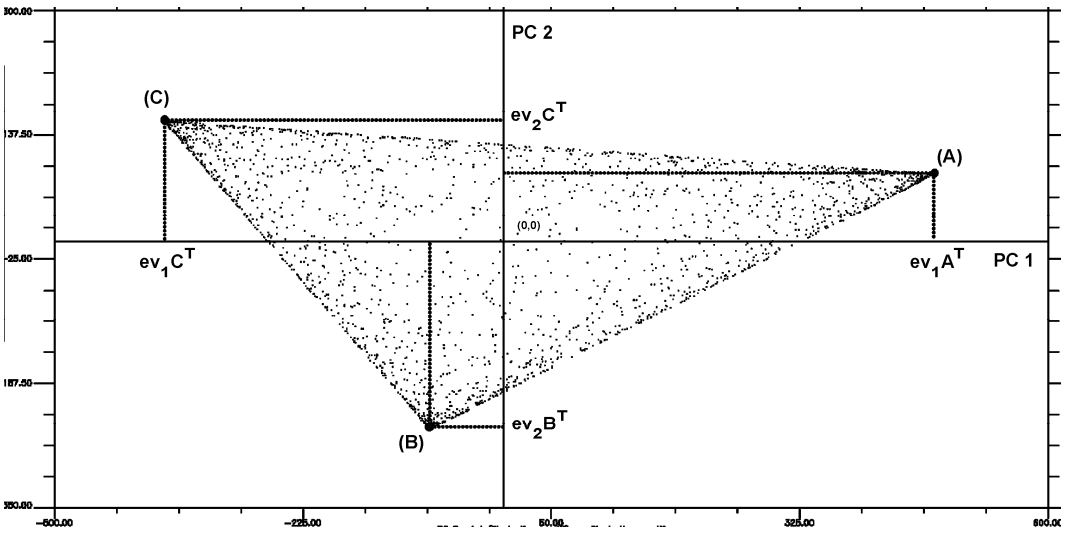
$$\hat{b}_i = (Z^T{}_i * Z_{\ i})^{-1} * Z^T{}_i * z_i$$

με τους όρους του τύπου να ακολουθούν τον ίδιο συμβολισμό, όπως στον προηγούμενο τύπο.

3.6. Μέθοδοι Εξαγωγής Καθαρών Στόχων

Μια υπερφασματική εικόνα μπορεί να αναλυθεί ως γινόμενο των φασματικών υπογραφών των καθαρών στόχων οι οποίοι εμφανίζονται σε αυτήν και του ποσοστού με το οποίο συμμετέχει ο κάθε καθαρός στόχος στη φασματική υπογραφή του κάθε εικονοστοιχείου. Επομένως μετασχηματίζοντας την εικόνα, με χρήση του αλγόριθμου PCA σε ένα χώρο με διαστάση όσο αυτή του υποχώρου του σήματος, η οποία εξάλλου είναι ίση με τον αριθμό των καθαρών στόχων μείον ένα, μπορούμε με ασφάλεια να θεωρήσουμε το σύνολο της εικόνας στο μετασχηματισμένο χώρο ως γραμμικό συνδυασμό των καθαρών στόχων.

Η μέθοδος εξαγωγής των φασματικών υπογραφών των καθαρών στόχων Simple Endmember Extraction (SEE), αξιοποιεί την ικανότητα της ανάλυσης κυρίων συνιστωσών, να μειώνει τις διαστάσεις της εικόνας χωρίς να μειώνει την πληροφορία που περιέχεται σε αυτή. Εξίσου σημαντικό είναι το γεγονός ότι οι καθαροί στόχοι της εικόνας βρίσκονται στις κορυφές του διαγράμματος διασποράς όταν η διαστάση της μετασχηματισμένης εικόνας είναι κατά μία λιγότερη από τον αριθμό των καθαρών στόχων. Αυτό προκύπτει από το γεγονός ότι το σύνολο της εικόνας αποτελεί ένα γραμμικό συνδυασμό των καθαρών στόχων.



Εικόνα 3.2. : Διάγραμμα διασποράς εικόνας με 3 καθαρούς στόχους, στις 2 πρώτες κύριες συνιστώσες της
Πηγή : Simple Endmember Extraction Methods Using Transformed Components for Hyperspectral Images

Στην εικόνα 3.2. φαίνεται ότι οι καθαροί στόχοι της τηλεπισκοπικής απεικόνισης μπορούν να ανιχνευθούν υπολογίζοντας το μέγιστο και το ελάχιστο κάθε κύριας συνιστώσας της μετασχηματισμένης εικόνας σε χώρο με τόσες διαστάσεις όσο το πλήθος των καθαρών στόχων μειωμένο κατά 1.

Έτσι, χρησιμοποιώντας την μέγιστη και την ελάχιστη τιμή του κάθε μετασχηματισμένου καναλιού, προκύπτουν 2(p-1) υποψήφιοι καθαροί στόχοι, όπου p, το πλήθος των καθαρών στόχων όπως έχει προκύψει από την μέθοδο εκτίμησης του πλήθους των καθαρών στόχων.

Με τη μέθοδο αυτή, προκύπτουν περισσότεροι καθαροί στόχοι από όσους εκτίμησε η μέθοδος εκτίμησης του υποχώρου του σήματος, καθώς, όπως φαίνεται και στην εικόνα 3.2., κάποιοι καθαροί στόχοι έχουν εντοπιστεί περισσότερες από μια φορές. Για να περιοριστεί ο αριθμός αυτός χρησιμοποιείται μια μέθοδος διαχωρισμού των υποψήφιων καθαρών στόχων με βάση τις φασματικές τους γωνίες. Έτσι πρέπει να υπολογιστούν οι φασματικές γωνίες των υποψήφιων καθαρών στόχων, με βάση τον τύπο:

$$SAD_{i,j} = \arccos\left(\frac{\sum_{k=0}^{n}(ce_{i,k}ce_{j,k})}{\sqrt{\sum_{k=0}^{n}(ce_{i.k})^2\sum_{k=0}^{n}(ce_{j,k})^2}}\right)$$

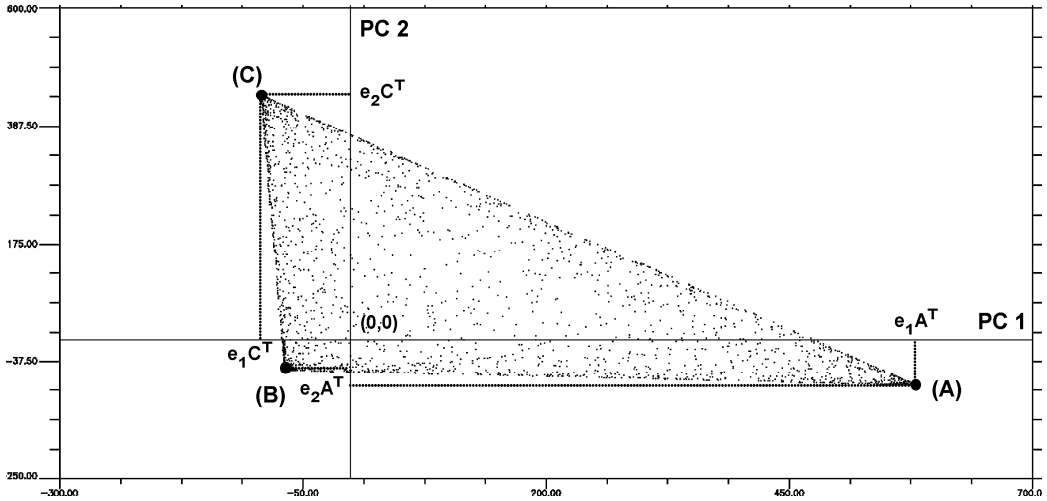
όπου, $SAD_{i,j}$ είναι η φασματική γωνία μεταξύ των υποψήφιων καθαρών στόχων i και j,

και $ce_{i,k}$ και $ce_{j,k}$ συμβολίζεται η τιμή της φασματικής υπογραφής του υποψήφιου καθαρού στόχου i και j αντίστοιχα, στο κανάλι k.

Οι υποψήφιοι καθαροί στόχοι που επαναλαμβάνονται έχουν μηδενική ή πολύ μικρή φασματική γωνία μεταξύ τους. Έτσι, το άθροισμα των φασματικών γωνιών αυτών των καθαρών στόχων, θα είναι μικρότερο από το αντίστοιχο των καθαρών στόχων που δεν επαναλαμβάνονται.

Επομένως, γνωρίζοντας το πλήθος των καθαρών στόχων και το άθροισμα των φασματικών γωνιών των υποψήφιων καθαρών στόχων, είναι δυνατό να επιλεγούν οι πραγματικοί καθαροί στόχοι, αφού πρώτα κανονικοποιηθούν τα αθροίσματα αυτά.

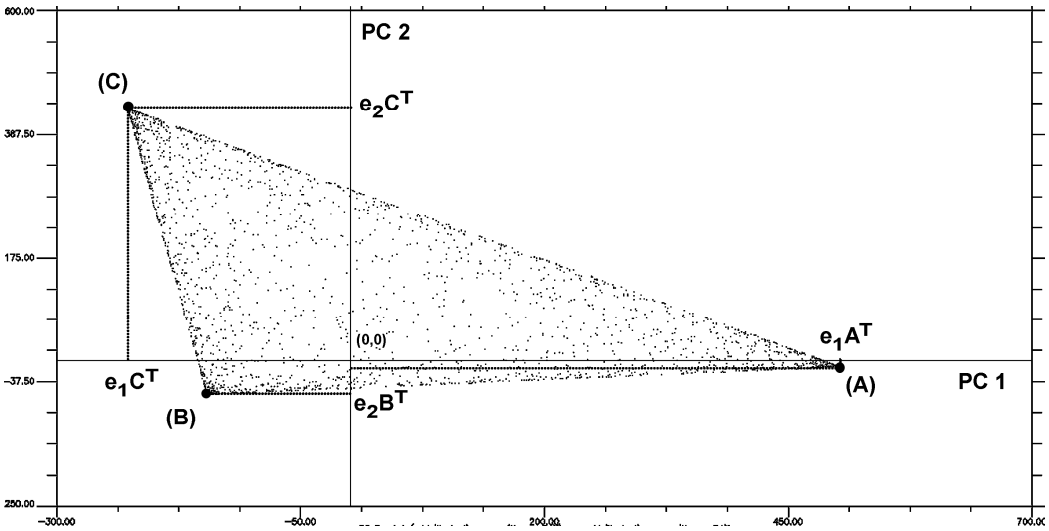
Το πρόβλημα της μεθόδου SEE, έγκειται στο γεγονός ότι στην περίπτωση που ένας καθαρός στόχος είναι κοντά στη μέση τιμή του διαγράμματος συσχέτισης των καναλιών, δεν θα ανιχνευθεί. Ένα παράδειγμα παρουσιάζεται στην παρακάτω εικόνα, όπου ο SEE θα ανιχνεύσει μόνο τους καθαρούς στόχους A και C, παρότι ο B είναι στην άκρη της χωρικής διάταξης (simplex). Αυτό συμβαίνει γιατί το μέγιστο στην πρώτη κύρια συνιστώσα είναι ο καθαρός στόχος A και το ελάχιστο ο C, ενώ στη δεύτερη, το μέγιστο είναι ο καθαρός στόχος C και το ελάχιστο ο A.



Εικόνα 3.3. : Διάγραμμα διασποράς εικόνας με 3 καθαρούς στόχους, στις 2 πρώτες κύριες συνιστώσες της με ένα καθαρό στόχο κοντά στη μέση τιμή.

Πηγή : Simple Endmember Extraction Methods Using Transformed Components for Hyperspectral Images

Γίνεται φανερό, ότι αν στραφεί όλο το σύμπλεγμα, ο καθαρός στόχος Β θα ανιχνευθεί. Έτσι χρειάζεται να εφαρμοστεί μια μέθοδος όπου να στρέφει τους άξονες χωρίς να επηρεάζεται σημαντικά η μεταβλητότητα των δεδομένων. Αυτό μπορεί να επιτευχθεί με την προσθήκη, στην αρχική εικόνα, εικονοστοιχείων με την φασματική υπογραφή του πρώτου καθαρού στόχου. Επομένως μπορεί να επιτευχθεί η επιθυμητή στροφή, αν προστεθεί στην αρχική εικόνα η φασματική υπογραφή του εικονοστοιχείου με τη μεγαλύτερη τιμή στην πρώτη κύρια συνιστώσα. Μετά από πειραματικές παρατηρήσεις, προέκυψε ότι, αν προστεθεί στην εικόνα άλλο 300% με την συγκεκριμένη φασματική υπογραφή, προκύπτει η επιθυμητή στροφή, χωρίς να επιβαρύνεται με περιττές πράξεις ο αλγόριθμος. Αυτή την τεχνική υιοθέτησε η μέθοδος Enhanced - Simple Endmember Extraction (E-SEE). Η εικόνα που εμφανίζει τη στροφή των αξόνων που επιχειρεί η μέθοδος E-SEE απεικονίζεται παρακάτω (εικόνα 3.4):



Εικόνα 3.4. : Στραμμένο διάγραμμα διασποράς εικόνας με 3 καθαρούς στόχους, στις 2 πρώτες κύριες συνιστώσες της με ένα καθαρό στόχο κοντά στη μέση τιμή.

Πηγή : Simple Endmember Extraction Methods Using Transformed Components for Hyperspectral Images

Στην εικόνα 3.4 φαίνεται καθαρά πως η στροφή των δεδομένων ως προς τους κύριους άξονες της μετασχηματισμένης τηλεπισκοπικής απεικόνισης, αναδεικνύει και τον καθαρό στόχο Β ο οποίος εμφανίζει ελάχιστη τιμή στη δεύτερη συνιστώσα.

3.7. Μέθοδος Εκτίμησης Αφθονίας

Η μέθοδος εκτίμησης της αφθονίας των καθαρών στόχων μιας εικόνας σε κάθε εικονοστοιχείο, αξιοποιεί το γεγονός ότι εφόσον όλα τα εικονοστοιχεία της εικόνας περιέχουν σε κάποιο ποσοστό κάθε καθαρό στόχο, αποτελούν γραμμικό συνδυασμό των τελευταίων. (Karathanassi V. et al. 2011)

Επομένως, αν ληφθούν οι φασματικές υπογραφές των καθαρών στόχων, ως εξισώσεις παρατήρησης, σύμφωνα με τον τύπο:

$$b_1 = a_1 * e_{1,1} + a_2 * e_{1,2} + ... + a_p * e_{1,p}$$
$$b_2 = a_1 * e_{2,1} + a_2 * e_{2,2} + ... + a_p * e_{2,p}$$
$$.$$
$$.$$
$$.$$
$$b_n = a_1 * e_{n,1} + a_2 * e_{n,2} + ... + a_p * e_{n,p}$$

όπου, b_i συμβολίζεται η τιμή του προς επεξεργασία εικονοστοιχείου,

a_j είναι η τιμή αφθονίας του καθαρού στόχου με αριθμό j που παίρνει τιμές από 1 έως p, και

$e_{i,j}$ είναι η τιμή ανακλαστικότητας του καθαρού στόχου με αριθμό j, στο κανάλι i.

Με αυτόν τον τρόπο, δημιουργείται ένα γραμμικό μοντέλο, για κάθε εικονοστοιχείο, που περιγράφεται από τον τύπο:

$$A\hat{x} = B$$

όπου, ο πίνακας A είναι ο πίνακας με μέγεθος n (πλήθος καναλιών) × p (πλήθος καθαρών στόχων), που αποτελεί τον πίνακα των εξισώσεων παρατήρησης του μοντέλου,

ο πίνακας x αποτελεί τον πίνακα με τις τιμές αφθονίας του κάθε καθαρού στόχου στο εν λόγω εικονοστοιχείο και

ο πίνακας B αποτελεί τον πίνακα με τις τιμές του προς επεξεργασία εικονοστοιχείου στα n κανάλια της απεικόνισης.

Έτσι, για να υπολογιστεί ο πίνακας x, ο παραπάνω τύπος διαμορφώνεται ως εξής:

$$\hat{x} = (A^T A)^{-1} A^T B$$

όπου, ο πίνακας x είναι ο πίνακας με τις τιμές αφθονίας των καθαρών στόχων για κάθε εικονοστοιχείο,

ο πίνακας A συμβολίζει τον πίνακα των εξισώσεων παρατήρησης, δηλαδή τον πίνακα με τις φασματικές υπογραφές των καθαρών στόχων,

ο πίνακας A^T είναι ο ανάστροφος του πίνακα A και

ο πίνακας B αποτελείται από τις τιμές του εικονοστοιχείου σε κάθε κανάλι της αρχικής εικόνας.

Επομένως, το αποτέλεσμα της επίλυσης του γραμμικού συστήματος μπορεί να παρουσιαστεί με τη μορφή μιας απεικόνισης. Το κάθε κανάλι της εικόνας αυτής, θα αποτελείται από τις τιμές αφθονίας του αντίστοιχου καθαρού στόχου. Έτσι, προκύπτει μια απεικόνιση με αριθμό καναλιών όσοι οι καθαροί στόχοι, στην οποία το κάθε κανάλι αποτελείται, ουσιαστικά, από τους συντελεστές αφθονίας του κάθε καθαρού στόχου, στο κάθε εικονοστοιχείο. Επομένως, το κάθε κανάλι της νέας απεικόνισης αποτελεί τον χάρτη αφθονίας του αντίστοιχου καθαρού στόχου, παρέχοντας έτσι τη δυνατότητα οπτικοποίησης της ύπαρξης ή όχι ενός καθαρού στόχου, καθώς και την τιμή του συντελεστή αφθονίας του.

4. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

4.1. Γενικά

Η γλώσσα προγραμματισμού C είναι μια γενικής χρήσης γλώσσα η οποία δημιουργήθηκε από τον Dennis Ritchie κατά την τετραετία 1969 με 1973. Λόγω του σχεδιασμού της που υιοθετούσε στοιχεία συνηθισμένων εντολών μηχανής, αρχικά χρησιμοποιήθηκε για τη μετατροπή προγραμμάτων που ήταν γραμμένα σε assembly language (μια γλώσσα πολύ κοντά στην γλώσσα μηχανής), όπως το λογισμικό Unix. Η C είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού καθώς μπορεί να χρησιμοποιηθεί σε όλες σχεδόν τις αρχιτεκτονικές υπολογιστών.

Η ταχύτητα που παρέχει η συγκεκριμένη γλώσσα και οι σχεδόν μηδενικοί περιορισμοί που θέτει, σε συνδυασμό με την ύπαρξη εργαλείων και βιβλιοθηκών που επιτρέπουν την παράλληλη εκτέλεση του κώδικα της, είναι οι κυριότεροι λόγοι για τους οποίους επιλέχθηκε για την συγκεκριμένη εργασία.

4.2. Ιστορικά στοιχεία

Σε πρώτη φάση, η C αναπτύχθηκε στα AT&T Bell Labs ανάμεσα στο 1969 και το 1973, σύμφωνα με τον D. Ritchie, η πιο δημιουργική περίοδος υπήρξε το 1972. Η νέα γλώσσα ονομάστηκε "C" λόγω του ότι πολλά από τα χαρακτηριστικά της προήλθαν από μια παλαιότερη γλώσσα, η οποία ονομαζόταν "B".

Μέχρι το 1973, η C είχε γίνει αρκετά ισχυρή και αποτελεσματική, ώστε το μεγαλύτερο μέρος του πυρήνα του UNIX (UNIX kernel), γραμμένο αρχικά σε PDP-11/20 assembly, επανεγράφηκε σε C. Ήταν ένας από τους πρώτους πυρήνες που υλοποιήθηκε σε μια γλώσσα διαφορετική της assembly.

Το 1978, ο Dennis Ritchie και ο Brian Kernighan δημοσίευσαν την πρώτη έκδοση του βιβλίου "The C Programming Language" (Dennis Ritchie et al., 1988). Το συγκεκριμένο βιβλίο, γνωστό στους προγραμματιστές της C ως "K&R", χρησίμευσε πολλά χρόνια ως ανεπίσημος ορισμός της γλώσσας. Η έκδοση της C που περιγράφει αναφέρεται συνήθως ως "K&R C." ή "Common C". (Η δεύτερη έκδοση του βιβλίου καλύπτει το μεταγενέστερο πρότυπο ANSI για τη C (ANSI C standard).)

Η K&R C συχνά λογίζεται ως το βασικό μέρος της γλώσσας που πρέπει να υποστηρίζει ένας μεταγλωττιστής της C. Για αρκετά χρόνια, ακόμη και μετά την εισαγωγή της ANSI C, θεωρούνταν ο "ελάχιστος συνήθης παρονομαστής" στον οποίο έπρεπε να προσαρμοστούν οι προγραμματιστές της C σε περιπτώσεις κατά τις οποίες ήταν επιθυμητή η μέγιστη μεταφερσιμότητα (portability), καθώς δεν είχαν ενημερωθεί όλοι οι μεταγλωττιστές για πλήρη υποστήριξη της ANSI C. Επίσης, με προσοχή, ο κώδικας σε K&R C μπορούσε να γραφεί ώστε να είναι σύμφωνος και με το πρότυπο ANSI.

Το 1983, το American National Standards Institute (ANSI) όρισε επιτροπή, τη X3J11, για να δώσει ένα σύγχρονο, πλήρη ορισμό της C. Μετά από μακρά και επίπονη επεξεργασία, το πρότυπο (standard) ολοκληρώθηκε το 1989 και επικυρώθηκε ως as ANSI X3.159-1989 "Programming Language C." Η συγκεκριμένη έκδοση της γλώσσας ονομάζεται συχνά ANSI C, ή ορισμένες φορές C89 (για να διαχωρίζεται από τη C99).

Ένας από τους στόχους της διαδικασίας δημιουργίας του προτύπου ANSI για τη C ήταν να δημιουργήσει ένα υπερσύνολο της K&R C, το οποίο θα απορροφούσε πολλά χαρακτηριστικά που είχαν εισαχθεί στην πορεία. Παρόλα αυτά, η επιτροπή συμπεριέλαβε και ορισμένα νέα χαρακτηριστικά, όπως function prototypes (δανεισμένα από τη C++), και ένα πιο ικανό προεπεξεργαστή (preprocessor). Η σύνταξη για τους ορισμούς παραμέτρων άλλαξε επίσης, ώστε να αντικατοπτρίζει το στυλ της C++.

Μετά τη διαδικασία καθορισμού του προτύπου ANSI, ο ορισμός της γλώσσας C παρέμενε σχετικά σταθερός για ορισμένο καιρό, ενώ η C++ συνέχιζε να αναπτύσσεται. Ωστόσο, το πρότυπο επανεξετάστηκε προς το τέλος της δεκαετίας του '90, γεγονός που οδήγησε στην έκδοση του ISO 9899:1999 το 1999. Το πρότυπο αυτό συχνά αναφέρεται ως "C99". Υιοθετήθηκε ως πρότυπο ANSI το Μάρτιο του 2000.

Ο GCC και μερικοί άλλοι C compilers υποστηρίζουν πλέον τα περισσότερα χαρακτηριστικά του C99. Ωστόσο, υπάρχει μικρότερη υποστήριξη από εταιρίες όπως η Microsoft και η Borland που εστίασαν περισσότερο στη C++, καθώς η C++ παρέχει παρόμοια λειτουργικότητα. Ακόμη και ο GCC με την εκτεταμένη υποστήριξη του C99 ακόμη δεν προσεγγίζει μια πλήρως συμβατή υλοποίηση, ορισμένα χαρακτηριστικά-κλειδιά λείπουν ή δεν λειτουργούν σωστά.

4.3. Παράλληλος Προγραμματισμός

4.3.1. Γενικά

Παράλληλος προγραμματισμός είναι η μορφή του προγραμματισμού στον οποίο πολλοί υπολογισμοί γίνονται ταυτόχρονα, λειτουργώντας με τη λογική ότι τα μεγάλα προβλήματα μπορούν να χωριστούν σε μικρότερα τα οποία λύνονται παράλληλα.

Για πολλά χρόνια οι κατασκευαστές υπολογιστών μεγιστοποιούσαν τις επιδόσεις των επεξεργαστών με την αύξηση της συχνότητας με την οποία αυτοί λειτουργούσαν. Όταν όμως έφτασαν σε ένα σημείο όπου η κατανάλωση ενέργειας και η θερμοκρασία των επεξεργαστών έφταναν σε πολύ υψηλά επίπεδα ήρθαν στο προσκήνιο οι επεξεργαστές με περισσότερα από ένα νήματα. Έτσι λοιπόν ο παράλληλος προγραμματισμός μετατράπηκε από ένα αντικείμενο που χρησιμοποιούσαν οι υπερυπολογιστές, σε εργαλείο για την επιτάχυνση προγραμμάτων σε προσωπικούς ή και φορητούς υπολογιστές.

Τα παράλληλα προγράμματα είναι πιο δύσκολα στην συγγραφή, από τα σειριακά, αφού το γεγονός ότι πολλές διεργασίες γίνονται ταυτόχρονα εισάγει νέες κατηγορίες σφαλμάτων με πιο συνηθισμένη την κατάσταση συναγωνισμού. Η επικοινωνία και ο συγχρονισμός μεταξύ των νημάτων είναι το μεγαλύτερο εμπόδιο για την επίτευξη καλών επιδόσεων, γι’ αυτό πρέπει να χρησιμοποιούνται μόνο όπου είναι απαραίτητο.

4.3.2. Ο νόμος του Amdahl.

Θεωρητικά, η επιτάχυνση ενός παράλληλου προγράμματος είναι γραμμική, δηλαδή με τον διπλασιασμό των νημάτων θα μειωνόταν στο μισό ο χρόνος εκτέλεσης του προγράμματος. Όμως κάτι τέτοιο δεν ισχύει. Το πόσο θα μειωθεί ο χρόνος εκτέλεσης ενός παράλληλου προγράμματος ακολουθεί τον νόμο του Amdahl, ο οποίος διατυπώθηκε την δεκαετία του 60 από τον Gene Amdahl.

Σύμφωνα με το νόμο του Amdahl, λοιπόν, αν ένα μικρό κομμάτι του προγράμματος δεν μπορεί να παραλληλοποιηθεί, τότε αυτό το κομμάτι θέτει το κάτω όριο στον εκτιμώμενο χρόνο εκτέλεσης του προγράμματος. Ο νόμος του Άμνταλ συνήθως ορίζεται με χρήση του παρακάτω μαθηματικού τύπου:

$$\frac{1}{(1-P)+\frac{P}{N}}$$

Στον τύπο αυτό, η μεταβλητή **P** δηλώνει το ποσοστό των συνολικών υπολογισμών του προγράμματος οι οποίοι μπορούν να παραλληλοποιηθούν και η μεταβλητή **N** το πλήθος των διαθέσιμων επεξεργαστών.

4.3.3. Είδη παράλληλου προγραμματισμού.

Υπάρχουν διάφορα είδη παράλληλου προγραμματισμού, τα κυριότερα όμως είναι δύο, ο παραλληλισμός με βάση τα δεδομένα (data parallelism) και ο παραλληλισμός με βάσης τις εργασίες (task parallelism).

Ο παραλληλισμός με βάση τα δεδομένα συνήθως προτιμάται σε περιπτώσεις επαναληπτικών διαδικασιών και επικεντρώνεται στο να χωρίσει τα δεδομένα μιας εργασίας στα διάφορα νήματα. Έτσι εκτελείται η ίδια διαδικασία όσες φορές απαιτείται, με το κάθε νήμα να αναλαμβάνει τον φόρτο εργασίας που του αναλογεί. Η κατανομή του φόρτου μπορεί να γίνει είτε με μοίρασμα των επαναλήψεων σε όλα τα νήματα, είτε λαμβάνοντας υπ' όψη τον φόρτο εργασίας που μπορεί να έχει ήδη κάποιο νήμα.

Ο παραλληλισμός με βάση τις εργασίες είναι το είδος του παράλληλου προγραμματισμού όπου το κάθε νήμα εκτελεί τελείως διαφορετικούς υπολογισμούς με διαφορετικά ή ακόμα και τα ίδια δεδομένα. Έτσι όταν υπάρχουν περισσότερες από μια εργασίες οι οποίες είναι ανεξάρτητες μεταξύ τους, ο τρόπος αυτός παραλληλισμού μπορεί να χρησιμοποιηθεί.

4.3.4. Καταστάσεις Συναγωνισμού

Ένα πρόβλημα που συναντάται συχνά στον παράλληλο προγραμματισμό είναι η κατάσταση συναγωνισμού (Race conditions). Όταν υπάρχουν πολλά νήματα τα οποία μοιράζονται την ίδια μνήμη, και το καθένα αλλάζει τη μνήμη αυτή, υπάρχει μεγάλη πιθανότητα να γίνει κάποια αλλαγή χωρίς να το επιδιώκει ο προγραμματιστής, και αυτό γιατί οι διαδικασίες δεν γίνονται πάντα με τη σειρά με την οποία είναι γραμμένες στον πηγαίο κώδικα αλλά ανάλογα με το φόρτο εργασίας του κάθε νήματος. Έτσι μπορεί το πρόγραμμα να εκτελεστεί σωστά αρκετές φορές όμως να μην λειτουργήσει για κάποιον λόγο ο οποίος φαινομενικά είναι άσχετος, ακόμα και με το ίδιο το πρόγραμμα. Αυτού του είδους τα σφάλματα είναι πιο δύσκολα να επιλυθούν καθώς μπορεί να υπάρχει δυσκολία ακόμα και στο να αναπαραχθούν.

5. ΠΡΟΣΘΕΤΑ ΕΡΓΑΛΕΙΑ

5.1. Η Βιβλιοθήκη GDAL

Η βιβλιοθήκη GDAL (Geospatial Data Abstraction Library) είναι ένα πρόγραμμα τερματικού (δηλαδή χωρίς γραφικό περιβάλλον), ανοιχτού κώδικα, το οποίο μετατρέπει αρχεία εικόνας από έναν οποιοδήποτε τύπο στο επιθυμητό. Αυτό γίνεται μέσω μια εντολής τερματικού η οποία περιλαμβάνει την εντολή την ίδια και τις παραμέτρους της όπως ο τύπος του επιθυμητού αρχείου και προαιρετικά διάφορες διευκρινίσεις σε αυτό, καθώς και την διαδρομή (Path) του αρχείου εισόδου και εξόδου. Η συγκεκριμένη βιβλιοθήκη υποστηρίζει μια μεγάλη ποικιλία αρχείων όπως φαίνεται και παρακάτω :

```
The following format drivers are configured and support output:
  FITS: Flexible Image Transport System
  GMT: GMT NetCDF Grid Format
  netCDF: Network Common Data Format
  VRT: Virtual Raster
  GTiff: GeoTIFF
  NITF: National Imagery Transmission Format
  HFA: Erdas Imagine Images (.img)
  ELAS: ELAS
  AAIGrid: Arc/Info ASCII Grid
  DTED: DTED Elevation Raster
  PNG: Portable Network Graphics
  JPEG: JPEG JFIF
  MEM: In Memory Raster
  GIF: Graphics Interchange Format (.gif)
  XPM: X11 PixMap Format
  BMP: MS Windows Device Independent Bitmap
  PCIDSK: PCIDSK Database File
  PCRaster: PCRaster Raster File
  ILWIS: ILWIS Raster Map
  SGI: SGI Image File Format 1.0
  SRTMHGT: SRTMHGT File Format
  Leveller: Leveller heightfield
  Terragen: Terragen heightfield
  ISIS2: USGS Astrogeology ISIS cube (Version 2)
  ERS: ERMapper .ers Labelled
  JP2OpenJPEG: JPEG-2000 driver based on OpenJPEG library
  FIT: FIT Image
  RMF: Raster Matrix Format
  WMS: OGC Web Map Service
  RST: Idrisi Raster A.1
  INGR: Intergraph Raster
  GSAG: Golden Software ASCII Grid (.grd)
```

```
GSBG: Golden Software Binary Grid (.grd)
R: R Object Data Store
PNM: Portable Pixmap Format (netpbm)
ENVI: ENVI .hdr Labelled
EHdr: ESRI .hdr Labelled
PAux: PCI .aux Labelled
MFF: Vexcel MFF Raster
MFF2: Vexcel MFF2 (HKV) Raster
BT: VTP .bt (Binary Terrain) 1.3 Format
LAN: Erdas .LAN/.GIS
IDA: Image Data and Analysis
GTX: NOAA Vertical Datum .GTX
NTv2: NTv2 Datum Grid Shift
USGSDEM: USGS Optional ASCII DEM (and CDED)
ADRG: ARC Digitized Raster Graphics
BLX: Magellan topo (.blx)
Rasterlite: Rasterlite
SAGA: SAGA GIS Binary Grid (.sdat)
KMLSUPEROVERLAY: Kml Super Overlay
XYZ: ASCII Gridded XYZ
HF2: HF2/HFZ heightfield raster
ZMap: ZMap Plus Grid
```

Εικόνα 5.1.Τύποι αρχείων που υποστηρίζονται από το πρόγραμμα GDAL
Πηγή : GDAL tutorial

Στα προγράμματα της εργασίας μετατρέπεται οποιοδήποτε από τα παραπάνω είδη αρχείων σε αρχείο του λογισμικού ERMapper, καθώς ο συγκεκριμένος τύπος αρχείου χωρίζει την εικόνα σε δύο ξεχωριστά αρχεία, το ένα περιέχει την επικεφαλίδα (header) της εικόνας σε αρχείο κειμένου και το δεύτερο την εικόνα σε δυαδική (binary) μορφή. Με αυτόν τον τρόπο δίνεται η δυνατότητα στο πρόγραμμα να επεξεργαστεί την εικόνα πιο εύκολα καθώς ακολουθεί το πρότυπο BIL (Band Interleaved by Line), σύμφωνα με όσα έχουν αναφερθεί παραπάνω.

Η χρήση αυτής της βιβλιοθήκης γίνεται μέσω μιας εντολής συστήματος, η οποία θέτει σε αναμονή το πρόγραμμα που την έδωσε έως ότου αυτή ολοκληρωθεί. Έτσι το πρόγραμμα της εργασίας περιμένει να ολοκληρωθεί η μετατροπή της εικόνας εισόδου από την GDAL και μετά εκτελεί τον υπόλοιπο αλγόριθμο. Για να λειτουργήσουν σωστά τα προγράμματα χρειάζεται να είναι εγκατεστημένο το πρόγραμμα GDAL στον υπολογιστή που υλοποιεί τους αλγόριθμούς.

Αξίζει να αναφερθεί ότι η βιβλιοθήκη χρησιμοποιείται από πολλά λογισμικά ψηφιακής τηλεπισκόπησης.

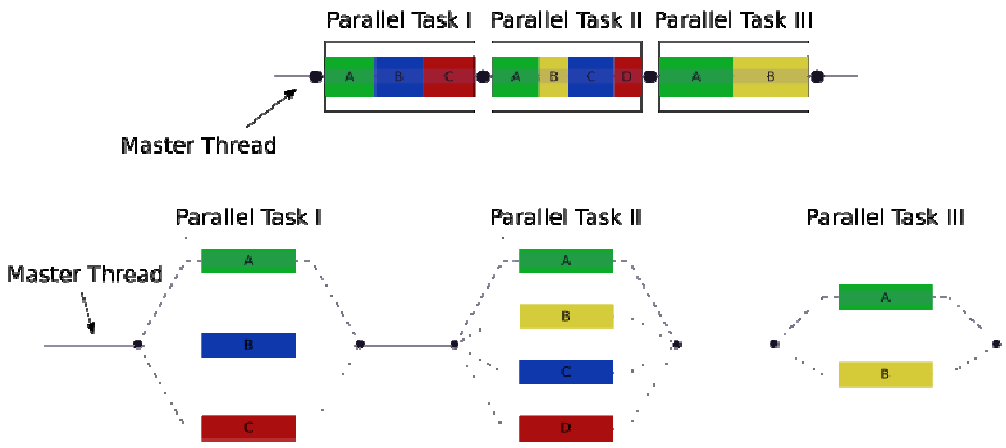
5.2. Η Βιβλιοθήκη OpenMP

Η βιβλιοθήκη OpenMP (Open Multiprocessing) είναι μια βιβλιοθήκη που χρησιμοποιείται ως εξωτερική βιβλιοθήκη στις γλώσσες C, C++ και Fortran ή οποία μέσω συγκεκριμένων εντολών που εισάγει ο προγραμματιστής χειροκίνητα, αξιοποιεί περισσότερους από έναν επεξεργαστές για την εκτέλεση του κώδικα, εάν, προφανώς, αυτοί είναι διαθέσιμοι.

Η συγκεκριμένη βιβλιοθήκη αποτελείται από εντολές οι οποίες δίνονται στο μεταγλωττιστή μέσω των «# pragmas», δηλαδή μέσω εντολών που διαβάζονται μόνο από τον προεπεξεργαστή. Έτσι ο προγραμματιστής μπορεί να δίνει εντολές οι οποίες δεν μπερδεύονται με τον πηγαίο σειριακό κώδικα και υπάρχει η δυνατότητα μεταγλώττισης του παράλληλου κώδικα ακόμα και με σειριακό μεταγλωττιστή, προφανώς δημιουργώντας σειριακό πρόγραμμα.

Η OpenMP χρησιμοποιεί την τεχνολογία της πολυνημάτωσης (multithreading), μια μέθοδο όπου το κύριο νήμα (master thread) της διεργασίας (δηλαδή το νήμα που εκτελείται σειριακά) σπάει σε έναν συγκεκριμένο αριθμό από δευτερεύοντα νήματα στα οποία μοιράζεται ο φόρτος εργασίας. Έτσι, τρέχοντας τον κώδικα παράλληλα και με την προϋπόθεση ότι ο φόρτος εργασίας είναι ισορροπημένος, ο χρόνος ολοκλήρωσης του συγκεκριμένου κομματιού του κώδικα μειώνεται σημαντικά.

Το κομμάτι του κώδικα το οποίο προορίζεται για παράλληλη εκτέλεση αποκτά ειδική σήμανση, με εντολή στον προεπεξεργαστή που οδηγεί στη δημιουργία των νημάτων πριν την εκτέλεση της περιοχής αυτής του κώδικα. Κάθε νήμα λαμβάνει ένα id το οποίο μπορεί να διαβαστεί από τον κώδικα με κατάλληλη εντολή. Το id αυτό είναι ένας ακέραιος αριθμός με το κύριο νήμα να λαμβάνει το id με αριθμό 0. Μετά την παράλληλη εκτέλεση της περιοχής τα νήματα συγχωνεύονται με το κύριο νήμα ξανά για να συνεχιστεί η σειριακή εκτέλεση του προγράμματος, όπως φαίνεται και στην παρακάτω εικόνα.



Εικόνα 5.2. Διάγραμμα ροής παράλληλου προγράμματος
Πηγή : Wikipedia

Παραλληλοποιώντας, χωρίς τις κατάλληλες παραμέτρους, ένα κομμάτι του κώδικα θα οδηγήσει απλά στην εκτέλεση αυτού του κομματιού του κώδικα από όλα τα νήματα, δηλαδή στην εκτέλεση του ίδιου κώδικα πολλές φορές. Γι' αυτό υπάρχουν δομές μοιράσματος του φόρτου, έτσι ώστε κάθε νήμα να εκτελέσει το κομμάτι που του αναλογεί. Η βιβλιοθήκη OpenMP δίνει την δυνατότητα και για παραλληλοποιήση και ως προς τις εργασίες και ως προς τα δεδομένα, όπως έχει αναφερθεί παραπάνω.

5.2.1. Δημιουργία νημάτων

Για την δημιουργία των Νημάτων χρησιμοποιείται η εντολή # pragma omp parallel έτσι ώστε να σημανθεί ότι η περιοχή που ακολουθεί θα εκτελεστεί από όλους τους πυρήνες, όπως φαίνεται και την παρακάτω εικόνα.

```
# include <omp.h>
# include <stdio.h>

int main(void)
{
  #pragma omp parallel
    printf("Hello, world.\n");
  return 0;
}
```

Εικόνα 5.3. Παράδειγμα κώδικα του OpenMP

Έτσι, αφού το κομμάτι του κώδικα που τυπώνει στην οθόνη την φράση "Hello World" εκτελείται από όλα τα νήματα, το αποτέλεσμα του παραπάνω προγράμματος σε έναν υπολογιστή με δύο νήματα θα είναι το παρακάτω :

```
Hello, world.
Hello, world.
```

Εικόνα 5.4. Αποτέλεσμα του παραδείγματος της εικόνας 5.3.

5.2.2. Δομές κατανομής φόρτου εργασίας (Work-sharing Constructs)

Στο παράδειγμα τις προηγούμενης ενότητας η περιοχή που εκτελέστηκε παράλληλα, έτρεξε τόσες φορές όσα και τα νήματα του υπολογιστή. Όμως ένα μεγάλο κομμάτι των εφαρμογών που χρειάζονται παραλληλοποίηση, πρέπει να μοιράζουν την εργασία στα νήματα και όχι να εκτελούν την ίδια εργασία πολλές φορές. Έτσι η βιβλιοθήκη OpenMP προσφέρει τέτοια εργαλεία, όπως τα παρακάτω:

- *omp for* : Χρησιμοποιείται για να μοιράσει επαναληπτικές διαδικασίες τύπου «for» στα διαθέσιμα νήματα.
- *sections* : Χωρίζει τον κώδικα σε blocks που το καθένα εκτελείται από ένα νήμα.
- *single*: Εκτελεί το κομμάτι που ακολουθεί μία μόνο φορά, χρησιμοποιώντας το νήμα που θα φτάσει σε αυτό το σημείο πρώτο, αναγκάζοντας τα υπόλοιπα νήματα να περιμένουν την ολοκλήρωση του.
- *master* : Όμοια με το single, με τη διαφορά ότι το νήμα που θα εκτελέσει τον κώδικα θα είναι το κύριο.

Ένα παράδειγμα δομής κατανομής του φόρτου εργασίας είναι το παρακάτω. Σε αυτό, το κάθε νήμα εκτελεί N/nthreads επαναλήψεις, όπου nthreads είναι ο αριθμός των νημάτων δηλαδή ο αριθμός των πυρήνων του επεξεργαστή.

```
int main(int argc, char *argv[]) {
    const int N = 100000;
    int i, a[N];

    #pragma omp parallel for
    for (i = 0; i < N; i++)
        a[i] = 2 * i;

    return 0;
}
```

Εικόνα 5.5. Παράδειγμα δομής κατανομής του φόρτου εργασίας

Εδώ αξίζει να αναφερθεί ότι μπορούν να παραλληλοποιηθούν μόνο επαναληπτικές διαδικασίες για τις οποίες είναι εξ αρχής γνωστός ο αριθμός των επαναλήψεων, έτσι ώστε να μπορεί αυτός ο αριθμός να μοιραστεί ανάλογα με τον αριθμό των νημάτων. Δηλαδή δεν μπορεί να παραλληλοποιηθεί διαδικασία η οποία δεν έχει έναν συγκεκριμένο φόρτο εργασίας και μπορεί να τελειώσει απροσδόκητα. (Chapman B. et al. 2008)

5.2.3. Παράμετροι (clauses) του OpenMP

Η βιβλιοθήκη OpenMP προσφέρει εργαλεία, που είτε είναι απαραίτητα για την σωστή εκτέλεση του προγράμματος, είτε για την επίτευξη καλύτερων επιδόσεων από θέμα ταχύτητας. Τα εργαλεία αυτά ρυθμίζονται με την εισαγωγή παραμέτρων που αφορούν τον τρόπο λειτουργίας των μεταβλητών, του συγχρονισμού των νημάτων, τον προγραμματισμό του μοιράσματος του φόρτου εργασίας, και άλλα.

Ένα πολύ σημαντικό στοιχείο που οφείλει ο χρήστης να συμπεριλάβει είναι ο τρόπος με τον οποίο τα νήματα αποκτούν πρόσβαση στις μεταβλητές. Η λανθασμένη δήλωση του τύπου των μεταβλητών, δηλαδή το εάν αυτές θα είναι ατομικές (private) ή μοιραζόμενες (shared), μπορεί να οδηγήσει σε μείωση της απόδοσης, απώλεια δεδομένων ή και συνηθέστερα σε race conditions.

Τύποι μεταβλητών

Οι μοιραζόμενες μεταβλητές (shared) διαβάζονται και ενημερώνονται από όλα τα νήματα ταυτόχρονα. Από προεπιλογή όλες οι μεταβλητές σε μια δομή καταμερισμού εργασίας είναι μοιραζόμενες, εκτός της μεταβλητής επανάληψης η οποία είναι ατομική. Οι ατομικές μεταβλητές (private) είναι αποκλειστικές για κάθε νήμα, καθώς κάθε νήμα κρατά ένα αντίγραφο από κάθε μια από αυτές ως προσωρινή μεταβλητή. Αυτό όμως έχει ως συνέπεια τη μη διατήρηση της τιμής της ατομικής μεταβλητής μετά από την παράλληλη περιοχή, καθώς υπάρχουν παραπάνω από μια τιμές για αυτήν, μια για κάθε νήμα. Για αυτό το λόγο υπάρχουν και οι τύποι firstprivate και lastprivate όπου η ατομική μεταβλητή παίρνει τιμή ίση με αυτήν που είχε πριν την παράλληλη περιοχή ή την τελευταία τιμή που της δόθηκε εντός της παράλληλης περιοχής, αντίστοιχα. (Chapman B. et al. 2008)

Συγχρονισμός Νημάτων

Ο καθορισμός του τρόπου με τον οποίο τα νήματα συγχρονίζονται σε πολλές περιπτώσεις είναι, όχι μόνο απαραίτητος για την επίτευξη καλύτερων επιδόσεων κατά την εκτέλεση του προγράμματος αλλά και για την εξασφάλιση της σωστής λειτουργίας του, σε κάθε περίπτωση. Για το λόγο αυτό η βιβλιοθήκη OpenMP περιλαμβάνει παραμέτρους για τον τρόπο και την σειρά με την οποία θα λειτουργήσουν τα νήματα, οι κυριότερες από τις οποίες είναι οι παρακάτω:

- *critical:* το κομμάτι του κώδικα που ακολουθεί την παράμετρο αυτή θα εκτελεστεί από ένα νήμα κάθε φορά και όχι ταυτόχρονα από όλα τα νήματα μαζί.
- *ordered*: η επαναληπτική διαδικασία που ακολουθεί αυτήν την εντολή θα εκτελεστεί με την σειρά που θα εκτελούνταν εάν το πρόγραμμα ήταν σειριακό.
- *barrier*: όταν συναντάται από ένα νήμα αυτή η εντολή, αυτό θα περιμένει μέχρι να φτάσουν σε αυτό το σημείο και όλα τα υπόλοιπα νήματα. Πολλές παράμετροι του OpenMP εισάγουν αυτόματα και αυτήν την παράμετρο στο τέλος τους.
- *nowait*: εξασφαλίζει την μη υλοποίηση του barrier κατά το τέλος της δομής κατανομής εργασίας, η οποία σε αντίθετη περίπτωση, θα το εισήγαγε αυτόματα.

Χρονοπρογραμματισμός νημάτων

- *schedule(type, chunk)*: Η παράμετρος αυτή χρησιμοποιείται σε work sharing construct και κυρίως σε επαναληπτικές διαδικασίες τύπου for. Οι επαναλήψεις μοιράζονται στα νήματα σύμφωνα με τα παρακάτω είδη χρονοπρογραμματισμού:

1. *static*: Με τη χρήση αυτής της παραμέτρου όλα τα νήματα παίρνουν ίσο αριθμό κομματιών το καθένα. Αυτή η παράμετρος είναι ορισμένη από προεπιλογή. Παρόλα αυτά δίνοντας τιμή στην παράμετρο chunk έχουμε τον αριθμό των επαναλήψεων που κατανέμονται σε κάθε νήμα, κάθε φορά μέχρι να τελειώσουν οι επαναλήψεις.
2. *dynamic*: Σε αυτήν την περίπτωση οι επαναλήψεις χωρίζονται σε μικρές ομάδες αυτόματα και κατανέμονται σε κάθε νήμα. Όταν ένα νήμα τελειώσει με τον αριθμό των επαναλήψεων που του έχει ανατεθεί

λαμβάνει άλλη μια τέτοια ομάδα μέχρι να τελειώσουν οι επαναλήψεις. Η παράμετρος chunk ορίζει το μέγεθος των ομάδων αυτών.

3. *guided*: Όμοια με την παράμετρο dynamic με την διαφορά ότι στην αρχή το μέγεθος του chunk είναι μεγάλο και όσο προχωρά η διαδικασία αυτό μικραίνει. Η παράμετρος chunk ορίζει τη μικρότερη ομάδα επαναλήψεων.

5.2.4. Πλεονεκτήματα και μειονεκτήματα του OpenMP

Η βιβλιοθήκη OpenMP παρουσιάζει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα με τη χρήση της στις γλώσσες C, C++ και Fortran, μερικά από τα οποία είναι :

Πλεονεκτήματα :

- Παραλληλοποίηση της διαδικασίας επιτυγχάνοντας σημαντική βελτίωση στον χρόνο εκτέλεσης, όταν χρησιμοποιείται σωστά.
- Μετά την μεταγλώττιση του προγράμματος δεν απαιτείται κάποιο πρόγραμμα ή βιβλιοθήκη για να λειτουργήσει σε κάθε υπολογιστή.
- Δεν χρειάζεται να ασχοληθεί κανείς με την ανταλλαγή μηνυμάτων μεταξύ των νημάτων. Έτσι η χρήση του καθίσταται αρκετά πιο απλή από άλλα συστήματα παραλληλοποίησης.
- Δυνατότητα εφαρμογής σε μικρό κομμάτι του κώδικα κάθε φορά για αποφυγή λαθών.
- Ενιαίος κώδικας για σειριακό και παράλληλο πρόγραμμα.
- Μικρή, εάν όχι καθόλου παρέμβαση στον σειριακό κώδικα.
- Δυνατότητα επιλογής μεταξύ γενικότερου ή ειδικού κώδικα για κάθε εφαρμογή.

Μειονεκτήματα

- Πιθανότητα εισαγωγής στον κώδικα σφαλμάτων (bugs) δύσκολων στην επίλυση, αλλά ακόμα και στην αναπαραγωγή καθώς και εισαγωγή καταστάσεων συναγωνισμού (race conditions),.
- Χρειάζεται κατάλληλο μεταγλωττιστή.
- Όταν χρησιμοποιείται σε υπολογιστές με ένα νήμα οι επιδόσεις μειώνονται σε σχέση με το σειριακό πρόγραμμα.
- Απουσία μηχανισμού χειρισμού λαθών.

6. ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ

Για την παρούσα διπλωματική εργασία δημιουργήθηκαν μια σειρά προγραμμάτων, τα οποία, όπως έχει ειπωθεί καλύπτουν την διαδικασία του φασματικού διαχωρισμού. Τα προγράμματα αυτά είναι υλοποιήσεις αλγορίθμων που είτε έχουν δημιουργηθεί από μέλη του Εργαστηρίου Τηλεπισκόπησης του Ε.Μ.Π. όπως ο ODM, ο SEE και ο SEE-E, είτε αποτελούν γνωστές μεθόδους, όπως PCA, MNF, SAD, AFEM κλπ. Ο φασματικός διαχωρισμός μπορεί να υλοποιηθεί με εκτέλεση των προγραμμάτων που δημιουργήθηκαν, με δύο εναλλακτικούς τρόπους. Είτε εκτελώντας προγράμματα που αναφέρονται σε κάθε βήμα του φασματικού διαχωρισμού, ή εκτελώντας ένα ενιαίο πρόγραμμα. Επίσης για τα βασικά εργαλεία/τεχνικές τα οποία χρησιμοποιούν οι μέθοδοι του κάθε βήματος, έχουν αναπτυχθεί και ξεχωριστά προγράμματα. Π.χ. η μέθοδος ODM περιλαμβάνει την PCA ή την MNF και γι αυτό έχουν αναπτυχθεί και ξεχωριστά προγράμματα για τις τελευταίες. Σε αυτή την ενότητα θα γίνει μια σύντομη αναφορά στο καθένα πρόγραμμα που δημιουργήθηκε. Αναλυτικότερα, τα προγράμματα που αναπτύχθηκαν είναι τα εξής:

- ODM - Υλοποίηση του αλγόριθμου Outlier Detection Method.
  - PCA - Υλοποίηση του Μετασχηματισμού Κυρίων Συνιστωσών.
  - MNF- Υλοποίηση του Μετασχηματισμού Ελαχιστοποίησης του Θορύβου.
- SEE - Υλοποίηση του αλγόριθμου Simple Endmember Extraction.
- SEE-E - Υλοποίηση του αλγόριθμου Simple Endmember Extraction – Enhanced.
  - SAD - Δημιουργία εικόνας με τιμές της φασματικές γωνίες δοσμένων φασματικών υπογραφών
  - SADtxt-Υπολογισμός φασματικών γωνιών μεταξύ δύο σετ φασματικών υπογραφών.
- AFEM - Υλοποίηση του αλγορίθμου Εκτίμησης της αφθονίας των καθαρών στόχων.
- Unmixing – Υλοποίηση της διαδικασίας του Φασματικού Διαχωρισμού σε ένα ενιαίο πρόγραμμα.

6.1. Εκτίμηση του Φασματικού Υπόχωρου του Σήματος

Για την εκτίμηση του πλήθους του φασματικού υπόχωρου του σήματος υλοποιήθηκε η μέθοδος Outlier Detection Method (ODM), όπως αυτή περιγράφεται θεωρητικά σε προηγούμενο κεφάλαιο. Όπως έχει ήδη αναφερθεί παραπάνω, η μέθοδος αυτή θεωρεί τον θόρυβο, ως το κύριο στατιστικό δεδομένο, ενώ το σήμα θεωρείται ανωμαλία του δεδομένου αυτού. Η ταξινόμηση των μετασχηματισμένων καναλιών σε θόρυβο ή σήμα γίνεται με βάση την απόσταση των καναλιών σε ένα διάγραμμα «διαφοράς κανονικοποιημένης μεταβλητότητας – αριθμού καναλιού».

Για τη διαδικασία αυτή, προφανώς πρέπει να προηγηθεί ο κατάλληλος μετασχηματισμός της εικόνας. Αυτός είναι είτε ο μετασχηματισμός ελαχιστοποίησης του θορύβου (MNF), όταν πρόκειται για εικόνα με πραγματικά δεδομένα όπου ο θόρυβος που περιέχεται δεν είναι δυνατόν να είναι λευκός, λόγω των συνθηκών προέλευσής του, είτε μετασχηματισμός κυρίων συνιστωσών (PCA), όταν πρόκειται για συνθετική εικόνα με λευκό θόρυβο. Στο πρόγραμμα αυτό ο χρήστης επιλέγει ανάλογα με τα δεδομένα εάν θα εκτελεστεί ο MNF ή η PCA. Επίσης έχει τη δυνατότητα να δώσει μετασχηματισμένη την εικόνα και το πρόγραμμα να μην εκτελέσει κανένα μετασχηματισμό στα αρχικά δεδομένα. Σε αυτή την περίπτωση ο χρήστης θα εκτελέσει εξωτερικά τα προγράμματα MNF ή PCA.

Για να επιτευχθεί η εκτίμηση του φασματικού υπόχωρου, δεδομένου ότι οι απεικονίσεις είναι μετασχηματισμένες, αρχικά πρέπει να υπολογιστούν τα στατιστικά της μετασχηματισμένης εικόνας, αρχίζοντας από το άθροισμα των τιμών των εικονοστοιχείων κάθε καναλιού και τη μέση τιμή του, όπως φαίνεται στο παρακάτω απόσπασμα του κώδικα.

```
rewind(fsignal);
for (i=0; i<rows; i++){
    fread (buffer, sizeof(float), cols*bands, fsignal);
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[(j*cols)+k];
                }
            }
        }
    }
```

Εικόνα 6.1. Απόσπασμα κώδικα υπολογισμού αθροίσματος καναλιών

Επόμενο βήμα είναι ο υπολογισμός της μεταβλητότητας του κάθε καναλιού με χρήση των τύπων της στατιστικής όπως και σε προηγούμενες περιπτώσεις. Το αντίστοιχο κομμάτι του κώδικα φαίνεται παρακάτω:

```
rewind(fsignal);
# pragma omp parallel default(none) private(i, j, k) shared(buffer,
mean, var, rows, cols, bands, fsignal, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fsignal);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                var[j]+=(buffer[(j*cols)+k]-
mean[j])*(buffer[(j*cols)+k]-mean[j]);
                }
            }
        }
    }
}

for (i=0; i<bands; i++){
    var[i]=sqrt(var[i]/(rows*cols-nvalue));
    }
```

Εικόνα 6.2. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης καναλιών

Επόμενο βήμα για την υλοποίηση του αλγόριθμου, είναι η κανονικοποίηση των τυπικών αποκλίσεων των καναλιών, στην περίπτωση που τα μετασχηματισμένα κανάλια είναι κανάλια MNF. Εάν αυτά προκύπτουν από μετασχηματισμό κυρίων συνιστωσών η κανονικοποίηση δεν είναι απαραίτητη. Η συγκεκριμένη εργασία παρουσιάζεται στο παρακάτω κομμάτι του κώδικα.

```
if (opt==1 || opt==2){
    varmax=0;
    varmin=5000;
    for (i=0; i<bands; i++){
        if (var[i]>varmax){
            varmax=var[i];
            }
        if (var[i]<varmin){
            varmin=var[i];
            }
        }
    fprintf(report, "varmax = %10.5lf, varmin= %5.5lf\n", varmax,
varmin);
    for (i=0; i<bands; i++){
        var[i]=(var[i]-varmin)/(varmax-varmin);
        }
    fprintf(report,"\nNormalised Variance :\n");
    for (i=0; i<bands; i++){
        fprintf(report,"Band %d : %10.5lf\n", i+1, var[i]);
        }
    }
```

Εικόνα 6.3. Απόσπασμα κώδικα κανονικοποίησης τυπικής απόκλισης καναλιών

Στη συνέχεια πρέπει να υπολογιστούν οι ευκλείδειες αποστάσεις της μεταβλητότητας δύο γειτονικών καναλιών στο διάγραμμα «διαφοράς κανονικοποιημένης μεταβλητότητας – αριθμού καναλιού» το οποίο έχει ως άξονα x τον αύξοντα αριθμό του πρώτου εκ των δύο γειτονικών καναλιών και άξονα ψ τη διαφορά της μεταβλητότητάς τους. Εφόσον τα κανάλια είναι γειτονικά, αρκεί να υπολογιστεί η απόσταση, όπου κατά ψ είναι η διαφορά της μεταβλητότητας των δύο καναλιών και κατά x η μονάδα. Το παρακάτω κομμάτι του κώδικα περιγράφει την συγκεκριμένη διαδικασία.

```
for (i=0; i<bands-1; i++){
    dist[i]=sqrt((dvar2[i]-dvar2[i+1])*(dvar2[i]-dvar2[i+1])+1);
    }
```

Εικόνα 6.4. Απόσπασμα κώδικα υπολογισμού ευκλείδειας απόστασης καναλιών

Πρέπει στη συνέχεια να υπολογιστεί το κατώφλι, το οποίο αν είναι μεγαλύτερο από την ευκλείδεια απόσταση μεταξύ δυο καναλιών, τα υπόλοιπα κανάλια θα θεωρηθούν ως θόρυβος. Αυτό, όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, γίνεται σύμφωνα με τον τύπο

$$Q_3 + 1.5 \times IRQ$$

όπου, Q_3 συμβολίζεται η τιμή του στοιχείου με αριθμό το 75% του συνόλου των στοιχείων του διαγράμματος «διαφοράς κανονικοποιημένης μεταβλητότητας – αριθμού καναλιού» αποστάσεων

και το μέγεθος IRQ υπολογίζεται από τον τύπο:

$$IRQ = Q_3 - Q_1$$

όπου, Q_3 συμβολίζεται η τιμή του στοιχείου με αριθμό το 75% του συνόλου των στοιχείων του διαγράμματος «διαφοράς κανονικοποιημένης μεταβλητότητας – αριθμού καναλιού» αποστάσεων

Q_1 η τιμή του στοιχείου με αριθμό το 25% του συνόλου των στοιχείων του διαγράμματος «διαφοράς κανονικοποιημένης μεταβλητότητας – αριθμού καναλιού» αποστάσεων

Το κατώφλι υπολογίζεται στο παρακάτω απόσπασμα του κώδικα.

```
threshold=dist[bands*1/4]+1.5*(fabs(dist[bands*1/4]-
                                                dist[bands*3/4]));
```

Εικόνα 6.5. Απόσπασμα κώδικα υπολογισμού κατωφλιού

Τέλος ακολουθεί η ταξινόμηση των ευκλείδειων αποστάσεων των καναλιών, σε πληροφορία ή θόρυβο. Για την διαδικασία αυτή οι αποστάσεις συγκρίνονται μια κάθε φορά με το προκαθορισμένο κατώφλι και η πρώτη για την οποία η τιμή της είναι μικρότερη από αυτό, μας δείχνει τον αριθμό των καθαρών στόχων ο οποίος μειωμένος κατά ένα μας δίνει την διάσταση του φασματικού υπόχωρου του σήματος. Η συγκεκριμένη διαδικασία παρουσιάζεται στο παρακάτω κομμάτι του κώδικα.

```
p=0;
for (i=0; i<bands-1; i++){
    if (dist[bands-2-i]<threshold){
        p=i+1;
        break;
        }
    }
*nendmembers=p;
```

Εικόνα 6.6. Απόσπασμα κώδικα υπολογισμού αριθμού καθαρών στόχων

6.2. Ανάλυση κυρίων συνιστωσών (PCA)

Η ανάλυση κυριών συνιστωσών αποτελεί μια στατιστική μέθοδο μετασχηματισμού ενός στατιστικού δείγματος, ώστε οι μεταβλητές του να είναι

ασυσχέτιστες μεταξύ τους και οι άξονες του διαγράμματος διασποράς να είναι στην διεύθυνση της μεγαλύτερης μεταβλητότητας. Στην ψηφιακή Τηλεπισκόπηση, το δείγμα είναι οι τιμές των εικονοστοιχείων της εικόνας και οι μεταβλητές είναι τα κανάλια.

Πρώτο βήμα της διαδικασίας αυτής είναι ο υπολογισμός των στατιστικών της εικόνας με χρήση των τύπων της στατιστικής. Για αυτό το σκοπό, αρχικά υπολογίζεται το άθροισμα των τιμών των εικονοστοιχείων για κάθε κανάλι και στη συνέχεια το άθροισμα αυτό διαιρείται με τον αριθμό των εικονοστοιχείων της εικόνας για να εξαχθεί η μέση τιμή, σύμφωνα με τον τύπο:

$$\overline{X_p} = \frac{\sum_{i=1}^{n} X_i}{n}$$

όπου, $\overline{X_p}$ είναι η μέση τιμή του καναλιού p,

$\sum_{i=1}^{n} X_i$ είναι το άθροισμα των n εικονοστοιχείων του καναλιού p

και n είναι το πλήθος των εικονοστοιχείων που περιέχει το κάθε κανάλι

Ο τύπος αυτός υλοποιείται από το παρακάτω κομμάτι του κώδικα:

```
for (i=0; i<rows; i++){
    fread (buffer, sizeof(float), cols*bands, fin);
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[(j*cols)+k];
                }
            }
        }
    }

for (i=0; i<bands; i++){
    mean[i]=sum[i]/(rows*cols-npixels);
    }
```

Εικόνα 6.7. Απόσπασμα κώδικα υπολογισμού αθροίσματος και μέσης τιμής καναλιών

Επόμενο βήμα είναι ο υπολογισμός του πίνακα μεταβλητότητας-συμμεταβλητότητας της εικόνας. Για τον σκοπό αυτό, πρέπει να υπολογιστούν οι μεταβλητότητες των καναλιών και οι συμμεταβλητότητες του καθενός με τα υπόλοιπα. Αυτό γίνεται με χρήση του τύπου:

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n}$$

όπου, $\mathrm{cov}(X,Y)$ είναι η συμμεταβλητότητα των καναλιών X και Y,

$\overline{X}$ και $\overline{Y}$ είναι η μέσες τιμές του καναλιού X και Y αντίστοιχα και

X_i και Y_i είναι οι τιμές των εικονοστοιχείων του καναλιού X και Y αντίστοιχα.

Η διαδικασία αυτή γίνεται σε δύο στάδια, καθώς πρώτα υπολογίζεται ο αριθμητής του παραπάνω πηλίκου, ο οποίος αργότερα διαιρείται με τον αριθμό των εικονοστοιχείων. Η πρώτη διαδικασία, λόγω του μεγάλου όγκου των πράξεων που περιέχει, έχει γραφτεί σε παράλληλο κώδικα, όπως φαίνεται παρακάτω:

```
# pragma omp parallel default(none) private(i, j, k, l)
           shared(buffer, fin, mean, sumv, rows, cols, bands, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fin);
    printf("Calculating Statistics at row %d of %d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[j*cols+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-mean[j])*
                                        (buffer[(k*cols)+l]-mean[k]);
                }
            }
        }
    }
}
```

Εικόνα 6.8. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης καναλιών(1)

Το δεύτερο κομμάτι του υπολογισμού υλοποιείται με την διαίρεση του αποτελέσματος του παραπάνω υπολογισμού με τον αριθμό των εικονοστοιχείων, όπως φαίνεται παρακάτω:

```
for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        cov[i][j]=sumv[i][j]/(rows*cols-npixels);
        }
    }
```

Εικόνα 6.9. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης καναλιών(2)

Στην συνέχεια πρέπει να υπολογιστούν οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα μεταβλητότητας-συμμεταβλητότητας. Ο αλγόριθμος αυτός δεν υλοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, αλλά πάρθηκε από το βιβλίο «Numerical Recipes, the art of scientific computing» (Press W. et al., 2007), γι' αυτό και δεν παρουσιάζεται σε αυτό το σημείο. Το ίδιο ισχύει και για τον αλγόριθμο ταξινόμησης των ιδιοδιανυσμάτων κατά φθίνουσα σειρά, με βάση την τιμή της αντίστοιχης ιδιοτιμής.

Τέλος, και εφόσον έχουν υπολογιστεί και ταξινομηθεί τα ιδιοδιανύσματα του πίνακα μεταβλητότητας, υπολογίζονται οι τιμές των εικονοστοιχείων της εικόνας των κυρίων συνιστωσών με βάση τον τύπο:

$$Y_{v\mu} = \sum_{i=0}^{p} [(X_{vi} - \overline{X_i}) * E_{\mu i}]$$

όπου, $Y_{v\mu}$ συμβολίζεται η τιμή του v–ιστού εικονοστοιχείου της μ-οστής συνιστώσας της μετασχηματισμένης εικόνας,

X_{vi}, η τιμή του v –οστού εικονοστοιχείου της αρχικής εικόνας στο i-οστό κανάλι της,

$\overline{X_i}$, η μέση τιμή του καναλιού i της αρχικής εικόνας,

$E_{\mu i}$ είναι η i-οστή τιμή του μ-οστού ιδιοδιανύσματος.

και p το πλήθος των καναλιών.

Ο συγκεκριμένος τύπος υλοποιείται στο παρακάτω απόσπασμα του κώδικα, ταυτόχρονα με την εγγραφή της νέας εικόνας στο δίσκο του υπολογιστή για περαιτέρω επεξεργασία. Η διαδικασία αυτή, όπως και οι περισσότερες στους αλγόριθμους της παρούσας εργασίας, γίνεται μια γραμμή κάθε φορά, για να αποφευχθεί η αποθήκευση ολόκληρης της εικόνας στην μνήμη, πράγμα που θα έθετε περιορισμούς στο μέγεθος της εικόνας που θα μπορούσε να επεξεργαστεί, με τους συγκεκριμένους αλγόριθμους. Λόγω του πλήθους των πράξεων, ο παρακάτω κώδικας είναι παράλληλος.

```
# pragma omp parallel default (none) private(i, j, k, l) shared(fin,
      buffer, rows, cols, bands, noc, pc, eigvec, fpca, mean, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fin);
    printf("Calculating Pr. Components at row %d of %d\r", i+1,
                                                              rows);
    }
    # pragma omp for schedule (dynamic)
    for (j=0; j<noc; j++){
        for (k=0; k<cols; k++){
            pc[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
                if (buffer[l*cols+k]!=nvalue){
                    pc[j*cols+k]+=1.*(buffer[(l*cols)+k]-mean[j])*
                                                    (eigvec[l][j]);
                    }
                else {
                    pc[j*cols+k]=nvalue;
                    }
                }
            }
        }
    # pragma omp single
    {
    fwrite(pc, sizeof(float), noc*cols, fpca);
    }
    }
}
```

Εικόνα 6.10. Απόσπασμα κώδικα υπολογισμού κυρίων συνιστωσών

6.3. Μετασχηματισμός Ελαχιστοποίησης του Θορύβου (MNF)

Ο συγκεκριμένος αλγόριθμος χρειάζεται ως δεδομένο την εικόνα του θορύβου, για να εξάγει τα στατιστικά της. Την εικόνα αυτή την λαμβάνει από κάποιον από τους παραπάνω αλγόριθμους. Επόμενος στόχος, για την υλοποίηση του αλγορίθμου MNF είναι να μετατρέψουμε τον θόρυβο αυτό σε λευκό θόρυβο, δηλαδή να έχει μέση τιμή ίση με το μηδέν και τυπική απόκλιση ίση με 1. Για να το καταφέρουμε αυτό πρέπει αρχικά να «στρέψουμε» την εικόνα από ένα σύστημα, το οποίο έχει κέντρο την τιμή μηδέν και άξονες τα κανάλια της εικόνας, σε ένα σύστημα με κέντρο τη μέση τιμή του θορύβου σε κάθε κανάλι και άξονες τις κύριες συνιστώσες της απεικόνισης του θορύβου. Αυτό θα συμβεί προβάλλοντας την εικόνα στα ιδιοδιανύσματα της εικόνας του θορύβου που πρέπει να έχουμε ήδη υπολογίσει. Επειδή ο θόρυβος της μετασχηματισμένης εικόνας θα πρέπει να έχει τυπική απόκλιση ίση με τη μονάδα, γι αυτό κατά την προβολή, η τιμή της μετασχηματισμένης εικόνας διαιρείται συγχρόνως με την τυπική απόκλιση του θορύβου.

Αρχικά ο χρήστης δηλώνει στο πρόγραμμα με ποια από τις δύο επιλέξιμες μεθόδους θέλει να υπολογιστεί ο θόρυβος της εικόνας, με τη μέθοδο NND ή τη μέθοδο Multiple Regression. Το πρόγραμμα αφού υπολογίσει την εικόνα του θορύβου (βλ. επόμενα κεφάλαια), υπολογίζει τα στατιστικά της εικόνας του θορύβου σε κάθε κανάλι, ξεκινώντας από το άθροισμα των τιμών των εικονοστοιχείων του κάθε καναλιού. όπως φαίνεται παρακάτω:

```
rewind(fnoise);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fnoise);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
        }
    }
```

Εικόνα 6.11. Απόσπασμα κώδικα υπολογισμού αθροίσματος θορύβου

Εφόσον είναι πλέον γνωστά τα αθροίσματα των τιμών των εικονοστοιχείων του κάθε καναλιού, μπορούν να εξαχθούν και οι μέσες τιμές τους, χρησιμοποιώντας τον απλό τύπο της μέσης τιμής

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Ο οποίος υλοποιείται με τον παρακάτω κώδικα:

```
for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }
```

Εικόνα 6.12. Απόσπασμα κώδικα υπολογισμού μέσης τιμής θορύβου

Στη συνέχεια χρειαζόμαστε τον πίνακα μεταβλητότητας- συμμεταβλητότητας των καναλιών. Ο πίνακας αυτός υπολογίζεται από τον γενικό τύπο της συμμεταβλητότητας:

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{n}$$

Η διαδικασία που ακολουθήθηκε για να υλοποιηθεί ο παραπάνω τύπος χωρίζεται σε δύο στάδια, πρώτα υπολογίζεται ο αριθμητής της σχέσης για κάθε κανάλι της εικόνας, σε όλους τους συνδυασμούς κατά το X και το Y και στην συνέχεια πραγματοποιείται η διαίρεση με τον αριθμό των καναλιών για κάθε μέγεθος. Λόγω του αριθμού των πράξεων στο πρώτο κομμάτι, ο κώδικας είναι παράλληλος.

```
# pragma omp parallel default(none) private(i, j, k, l)
       shared(rows, cols, bands, fnoise, buffer, sumv, Nmean, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fnoise);
    printf("Calculating Image Statistics at row %d of %d\r", i+1,
                                                          rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[(j*cols)+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-Nmean[j])*
                                    (buffer[(k*cols)+l]-Nmean[k]);
                }
            }
        }
    }
}
```

Εικόνα 6.13. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης θορύβου(1)

Στη συνέχεια ακολουθεί η διαίρεση όλων των τιμών του πίνακα με τον αριθμό των εικονοστοιχείων, δηλαδή με το γινόμενο των γραμμών της εικόνας με τις στήλες της.

```
for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        Ncov[i][j]=sumv[i][j]/(rows*cols-npixels);
        }
    }
```

Εικόνα 6.14. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης θορύβου(2)

Επόμενο βήμα είναι η εύρεση των ιδιοτιμών και των ιδιοδιανυσμάτων του πίνακα μεταβλητότητας-συμμεταβλητότητας του θορύβου και η ταξινόμηση τους κατά φθίνουσα σειρά με βάση την ιδιοτιμή. Αυτό επιτυγχάνεται με την χρήση του αλγορίθμου Jacobi.

Στη συνέχεια πρέπει να υπολογιστεί η μέση τιμή του κάθε καναλιού της αρχικής εικόνας ώστε, κατά τον πολλαπλασιασμό της αρχικής εικόνας με τα ιδιοδιανύσματα του πίνακα μεταβλητότητας του θορύβου, να αφαιρεθεί από την τιμή του κάθε εικονοστοιχείου για να έχει η τελική εικόνα μέση τιμή ίση με το μηδέν. Ο υπολογισμός της μέσης τιμής γίνεται στο παρακάτω απόσπασμα του κώδικα:

```
rewind(fin);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fin);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
        }
    }

for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }
```

Εικόνα 6.15. Απόσπασμα κώδικα υπολογισμού μέσης τιμής καναλιών

Η προβολή της αρχικής εικόνας γίνεται σύμφωνα με τον τύπο

$$X_{(b,r*c)} = \frac{\left(Y_{(b,r*c)} - M_{(b,r*c)}\right) * E_{(b,b)}}{s_{noise}{}_{(b,1)}}$$

όπου, ο πίνακας X είναι η μετασχηματισμένη εικόνα, με τόσες σειρές όσα τα κανάλια της και τόσες στήλες όσα τα εικονοστοιχεία της και ο πίνακας Y είναι η αρχική εικόνα με τον ίδιο αριθμό σειρών και στηλών. Ο πίνακας E συμβολίζει τον πίνακα των ιδιοδιανυσμάτων του πίνακα μεταβλητότητας του θορύβου, όπως υπολογίστηκε προηγούμενα. Ο πίνακας M παριστάνει την μέση τιμή των εικονοστοιχείων της εικόνας σε κάθε κανάλι της και αντίστοιχα ο πίνακας s_{noise} την τυπική απόκλιση των εικονοστοιχείων του θορύβου σε κάθε κανάλι. Αξίζει να αναφερθεί ότι ο πίνακας M έχει ουσιαστικά διαστάσεις (1xb), αλλά, αφού με τέτοιες διαστάσεις η αφαίρεση των πινάκων δεν ορίζεται, απλώς αντιγράφουμε αυτή τη στήλη τόσες φορές όσες ο αριθμός των εικονοστοιχείων της εικόνας, έτσι ώστε να έχει νόημα η πράξη.

Η όλη διαδικασία απεικονίζεται στο παρακάτω κομμάτι του παράλληλου κώδικα.

```
# pragma omp parallel default(none) private (i, j, k , l)
shared(buffer, buffer2, rows, cols, bands, Neigvec, fin, fpca1,
Nmean, Ncov, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fin);
    printf("Calculating White Noise Image at row %d of %d\r", i+1,
                                                          rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            buffer2[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
                if (buffer[l*cols+k]!=nvalue){
                    buffer2[j*cols+k]+=1.*(buffer[(l*cols)+k]-
                                    Nmean[l])*(Neigvec[l][j]);
                    }
                else {
                    buffer2[j*cols+k]=nvalue;
                    }
                }
            if (buffer2[j*cols+k]!=nvalue){

buffer2[j*cols+k]=buffer2[j*cols+k]/sqrt(fabs(Ncov[j][j]));
                }
            }
        }
    # pragma omp single
    {
    fwrite(buffer2,sizeof(float),bands*cols,fpca1);
    }
    }
}
```

Εικόνα 6.16. Απόσπασμα κώδικα υπολογισμού εικόνας λευκού θορύβου

Ένα σημαντικό στοιχείο του κομματιού αυτού είναι η διαίρεση του κάθε εικονοστοιχείου με την τυπική απόκλιση του θορύβου στο αντίστοιχο κανάλι. Έτσι ο θόρυβος παίρνει τιμή ίση με 1, από το οποίο και σε συνδυασμό με το ότι έχει μέση τιμή ίση με το 0, συμπεραίνουμε ότι ο γεωμετρικός τόπος των σημείων του στο διάγραμμα διασποράς είναι μια υπερσφαίρα στη n διάσταση με ακτίνα ίση με 1.

Στην συνέχεια πρέπει να στραφούν οι άξονες της μετασχηματισμένης εικόνας ώστε το μεγαλύτερο ποσοστό της πληροφορίας να περιέχεται στα πρώτα κανάλια. Γι' αυτό το λόγο υπολογίζουμε τα στατιστικά της μετασχηματισμένης εικόνας. Παρακάτω φαίνεται ο κώδικας που υπολογίζει το άθροισμα των εικονοστοιχείων του κάθε καναλιού και τη μέση τιμή τους για κάθε κανάλι.

```
for (i=0; i<bands; i++){
    sum[i]=0.0;
    for (j=0; j<bands; j++){
        sumv[i][j]=0.0;
        }
    }

rewind(fpca1);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fpca1);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
        }
    }

for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }
```

Εικόνα 6.17. Απόσπασμα κώδικα υπολογισμού μέσης τιμής καναλιών

Στην συνέχεια, με τρόπο όμοιο με παραπάνω, υπολογίζεται ο πίνακας μεταβλητότητας-συμμεταβλητότητας της εικόνας, όπως φαίνεται στο παρακάτω απόσπασμα του κώδικα.

```
# pragma omp parallel default(none) private(i, j, k, l)
shared(buffer, fpca1, Nmean, sumv, rows, cols, bands, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fpca1);
    printf("Calculating White Noise Image Statistics at row %d of
                                                %d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[(j*cols)+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-Nmean[j])*
                                (buffer[(k*cols)+l]-Nmean[k]);
                    }
                }
            }
        }
    }
}
```

Εικόνα 6.18. Απόσπασμα κώδικα υπολογισμού τυπικής απόκλισης καναλιών

Για να γίνει η στροφή των αξόνων, πρέπει να υπολογιστούν οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα μεταβλητότητας-συμμεταβλητότητας του θορύβου

που υπολογίστηκε παραπάνω. Η διαδικασία αυτή γίνεται με χρήση του αλγορίθμου jacobi, όπως και παραπάνω.

Το επόμενο, και τελευταίο, βήμα για τη υλοποίηση του αλγορίθμου MNF είναι η τελική στροφή των αξόνων. Αυτό γίνεται με χρήση του τύπου του πολλαπλασιασμού της εικόνας με τον πίνακα των ιδιοδιανυσμάτων και με ταυτόχρονη αφαίρεση της μέσης τιμής του κάθε καναλιού από την τιμή του κάθε εικονοστοιχείου, ο οποίος φαίνεται παρακάτω.

$$Z_{(b,r*c)} = \left(X_{(b,r*c)} - M_{(b,r*c)}\right) * E_{(b,b)}$$

όπου, ο πίνακας Z συμβολίζει την τελική εικόνα του αλγορίθμου MNF, με μέγεθος όσο η αρχική εικόνα,

ο πίνακας X είναι η μετασχηματισμένη εικόνα του λευκού θορύβου,

ο πίνακας M είναι ο πίνακας των μέσων τιμών των καναλιών της μετασχηματισμένης εικόνας του λευκού θορύβου και

ο πίνακας E αποτελεί τον πίνακα των ιδιοδιανυσμάτων του πίνακα μεταβλητότητας-συμμεταβλητότητας του θορύβου.

Ο πηγαίος κώδικας της παραπάνω διαδικασίας φαίνεται παρακάτω:

```
# pragma omp parallel default(none) private(i, j, k, l)
shared(buffer, buffer2, rows, cols, bands, fpca1, fsignal, Neigvec,
Nmean, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fpca1);
    printf("Calculating Pr. Components of Wh. Noise Image at row %d
                                                of %d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            buffer2[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
                if (buffer[l*cols+k]!=nvalue){
                    buffer2[j*cols+k]+=1.*(buffer[(l*cols)+k]-
                                        Nmean[l])*(Neigvec[l][j]);
                    }
                else {
                    buffer2[j*cols+k]=nvalue;
                    }
                }
            }
        }
    # pragma omp single
    {
    fwrite(buffer2,sizeof(float),bands*cols,fsignal);
    }
    }
}
```

Εικόνα 6.19. Απόσπασμα κώδικα υπολογισμού MNF καναλιών

6.4. Εκτίμηση του Θορύβου

6.4.1. Μέθοδος Nearest Neighbor Distance (NND)

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο για τη υλοποίηση του MNF απαιτείται η εκτίμηση του θορύβου. Αν και αυτό το μέρος του κώδικα περιλαμβάνεται στο προηγούμενο πρόγραμμα, παρατίθεται ξεχωριστά σε αυτό και στο επόμενο κεφάλαιο για λόγους διευκόλυνσης του αναγνώστη. Τα δύο αυτά κεφάλαια αναφέρονται στις δύο διαφορετικές εναλλακτικές μεθόδους τις οποίες υλοποιεί το πρόγραμμα. Κατά τη μέθοδο NND, υλοποιηείται το χωρικό φίλτρο το οποίο θα δώσει μια καλή εκτίμηση του, ο πίνακας συνέλιξης του οποίου μετά από δοκιμές κατέληξε να είναι ο παρακάτω.

	-1	
	2	**-1**

Το αρχείο εισόδου και το αρχείο εξόδου της συγκεκριμένης διαδικασίας, λόγω του μεγέθους τους είναι αποθηκευμένα στον σκληρό δίσκο, πράγμα το οποίο καθυστερεί σημαντικά την διαδικασία. Έτσι ο αλγόριθμος που υλοποιήθηκε προσπαθεί να ελαχιστοποιήσει το πλήθος αυτό των προσβάσεων στα αρχεία διαβάζοντας κάθε φορά μόνο μια σειρά της εικόνας και μεταθέτοντας τις υπόλοιπες ώστε να ισχύει ο αλγόριθμος του φίλτρου. Ο κώδικας της συγκεκριμένης διαδικασίας φαίνεται παρακάτω.

```
for (i=1; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fin);
    for (j=0; j<bands; j++){
        for (k=1; k<cols-1; k++){
            if (buffer2[j*cols+k]!= nvalue){
                dif[0]=buffer2[j*cols+k]-buffer2[j*cols+k+1];
                dif[1]=buffer2[j*cols+k]-buffer[j*cols+k];
                noise1[j*cols+k]=(dif[0]+dif[1])/2;
                }
            else {
                noise1[j*cols+k]=nvalue;
                }
            }
        }
    fwrite(noise1, sizeof(float), cols*bands, fnoise);
    memcpy(buffer, buffer2, cols*bands*sizeof(float));
    }
```

Εικόνα 6.20. Απόσπασμα κώδικα εφαρμογής του φίλτρου

Έτσι, προκύπτει η εικόνα του θορύβου την οποία χρειάζεται να επεξεργαστεί το πρόγραμμα MNF για να μετατρέψει την αρχική εικόνα, σε εικόνα λευκού θορύβου.

6.4.2. Multiple Regression Theory-Based Method

Η μέθοδος της πολλαπλής παλινδρόμησης αξιοποιεί το γεγονός ότι οι τιμές των εικονοστοιχείων σε κάθε κανάλι δεν είναι ασυσχέτιστες μεταξύ τους και ότι μπορεί να προβλεφθεί η τιμή ενός εικονοστοιχείου σε κάποιο κανάλι από τις τιμές που έχει στα υπόλοιπα κανάλια. Έτσι η μέθοδος αυτή παράγει μια νέα εικόνα η οποία είναι η εκτίμηση των τιμών που θα έπρεπε να έχει το κάθε εικονοστοιχείο σε κάθε κανάλι αν έλειπε το κανάλι αυτό. Η διαφορά της εκτιμώμενης τιμής από την πραγματική αποτελεί τον θόρυβο.

Η εκτιμώμενη εικόνα προκύπτει από τον τύπο:

$$\hat{z}_i = Z_{\ i} * \hat{b}_i$$

όπου, $\hat{z}_i$ είναι οι εκτιμώμενες από τη μέθοδο της παλινδρόμησης τιμές του καναλιού i,

$Z_{\ i}$ είναι οι τιμές της αρχικής εικόνας σε όλα τα κανάλια εκτός από το i,

και $\hat{b}_i$ είναι οι συντελεστές παλινδρόμησης για το κανάλι i.

Οι συντελεστές παλινδρόμησης υπολογίζονται από τον τύπο:

$$\hat{b}_i = (Z^T_{\ i} * Z_{\ i})^{-1} * Z^T_{\ i} * z_i$$

όπου, z_i είναι οι τιμές του καναλιού i στην αρχική εικόνα και

$Z^T_{\ i}$ είναι ο ανάστροφος του πίνακα $Z_{\ i}$.

Όπως φαίνεται και από τους τύπους, ο υπολογισμός της νέας εικόνας γίνεται ανά ένα κανάλι κάθε φορά. Έτσι, οι παραπάνω τύποι υπολογίζονται για κάθε κανάλι της εικόνας, πράγμα το οποίο, ανάλογα και με το μέγεθος της εικόνας, αυξάνει κατά πολύ τον χρόνο εκτέλεσης του προγράμματος.

Όσον αφορά τη διαδικασία της δημιουργίας των πινάκων, για να μειωθούν οι απαιτήσεις του προγράμματος, ο πίνακας $Z^T_{\ i}$ δεν δημιουργήθηκε. Για να γίνουν σωστά οι πράξεις, διαβάστηκε αντίστροφα ο πίνακας $Z_{\ i}$. Για αυτό το λόγο δημιουργήθηκε μόνο ο τελευταίος, στο παρακάτω απόσπασμα του κώδικα:

```
for (j=0; j<i; j++){
    fseek(fin, j*cols*sizeof(float), SEEK_SET);
    for (k=0; k<rows; k++){
        fread(buffer, sizeof(float), cols, fin);
        for (l=0; l<cols; l++){
            Z[k*cols+l][j]=buffer[l];
            }
        fseek(fin, (bands-1)*cols*sizeof(float), SEEK_CUR);
        }
    }

for (j=i; j<bands-1; j++){
    fseek(fin, (j+1)*cols*sizeof(float), SEEK_SET);
    for (k=0; k<rows; k++){
        fread(buffer, sizeof(float), cols, fin);
        for (l=0; l<cols; l++){
            Z[k*cols+l][j]=buffer[l];
            }
        fseek(fin, (bands-1)*cols*sizeof(float), SEEK_CUR);
        }
    }
```

Εικόνα 6.21. Απόσπασμα κώδικα δημιουργίας του πίνακα Z

Επόμενο βήμα είναι ο πολλαπλασιασμός του πίνακα Z_{gi} με τον ανάστροφό του. Το κομμάτι αυτό, όπως αναφέρθηκε και παραπάνω, έγινε διαβάζοντας αντίστροφα τον πίνακα, για να μην δημιουργηθεί ο ανάστροφος, ο οποίος, λόγω του μεγέθους του, θα αύξανε σημαντικά τις απαιτήσεις του προγράμματος σε μνήμη. Έτσι ο πολλαπλασιασμός πραγματοποιείται όπως φαίνεται παρακάτω:

```
for (j=0; j<bands-1; j++){
    for (k=0; k<bands-1; k++){
        tmp=0.0;
        for (l=0; l<rows*cols; l++){
            tmp+=Z[l][k]*Z[l][j];
            }
        zTz[j][k]=tmp;
        }
    }
```

Εικόνα 6.22. Απόσπασμα κώδικα υπολογισμού πίνακα Z^TZ

Στη συνέχεια ακολουθεί η αντιστροφή του αποτελέσματος του πολλαπλασιασμού. Το συγκεκριμένο κομμάτι του κώδικα δεν δημιουργήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, αλλά πάρθηκε έτοιμο από το βιβλίο «Numerical Recipes, the art of scientific computing» (Press W. et al., 2007). Έτσι δεν χρειάζεται να παρουσιαστεί στο σημείο αυτό. Ο συγκεκριμένος αλγόριθμος φαίνεται στο παράρτημα.

Επόμενο βήμα είναι ο πολλαπλασιασμός του πίνακα που μόλις αντιστράφηκε, με τον πίνακα Z_i, όπως υπαγορεύουν οι τύποι της παλινδρόμησης. Ο

συγκεκριμένος πολλαπλασιασμός παρουσιάζεται στο παρακάτω κομμάτι του κώδικα.

```
for (j=0; j<bands-1; j++){
    for (k=0; k<rows*cols; k++){
        tmp=0.0;
        for (l=0; l<bands-1; l++){
            tmp+=zTz_inv[j][l]*Z[k][l];
            }
        R[j][k]=tmp;
        }
    }
```

Εικόνα 6.23. Απόσπασμα κώδικα υπολογισμού πίνακα Z^TZZ^T

Για να υπολογιστούν οι συντελεστές παλινδρόμησης για κάθε εικονοστοιχείο της εικόνας, πρέπει να πολλαπλασιαστεί το αποτέλεσμα του παραπάνω πολλαπλασιασμού με τις τιμές των εικονοστοιχείων του αντίστοιχου καναλιού στην αρχική εικόνα. Το αντίστοιχο κομμάτι του κώδικα φαίνεται παρακάτω:

```
for (j=0; j<bands-1; j++){
    tmp=0.0;
    for (k=0; k<rows*cols; k++){
        tmp+=R[j][k]*z[k];
        }
    b[j]=tmp;
    }
```

Εικόνα 6.24. Απόσπασμα κώδικα υπολογισμού των συντελεστών παλινδρόμησης

Οι τιμές των εικονοστοιχείων του προς επεξεργασία καναλιού της εικόνας του θορύβου υπολογίζονται από την αφαίρεση του αποτελέσματος του πολλαπλασιασμού του πίνακα $Z_{\vartheta i}$ με τους συντελεστές παλινδρόμησης, από την αρχική εικόνα. Οι πράξεις αυτές παρουσιάζονται στο παρακάτω κομμάτι του κώδικα:

```
for (j=0; j<rows*cols; j++){
    tmp=0.0;
    for (k=0; k<bands-1; k++){
        tmp+=Z[j][k]*b[k];
        }
    c[j]=tmp;
    }

for (j=0; j<rows*cols; j++){
    if (z[j]!=nvalue){
        noise2[j]=(float)(z[j]-c[j]);
        }
    else {
        noise2[j]=nvalue;
        }
    }
```

Εικόνα 6.25. Απόσπασμα κώδικα υπολογισμού θορύβου

Το τελικό κομμάτι του αλγόριθμου είναι το γράψιμο της εικόνας του θορύβου στο δίσκο για να συνεχιστεί η επεξεργασία της. Η διαδικασία περιπλέκεται λόγω της τυποποίησης του αρχείου, καθώς ενώ το αρχείο πρέπει να γραφτεί σύμφωνα με το πρότυπο BIL, στο οποίο κάθε σειρά ακολουθείται από την ίδια σειρά του επόμενου καναλιού, η μέθοδος της παλινδρόμησης δίνει αποτελέσματα για κάθε κανάλι κάθε φορά. Έτσι το γράψιμο της εικόνας χαρακτηρίζεται από πολλές μετατοπίσεις του δείκτη θέσης του αρχείου, όπως φαίνεται παρακάτω.

```
if (fseek(fnoise2, i*cols*sizeof(float), SEEK_SET) != 0){
   puts ("Bad File Rewind");
   }

for (j=0; j<rows; j++){
    fwrite(noise2+j*cols, sizeof(float), cols, fnoise2);
    if (fseek ( fnoise2, cols*(bands-1)*sizeof(float), SEEK_CUR)
                                                          != 0){
       puts( "Bad File Control");
       }
    fflush(NULL);
    }
```

Εικόνα 6.26. Απόσπασμα κώδικα εγγραφής εικόνας θορύβου

6.5. Εξαγωγή των Καθαρών Στόχων

Ο αλγόριθμος Simple Endmember Extraction (SEE) εξάγει τις φασματικές υπογραφές των καθαρών στόχων μια εικόνας. Αυτό επιτυγχάνεται με την αξιοποίηση της ιδιότητας των καθαρών στόχων να βρίσκονται στις άκρες του διαγράμματος διασποράς μιας εικόνας με αριθμό καναλιών ίσο με τη διάσταση του υποχώρου του σήματος. Αυτό συμβαίνει διότι το σύνολο των εικονοστοιχείων της εικόνας αποτελεί γραμμικό συνδυασμό των καθαρών στόχων της.

Το πρώτο βήμα του αλγόριθμου Simple Endmember Extraction είναι η μείωση των διαστάσεων της εικόνας σύμφωνα με τη διάσταση του υποχώρου του σήματος, με χρήση του μετασχηματισμού κυρίων συνιστωσών εάν πρόκειται για εικόνα με λευκό θόρυβο ή Μετασχηματισμό Ελαχιστοποίησης του Θορύβου για όλες τις υπόλοιπες εικόνες. Ο χρήστης επομένως δηλώνει το μετασχηματισμό που θέλει αρχικά να εκτελεστεί. Αξιοποιώντας το πρόγραμμα τη διάσταση του σήματος που έχει προκύψει από τον ODM, ή εναλλακτικά έχει δοθεί από τον χρήστη, προκύπτει μια μετασχηματισμένη εικόνα με

διαστάσεις όσοι οι καθαροί στόχοι της αρχικής εικόνας μείον ένα, χωρίς όμως να έχει χαθεί σημαντικό κομμάτι της πληροφορίας.

Στην συνεχεία ακολουθεί η εύρεση της μέγιστης και της ελάχιστης τιμής του κάθε καναλιού. Όμως, το πιο σημαντικό στοιχείο της συγκεκριμένης διαδικασίας είναι η αποθήκευση των εικονοσυντεταγμένων, στις οποίες παρατηρούνται οι μέγιστες και οι ελάχιστες τιμές, καθώς θα πρέπει αργότερα να ληφθούν οι φασματικές υπογραφές των εικονοστοιχείων που έχουν τις συγκεκριμένες συντεταγμένες. Το παρακάτω απόσπασμα του κώδικα δείχνει την συγκεκριμένη διαδικασία:

```
for (i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*(nendmembers-1)*4, fpca4);
    for (j=0; j<(nendmembers-1); j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols*4+k]!=nvalue){
                if (buffer[j*cols*4+k]<hmin[j]){
                    hmin[j]=buffer[j*cols*4+k];
                    coordsr[j]=i;
                    coordsc[j]=k;
                    }
                if (buffer[j*cols*4+k]>hmax[j]){
                    hmax[j]=buffer[j*cols*4+k];
                    coordsr[(nendmembers-1)+j]=i;
                    coordsc[(nendmembers-1)+j]=k;
                    }
                }
            }
        }
    }
```

Εικόνα 6.27. Απόσπασμα κώδικα εντοπισμού υποψήφιων καθαρών στόχων

Όπως αναφέρθηκε και παραπάνω, το επόμενο βήμα είναι η εξαγωγή των φασματικών υπογραφών των εικονοστοιχείων στα οποία παρατηρήθηκαν οι μέγιστες και οι ελάχιστες τιμές. Τα εικονοστοιχεία αυτά αποτελούν τους υποψήφιους καθαρούς στόχους της εικόνας. Η διαδικασία αυτή παρουσιάζεται παρακάτω.

```
for (i=0; i<2*(nendmembers-1); i++){
    fseek(fin, coordsr[i]*cols*bands*sizeof(float), SEEK_SET);
    fread(buffer2, sizeof(float), cols*bands, fin);
    for (j=0; j<bands; j++){
        ssignature[i][j]=buffer2[j*cols+coordsc[i]];
        }
    }
```

Εικόνα 6.28. Απόσπασμα κώδικα απόκτησης φασματικών υπογραφών των υποψήφιων καθαρών στόχων

Χρησιμοποιώντας τη μέθοδο που αναφέρθηκε παραπάνω, καταλήγουμε σε 2(p-1) υποψήφιους καθαρούς στόχους, με p να συμβολίζεται ο αριθμός των καθαρών στόχων που προέκυψαν από την χρήση του αλγόριθμου ODM. Αυτό συμβαίνει, γιατί κάποιοι καθαροί στόχοι καταλαμβάνουν τις μέγιστες ή τις ελάχιστες τιμές σε περισσότερα από ένα μετασχηματισμένα κανάλια. Επομένως, ο διαχωρισμός τους γίνεται με βάση τις φασματικές γωνίες που σχηματίζουν με τους υπόλοιπους υποψήφιους καθαρούς στόχους. Έτσι, από τις γνωστές φασματικές υπογραφές, πρέπει να υπολογιστούν οι φασματικές γωνίες μεταξύ των υποψήφιων καθαρών στόχων, με χρήση του τύπου που φαίνεται παρακάτω.

$$SAD_{i,j} = \arccos\left(\frac{\sum_{k=0}^{n}(ce_{i,k}ce_{j,k})}{\sqrt{\sum_{k=0}^{n}(ce_{i.k})^2\sum_{k=0}^{n}(ce_{j,k})^2}}\right)$$

όπου, $SAD_{i,j}$ είναι η φασματική γωνία μεταξύ των υποψήφιων καθαρών στόχων i και j,

και $ce_{i,k}$ και $ce_{j,k}$ συμβολίζεται η τιμή της φασματικής υπογραφής του υποψήφιου καθαρού στόχου i και j αντίστοιχα, στο κανάλι k.

Για να υπολογιστούν τα μεγέθη αυτά, υπολογίζονται πρώτα οι όροι του κλάσματος για κάθε ζεύγος υποψήφιων καθαρών στόχων, και στη συνέχεια οι φασματικές γωνίες, όπως φαίνεται και παρακάτω. Το κομμάτι αυτό του κώδικα λειτουργεί και ως ξεχωριστό πρόγραμμα γιατί οι φασματικές γωνίες χρησιμοποιούνται ευρέως στην Υπερφασματική Τηλεπισκόπηση για την αξιολόγηση των μεθόδων. Π.χ., σε αυτή τη διπλωματική εργασία χρησιμοποιήθηκε το πρόγραμμα υπολογισμού των φασματικών γωνιών για την αξιολόγηση του προγράμματος εξαγωγής των καθαρών στόχων.

```
for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        for (k=0; k<bands; k++){
            if (i!=j){
                pro[i][j]+=ssignature[i][k]*ssignature[j][k];
                sqr[i][j]+=ssignature[j][k]*ssignature[j][k];
                }
            }
        }
    for (l=0; l<bands; l++){
        sum[i]+=ssignature[i][l]*ssignature[i][l];
        }
    }

for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        if (i!=j){

sad[i][j]=acos((pro[i][j])/((sqrt(sum[i]))*(sqrt(sqr[i][j]))));
            }
        }
    }
```

Εικόνα 6.29. Απόσπασμα κώδικα υπολογισμού των φασματικών γωνιών

Στη συνέχεια, για να είναι δυνατή η σύγκριση των φασματικών γωνιών, πρέπει να υπολογιστεί το άθροισμα των γωνιών κάθε υποψήφιου καθαρού στόχου με τους υπόλοιπους και τα αθροίσματα αυτά να κανονικοποιηθούν. Για την κανονικοποίηση αυτή πρέπει να υπολογιστούν, η μέγιστη και η ελάχιστη τιμή των αθροισμάτων αυτών, ώστε να είναι γνωστό το εύρος στο οποίο κυμαίνονται. Η διαδικασία αυτή παρουσιάζεται στο παρακάτω απόσπασμα του πηγαίου κώδικα.

```
ssadmax=0.;
ssadmin=5000.;
for (i=0; i<2*(nendmembers-1); i++){
    if (ssadmax<ssad[i]){
        ssadmax=ssad[i];
        }
    if (ssadmin>ssad[i]){
        ssadmin=ssad[i];
        }
    }

for (i=0; i<2*(nendmembers-1); i++){
    ssad[i]= (ssad[i]-ssadmin)/(ssadmax-ssadmin);
    }
```

Εικόνα 6.30. Απόσπασμα κώδικα κανονικοποίησης των αθροισμάτων των φασματικών γωνιών

Επόμενο βήμα είναι ο υπολογισμός των διαφορών, των αθροισμάτων των φασματικών γωνιών κάθε υποψήφιου καθαρού στόχου με τους υπόλοιπους. Με βάση αυτές τις διαφορές, ο αλγόριθμος θα διακρίνει τις όμοιες φασματικές

υπογραφές και θα επιλέξει αυτές που δεν μοιάζουν, ώστε να κρατήσει μόνο αυτές που ανήκουν στους καθαρούς στόχους.

Για να γίνει η επιλογή των καθαρών στόχων από το πρόγραμμα, ορίζεται ένα κατώφλι, πάνω από το οποίο πρέπει να είναι όλες οι προαναφερθείσες διαφορές κάθε υποψήφιου καθαρού στόχου με τους υπόλοιπους. Έτσι, αρχικά ορίζεται το κατώφλι αυτό, χαμηλά και γίνεται δοκιμή για το πόσοι καθαροί στόχοι θα προκύψουν από αυτό. Όσο οι καθαροί στόχοι είναι περισσότεροι από τον αριθμό που έχει προκύψει από τον ODM, το κατώφλι αυτό θα μεγαλώνει ελάχιστα, μέχρι να επιτευχθεί ο επιθυμητός αριθμός καθαρών στόχων. Το κομμάτι του κώδικα που είναι υπεύθυνο για την παραπάνω διαδικασία παρουσιάζεται παρακάτω.

```
threshold=0.0;
c=1000;
while (threshold<0.1 && c>nendmembers){
    c=0;
    threshold= threshold+0.00001;
    for (i=0; i<2*(nendmembers-1); i++){
        endm[i]=0;
        for (j=i+1; j<2*(nendmembers-1); j++){
            if (dif[i][j]<threshold){
                tempint[i]++;
                }
            }
        if (tempint[i]==0){
            c++;
            endm[i]=1;
            }
        }
    }
```

Εικόνα 6.31. Απόσπασμα κώδικα επιλογής των καθαρών στόχων

Τέλος, αυτό που απομένει είναι να δοθούν τα αποτελέσματα στον χρήστη, πράγμα το οποίο εκτελείται από την παρακάτω διαδικασία. Εδώ το πρόγραμμα τυπώνει στο αρχείο εξόδου, τους υποψήφιους καθαρούς στόχους, τους οποίους έχει σημάνει, ως πραγματικούς καθαρούς στόχους, η προηγούμενη διαδικασία, ενώ ταυτόχρονα προετοιμάζει τον πίνακα A του αλγόριθμου εκτίμησης της αφθονίας, που ακολουθεί.

```
k=0;
fprintf(report, "\nEndmembers :\n");
for (i=0; i<2*(nendmembers-1); i++){
    if (endm[i]!=0){
        fprintf(report, "\nEndmember %d: at row %d, col %d:\t", i+1,
                                        coordsr[i]+1, coordsc[i]+1);
        for (j=0; j<bands; j++){
            fprintf(report, "\t%5.5lf", ssignature[i][j]);
            a[j][k]=ssignature[i][j];
            }
        k++;
        }
    }
```

Εικόνα 6.32. Απόσπασμα κώδικα εγγραφής των φασματικών υπογραφών των καθαρών στόχων

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, οι δύο αλγόριθμοι εξαγωγής φασματικών υπογραφών που υλοποιήθηκαν, ο SEE και ο SEE-E διαφέρουν στο γεγονός ότι ο SEE-E προϋποθέτει την πρόσθεση της φασματικής υπογραφής του πρώτου καθαρού στόχου στην αρχική εικόνα. Η υπογραφή αυτή προστίθεται σε τόσα εικονοστοιχεία όσα είναι τα εικονοστοιχεία της εικόνας εις τριπλούν. Η εικόνα που προκύπτει είναι και η εικόνα που θα μετασχηματιστεί στις κύριες συνιστώσες της, για να αναζητηθούν οι καθαροί στόχοι στις κορυφές του υπόχωρου.

Επομένως, πρώτη διαδικασία για την υλοποίηση του αλγόριθμου, είναι ο εντοπισμός της μέγιστης προβαλλόμενης τιμής στην κατάλληλα μετασχηματισμένη εικόνα (PCA ή MNF). Την αναζήτηση αυτή πραγματοποιεί το παρακάτω απόσπασμα του κώδικα:

```
smax=-500000.0;
rewind(fsignal);
for (i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols, fsignal);
    for (j=0; j<cols; j++){
        if (buffer[j]!=nvalue){
            if (smax<buffer[j]){
                smax=buffer[j];
                maxrow=i;
                maxcol=j;
                }
            }
        }
    fseek(fsignal, cols*(bands-1)*sizeof(float), SEEK_CUR);
    }
```

Εικόνα 6.33. Απόσπασμα κώδικα εντοπισμού της μέγιστης προβαλλόμενης τιμής

Έπειτα, πρέπει να ανακτηθεί η φασματική υπογραφή του εικονοστοιχείου αυτό στην αρχική εικόνα. Η διαδικασία αυτή γίνεται στο παρακάτω κομμάτι του κώδικα:

```
fseek(fin, maxrow*cols*bands*sizeof(float), SEEK_SET);
fread(buffer2, sizeof(float), cols*bands, fin);
for(i=0; i<bands; i++){
    mpv[i]=buffer2[i*cols+maxcol];
    }
```

Εικόνα 6.34. Απόσπασμα κώδικα εξαγωγής της φασματικής υπογραφής της μέγιστης προβαλλόμενης τιμής

Τέλος, γίνεται η προσθήκη της φασματικής υπογραφής αυτής στα δεξιά της εικόνας. Έτσι προκύπτει η εικόνα, την οποία θα μετασχηματίσει στις κύριες συνιστώσες της ο SEE-E. Το παρακάτω απόσπασμα κώδικα παρουσιάζει της διαδικασία της πρόσθεσης της προαναφερθείσας φασματικής υπογραφής στην αρχική εικόνα.

```
for (i=0; i<bands; i++){
    for (j=0; j<cols*3; j++){
        buffer3[i*cols*3+j]=mpv[i];
        }
    }

rewind(fin);
for (i=0 ; i<rows; i++){
    for (j=0; j<bands; j++){
        fread(buffer2, sizeof(float), cols, fin);
        fwrite(buffer2, sizeof(float), cols, fplus);
        fwrite(buffer3+j*cols*3, sizeof(float), cols*3, fplus);
        }
    }
```

Εικόνα 6.35. Απόσπασμα κώδικα προσθήκης της φασματικής υπογραφής της μέγιστης προβαλλόμενης τιμής στην αρχική εικόνα

6.6. Εκτίμηση της Αφθονίας

Η μέθοδος εκτίμησης της αφθονίας βασίζεται στο γεγονός ότι, οι τιμές των εικονοστοιχείων της εικόνας αποτελούν γραμμικό συνδυασμό των φασματικών υπογραφών των καθαρών στόχων. Με βάση την λογική αυτή, δημιουργείται ένα γραμμικό σύστημα, του τύπου Ax=B, με τον πίνακα A να είναι ο πίνακας των φασματικών υπογραφών των καθαρών στόχων, τον πίνακα x να είναι ο πίνακας των συντελεστών αφθονίας τους, στο κάθε εικονοστοιχείο, και τον πίνακα B να είναι οι τιμές του εικονοστοιχείου αυτού στο κάθε κανάλι. Η λύση του προβλήματος συνοψίζεται στην υλοποίηση της παρακάτω πράξης.

$$\hat{x} = (A^T A)^{-1} A^T B$$

όπου, ο πίνακας x είναι ο πίνακας με τις τιμές αφθονίας των καθαρών στόχων για κάθε εικονοστοιχείο,

ο πίνακας A συμβολίζει τον πίνακα των εξισώσεων παρατήρησης, δηλαδή τον πίνακα με τις φασματικές υπογραφές των καθαρών στόχων,

ο πίνακας A^T είναι ο ανάστροφος του πίνακα A και

ο πίνακας B αποτελείται από τις τιμές του κάθε εικονοστοιχείου σε κάθε κανάλι της αρχικής εικόνας.

Για τη βελτιστοποίηση της διαδικασίας, η παραπάνω πράξη μπορεί να χωριστεί σε δύο όρους. Ο όρος $(A^T A)^{-1} A^T$ είναι σταθερός για κάθε εικονοστοιχείο, καθώς χαρακτηρίζει μόνο το σύνολο των φασματικών υπογραφών των καθαρών στόχων. Έτσι, μπορεί να θεωρηθεί ότι $N = (A^T A)^{-1} A^T$ και ότι, για κάθε εικονοστοιχείο θα πραγματοποιείται μόνο η πράξη $\hat{x} = N \times B$.

Για τον υπολογισμό του πίνακα $A^T A$, υλοποιήθηκε ο πολλαπλασιασμός των γραμμών του πίνακα A με τις στήλες του. Έτσι, δεν χρειάστηκε να δεσμευτεί και άλλη μνήμη για την αποθήκευση του ανάστροφου πίνακα A^T. Η συγκεκριμένη πράξη φαίνεται στο παρακάτω κομμάτι του κώδικα.

```
for (i=0; i<nendmembers ; i++){
    for (j=0; j<nendmembers ; j++){
        for (k=0; k<bands; k++){
            ata[i][j]+=a[k][i]*a[k][j];
            }
        }
    }
```

Εικόνα 6.36. Απόσπασμα κώδικα υπολογισμού του πίνακα A^TA

Ο κώδικας για την αντιστροφή του πίνακα $A^T A$ δεν δημιουργήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, αλλά πάρθηκε έτοιμος από το βιβλίο «Numerical Recipes, the art of scientific computing». Έτσι δεν χρειάζεται να παρουσιαστεί στο σημείο αυτό.

Για τον υπολογισμό του πίνακα N, πρέπει να πολλαπλασιαστεί ο $(A^T A)^{-1}$ με τον A^T. Με την ίδια λογική με προηγουμένως, δεν γίνεται αυτή η πράξη αλλά ο πολλαπλασιασμός του $(A^T A)^{-1}$ με τις γραμμές του A, όπως φαίνεται και στο παρακάτω κομμάτι του κώδικα.

```
for (i=0; i<nendmembers; i++){
    for (j=0; j<bands; j++){
        tmpd=0.0;
        for (k=0; k<nendmembers; k++){
            tmpd+=ata_inv[i][k]*a[j][k];
            }
        N[i][j]=tmpd;
        }
    }
```

Εικόνα 6.37. Απόσπασμα κώδικα υπολογισμού του πίνακα N

Επομένως, για τον υπολογισμό των συντελεστών αφθονίας του κάθε καθαρού στόχου σε κάθε εικονοστοιχείο, αρκεί να γίνει ο πολλαπλασιασμός $\hat{x} = \mathrm{N} \times \mathrm{B}$ για κάθε εικονοστοιχείο, καθώς και το να γραφτεί με την κατάλληλη τυποποίηση στην εικόνα του τελικού αποτελέσματος. Η διαδικασία αυτή πραγματοποιείται από το παρακάτω κομμάτι του κώδικα.

```
for (i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fin);
    printf("Unmixing at row %d of %d\r", i+1, rows);
    for (j=0; j<cols; j++){
        for (k=0; k<bands; k++){
            b[k]=buffer[k*cols+j];
            }
        for (l=0; l<nendmembers; l++){
            c=0.0;
            if (b[0]==nvalue){
                buffer2[l*cols+j]=nvalue;
                }
            else {
                for (m=0; m<bands; m++){
                    c+=N[l][m]*b[m];
                    }
                buffer2[l*cols+j]=c;
                }
            }
        }
    fwrite(buffer2, sizeof(float), nendmembers*cols, fend);
    for (j=0; j<cols*nendmembers; j=j+30){
        fprintf(report, " %f", buffer2[j]);
        }
    }
```

Εικόνα 6.38. Απόσπασμα κώδικα εγγραφής εικόνας συντελεστών αφθονίας

7. ΑΞΙΟΛΟΓΗΣΗ ΤΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

Στο κεφάλαιο αυτό γίνεται η αξιολόγηση των αποτελεσμάτων των προγραμμάτων τα οποία υλοποιήθηκαν στο πλαίσιο αυτής της διπλωματικής εργασίας. Η αξιολόγηση θα γίνει ως προς τα ποιοτικά και χρονικά αποτελέσματα των προγραμμάτων που υλοποιούν τους βασικούς αλγορίθμους του φασματικού διαχωρισμού, τον ODM και τον E-SEE, τα οποία εξάλλου περιλαμβάνουν κατά την εκτέλεσή τους και την εκτέλεση των υπολοίπων προγραμμάτων για τα οποία δεν γίνεται ξεχωριστή αξιολόγηση. Τα αποτελέσματα συγκρίνονται με αντίστοιχους αλγόριθμους εμπορικών προγραμμάτων, καθώς και με τα αντίστοιχα προγράμματα, που έχουν γραφτεί στο περιβάλλον της γλώσσας προγραμματισμού IDL στο πλαίσιο της διδακτορικής διατριβής της Δρ. Ανδρέου Χαρούλας (Ανδρέου Χ. 2014).

Πρόγραμμα Εκτίμησης του Φασματικού Υπόχωρου του Σήματος

Το πρόγραμμα ODM δοκιμάστηκε σε συνθετικά και πραγματικά δεδομένα. Στην περίπτωση των συνθετικών εικόνων με λευκό θόρυβο, ο μετασχηματισμός που χρησιμοποιείται είναι αυτός των κυρίων συνιστωσών, ενώ στην περίπτωση των πραγματικών εικόνων, είναι ο Μετασχηματισμός Ελαχιστοποίησης του Θορύβου (MNF). Για τον αλγόριθμο ODM, συγκρίθηκαν τα αποτελέσματα του προγράμματος σε C με αυτά που προέκυψαν από τον προγραμματισμό του ίδιου αλγορίθμου σε IDL, για μία συνθετική και μια πραγματική εικόνα.

Η συνθετική εικόνα έχει προκύψει από 7 καθαρούς στόχους, έχει signal-to-noise ratio 30 και οι διαστάσεις της είναι 100 x 100 εικονοστοιχεία σε 423 κανάλια (Ανδρέου Χ. 2014). Το πρόγραμμα ODM, αφού μετασχημάτισε την εικόνα στις κύριες συνιστώσες της, εκτίμησε σωστά τους 7 καθαρούς στόχους και το αποτέλεσμα συνέπεσε με αυτό του προγράμματος σε περιβάλλον IDL.

Η πραγματική εικόνα απεικονίζει περιοχή από την Indian pines και έχει ληφθεί με το δέκτη AVIRIS. Έχει 145 σειρές, 145 στήλες και 186 κανάλια. Το αποτέλεσμα που πρέπει να δώσει το πρόγραμμα ως προς τη διάσταση του υπόχωρου του σήματος πρέπει να είναι πάνω από 16.

Μέθοδος Εκτίμησης Θορύβου	Αριθμός καθαρών στόχων (ODM IDL)	Αριθμός καθαρών στόχων (ODM C)
NND	24	22
Multiple Regression (MRTBM)	17	17

Πίνακας 7.1. Σύγκριση αποτελεσμάτων αλγόριθμου ODM σε IDL και C

Από τον παραπάνω πίνακα παρατηρείται ότι τα αποτελέσματα είναι σχεδόν τα ίδια και για τα δύο προγράμματα που υλοποιούν τον αλγόριθμο, με την μόνη διαφορά, να προκύπτει από το πρόγραμμα σε C, όταν ο θόρυβος εκτιμάται με τη μέθοδο NND. Αυτό συμβαίνει, λόγω του διαφορετικού τρόπου εφαρμογής του φίλτρου. Το αντίστοιχο πρόγραμμα της IDL εφαρμόζει διαφορετικά το φίλτρο, χωρίς όμως να παρουσιάζει τον τρόπο.

Εικόνα 7.1. Το πρώτο κανάλι της εικόνας Indian pines του δέκτη AVIRIS.

Σε αυτό το σημείο πρέπει να αναφερθεί ότι, το πρόγραμμα της πολλαπλής παλινδρόμησης (MRTBM) που δημιουργήθηκε στο πλαίσιο της παρούσας διπλωματικής, έχει περιορισμό στο μέγεθος της εικόνας την οποία μπορεί να επεξεργαστεί. Αυτό συμβαίνει, διότι κατά τη διάρκεια των υπολογισμών , η εικόνα παραμένει αποθηκευμένη στην μνήμη του υπολογιστή, η οποία θα πρέπει να είναι αρκετά μεγάλη για να την δεχτεί. Έτσι, μια από τις προτάσεις για την άρση του συγκεκριμένου περιορισμού, είναι η δειγματολειπτική ανάγνωση των δεδομένων κατά το στάδιο υπολογισμού του συντελεστή παλινδρόμησης.

Τα αποτελέσματα του προγράμματος που υλοποιεί τον αλγόριθμο ODM αξιολογήθηκαν και ως προς το χρόνο που απαιτείται για να εκτελεστεί. Οι

μετρήσεις των χρόνων εκτέλεσης των προγραμμάτων πραγματοποιήθηκαν για την προαναφερθείσα συνθετική εικόνα και οι χρόνοι παρουσιάζονται στον παρακάτω πίνακα.

	Linux	Windows
ODM (IDL)	14 sec	6.5 sec
ODM (C)	10 sec	12 sec

Πίνακας 7.2. Σύγκριση χρόνων εκτέλεσης αλγόριθμου ODM σε IDL και C

Ο επεξεργαστής που επιλέχθηκε για να τρέξουν τα προγράμματα ήταν i5 με 8 νήματα. Παρατηρείται μια επιτάχυνση του προγράμματος σε C στο λειτουργικό σύστημα Linux σε σχέση με αυτό της IDL, χωρίς, όμως την αντίστοιχη επιτάχυνση στο λειτουργικό σύστημα Microsoft Windows. Αυτό συμβαίνει κατά πάσα πιθανότητα γιατί οι συναρτήσεις που καλεί το πρόγραμμα που έχει γραφεί σε γλώσσα προγραμματισμού IDL αξιοποιούν στο βέλτιστο τον παράλληλο προγραμματισμό στο περιβάλλον των Windows,. Από την άλλη μεριά, η C έχει περίπου την ίδια ταχύτητα και στα δύο λειτουργικά συστήματα.

Πρόγραμμα Εξαγωγής Καθαρών Στόχων

Για την αξιολόγηση του προγράμματος SEE-E το οποίο εμπεριέχει όλες τις λειτουργίες του προγράμματος SEE και γι αυτό επιλέχθηκε για αξιολόγηση, χρησιμοποιήθηκε ένα απόσπασμα της απεικόνισης της περιοχής Cuprite που έχει ληφθεί με το δέκτη AVIRIS. Για να εξακριβωθεί εάν οι φασματικές υπογραφές που εξάγει ως αποτέλεσμα το πρόγραμμα που υλοποιεί τον SEE-E είναι σωστές, υπολογίστηκαν οι φασματικές γωνίες τους, με τις φασματικές υπογραφές των πραγματικών καθαρών στόχων.

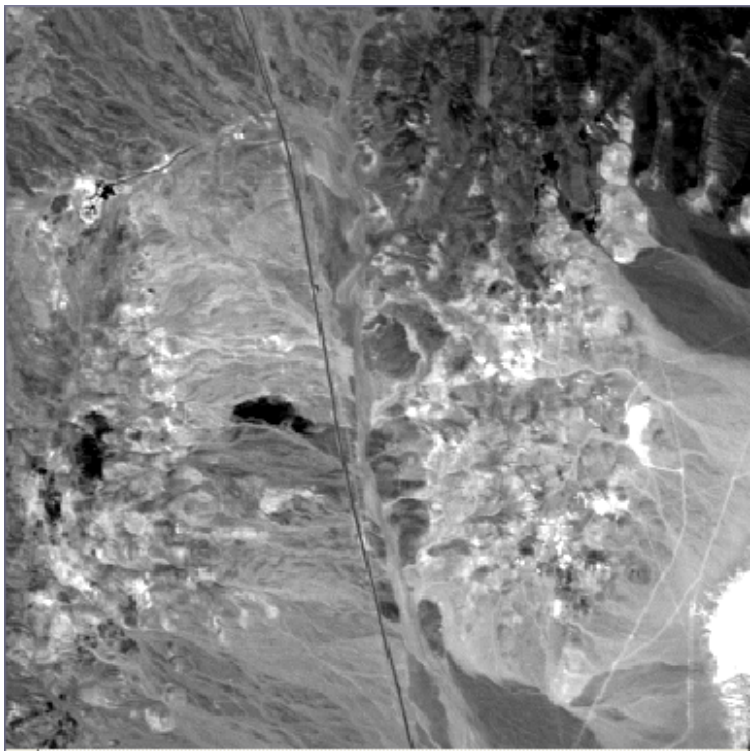
Οι φασματικές υπογραφές των καθαρών στόχων της συγκεκριμένης εικόνας έχουν προκύψει από επιτόπια έρευνα που έχει διεξαχθεί στην περιοχή και αντιστοιχούν στα κυρίαρχα ορυκτά που εντοπίζονται στην περιοχή. Οι φασματικές υπογραφές των συγκεκριμένων ορυκτών πάρθηκαν από την βιβλιοθήκη φασματικών υπογραφών της φασματικής βιβλιοθήκης USGS.

Στον παρακάτω πίνακα παρουσιάζονται οι φασματικές γωνίες του αποτελέσματος του E-SEE, με τους πραγματικούς καθαρούς στόχους και συγκρίνονται με τις φασματικές γωνίες που προέκυψαν από άλλες μεθόδους

εξαγωγής φασματικών χαρακτηριστικών, όπως η μέθοδος N-FINDR (Ανδρέου Χ. 2014) και η μέθοδος VCA (Ανδρέου Χ. 2014).

	Alunite	Buddingtonite	Calcite	Chabazite	Kaolinite	Muscovite	Sillimanite	Average
VCA	0.071	0.137	0.066	0.137	0.039	0.061	0.059	0.081
N-FINDR	0.075	0.139	0.069	0.137	0.066	0.062	0.056	0.086
SEE-E (IDL)	0.082	0.122	0.065	0.137	0.046	0.061	0.048	0.080
SEE-E (C)	0.068	0.129	0.065	0.140	0.038	0.061	0.052	0.079

Πίνακας 7.2. Σύγκριση αποτελεσμάτων αλγόριθμου SEE-E.



Εικόνα 7.3. Το πρώτο κανάλι της εικόνας Cuprite του δέκτη AVIRIS.

Το συμπέρασμα που εξάγεται από την εξέταση του παραπάνω πίνακα είναι ότι όλοι οι αλγόριθμοι καταλήγουν σε αποτελέσματα που βρίσκονται στην ίδια τάξη μεγέθους. Επίσης, ακολουθούν περίπου τις ίδιες διακυμάνσεις ανάλογα με τον εξεταζόμενο καθαρό στόχο. Τα δύο προγράμματα της μεθόδου SEE-E που υλοποιήθηκαν αντίστοιχα σε C και σε IDL εξάγουν περίπου την ίδια μέση φασματική γωνία. Οι καθαροί στόχοι τους οποίους εξάγουν έχουν την ίδια διάταξη κατά φασματική γωνία πράγμα που σημαίνει ότι δεν υπάρχει καμία απόκλιση μεταξύ των αποτελεσμάτων των δύο προγραμμάτων ως προς τους καθαρούς στόχους τους οποίους ανίχνευσαν. Οι αποκλίσεις που παρατηρούνται στη φασματική γωνία του κάθε καθαρού στόχου για τα δύο

προγράμματα αντίστοιχα μάλλον οφείλονται σε στρογγυλοποιήσεις πράξεων που εκτελούν.

Αντίστοιχη σύγκριση με αυτήν του αλγόριθμου ODM έγινε και για τον SEE-E, όσον αφορά τους χρόνους εκτέλεσης των δύο προγραμμάτων. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα.

	Linux	Windows
SEE-E (IDL)	58.5 sec	40 sec
SEE-E (C)	28 sec	31 sec

Πίνακας 7.4. Σύγκριση χρόνων εκτέλεσης αλγόριθμου SEE-E σε IDL και C

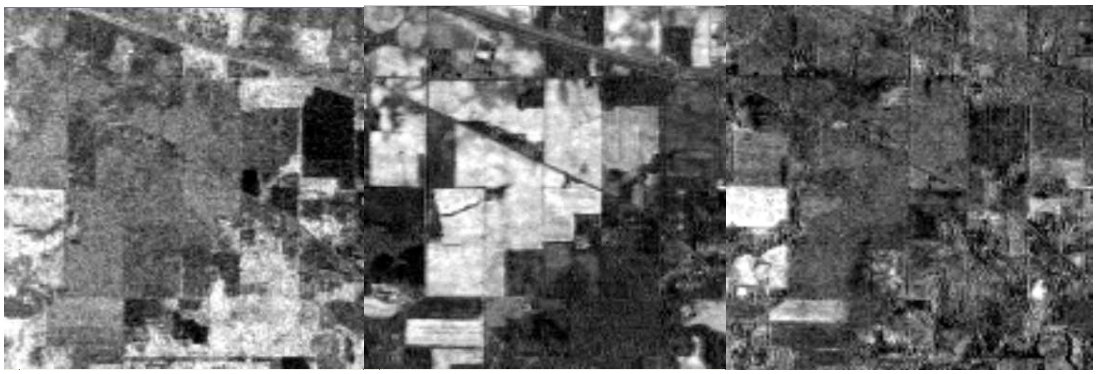
Από τον παραπάνω πίνακα εξάγεται το συμπέρασμα ότι ανεξάρτητα από το λειτουργικό σύστημα στο οποίο τρέχουν τα δύο προγράμματα, η έκδοση σε C εκτελείται ταχύτερα. Οι διαφορές στις εικόνες των δοκιμών μπορεί να είναι σχεδόν ανούσιες, όμως ο αλγόριθμος σε C δημιουργήθηκε για να λειτουργεί σε εικόνες εκατοντάδες φορές μεγαλύτερες από αυτές. Σε αυτή την περίπτωση, η επιτάχυνση αυτή θα παίζει σημαντικότερο ρόλο.

Πρόγραμμα Εκτίμησης της Αφθονίας

Όπως έχει αναφερθεί και σε προηγούμενα κεφάλαια, ο αλγόριθμος εκτίμησης της αφθονίας, εξάγει ως αποτέλεσμα μια εικόνα, με τιμές, τους συντελεστές αφθονίας κάθε καθαρού στόχου σε κάθε εικονοστοιχείο. Για τον υπολογισμό των συντελεστών αφθονίας σε κάθε εικονοστοιχείο, λαμβάνεται υπόψη η τιμή του εικονοστοιχείου σε κάθε κανάλι, χωρίς να υπάρχει τρόπος να διαχωριστεί η τιμή αυτή, σε σήμα ή θόρυβο. Οι συντελεστές αυτοί υπολογίζονται με την επίλυση του γραμμικού συστήματος, όπως έχει ήδη αναφερθεί. Επομένως, δεν εξασφαλίζεται υπολογιστικά, το ότι οι συντελεστές αυτοί θα είναι θετικοί και το άθροισμα τους θα είναι 1. Για τους παραπάνω λόγους, η εικόνα που εξάγει ο συγκεκριμένος αλγόριθμος, η οποία και παρουσιάζεται παρακάτω, φαίνεται θορυβώδης. Το γεγονός αυτό βέβαια δεν απαγορεύει στον χρήστη να έχει μία χωρική εκτίμηση του κάθε καθαρού στόχου, παρά το γεγονός ότι οι συντελεστές αφθονίας δεν έχουν κάποια φυσική σημασία.

Η αξιοπιστία του προγράμματος ελέγχθηκε με βάση τα αποτελέσματα του αντίστοιχου αλγορίθμου του τηλεπισκοπικού πακέτου ENVI. Η εικόνα της δοκιμής είναι η Indian Pines, που έχει αναφερθεί και παραπάνω. Οι

παρακάτω εικόνες αποτελούν έγχρωμα σύνθετα όπου, το κόκκινο προκύπτει από το αποτέλεσμα του ENVI και το πράσινο και το μπλε από το πρόγραμμα που διμιουργήθηκε. Το γεγονός ότι οι απεικονίσεις περιέχουν διαβαθμίσεις του γκρι και δεν διακρίνονται χρώματα υποδηλώνει την ομοιότητα των εικόνων.



Εικόνα 7.5. Τρία έγχρωμα σύνθετα του πρώτου, του έβδομου και του δωδέκατου καθαρού στόχου της εικόνας Indian Pines. (RGB EVNI, AFEM, AFEM)

8. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

Στο κεφάλαιο αυτό συνοψίζονται τα συμπεράσματα, που εξήχθησαν από τα αποτελέσματα της παρούσης διπλωματικής εργασίας. Αυτά αφορούν στους αλγόριθμους που προγραμματίστηκαν και αναφέρονται τόσο στο θεωρητικό, όσο και στο προγραμματιστικό επίπεδο. Από τα συμπεράσματα αυτά κατά κύριο λόγο προκύπτουν και οι προτάσεις και προοπτικές για περαιτέρω εξέλιξη των προγραμμάτων.

8.1. Συμπεράσματα

Ένα πρώτο συμπέρασμα που εξάγεται αμέσως από τα αποτελέσματα της παρούσης διπλωματικής εργασίας είναι ότι η διαδικασία του φασματικού διαχωρισμού μπορεί με τη χρήση των κατάλληλων μεθόδων να «αποσυνθέσει» το κάθε εικονοστοιχείο μιας υπερφασματικής απεικόνισης στα υλικά/αντικείμενα τα οποία το απαρτίζουν. Αυτός αποτελεί και έναν από τους βασικότερους στόχους της Υπερφασματικής Τηλεπισκόπησης.

Κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας, έγιναν δοκιμές στην ποιότητα των αποτελεσμάτων των μεθόδων. Οι δοκιμές αυτές, όπως φαίνεται και στο σχετικό κεφάλαιο, οδήγησαν στο συμπέρασμα ότι η χρήση της μεθόδου Multiple regression theory-based για την εκτίμηση του θορύβου, η οποία είναι απαραίτητη για την υλοποίηση του αλγορίθμου εκτίμησης της φασματικής διάστασης του υποχώρου του σήματος (ODM), δίνει καλύτερα αποτελέσματα από τη μέθοδο εκτίμησης του θορύβου Nearest Neighbor Distance (NND).

Επίσης, όπως διαπιστώθηκε κατά την αξιολόγηση των προγραμμάτων, η μέθοδος εκτίμησης της αφθονίας εξάγει ένα αρκετά θορυβώδες αποτέλεσμα. Αυτό γίνεται λόγω της έλλειψης δεσμεύσεων στον τρόπο υπολογισμού των συντελεστών αφθονίας. Η επιβολή δεσμεύσεων, λόγω του χρονικού περιορισμού της εκπόνησης της διπλωματικής εργασίας, θα αποτελέσει αντικείμενο περαιτέρω ενασχόλησης μετά το πέρας της διπλωματικής εργασίας.

Βασικός στόχος της διπλωματικής εργασίας ήταν η ανάπτυξη των συγκεκριμένων προγραμμάτων τα οποία υλοποιούν τα στάδια του φασματικού διαχωρισμού έξω από το πλαίσιο μιας γλώσσας προγραμματισμού ανωτέρου επιπέδου το οποίο θέτει περιορισμούς ως προς

τον όγκο των δεδομένων και τη βελτιστοποίηση του χρόνου. Άλλος στόχος της διπλωματικής εργασίας ήταν η παραγωγή εκτελέσιμων προγραμμάτων έτσι ώστε ο χρήστης να μην έχει ανάγκη το περιβάλλον ενός υπολογιστικού πακέτου. Επίσης τα προγράμματα που αναπτύχθηκαν θα έπρεπε να ακολουθούν τις αρχές του δομημένου προγραμματισμού, έτσι ώστε να είναι εύκολη η επαναχρησιμοποίησή τους σε επόμενα προγράμματα. Οι στόχοι αυτοί επιτεύχθηκαν, με κύριο πλεονέκτημα τη δυνατότητα εφαρμογής των αλγορίθμων αυτών σε δεδομένα μεγάλου όγκου. Επιπλέον, ο χρήστης των συγκεκριμένων αλγόριθμων δεν χρειάζεται να έχει αγοράσει και εγκαταστήσει στον υπολογιστή του κανένα άλλο πρόγραμμα. Η ταχύτητα εκτέλεσης των αλγορίθμων βελτιώθηκε σε σύγκριση με αυτή μιας γλώσσας προγραμματισμού ανώτερου επιπέδου όπως η IDL, ιδιαίτερα στο λειτουργικό σύστημα LINUX, επιτρέποντας έτσι στο χρήστη να εφαρμόσει τους αλγόριθμους σε μεγαλύτερες εικόνες και σε λιγότερο χρόνο.

8.2. Προοπτικές

Μια από τις προτάσεις για περαιτέρω ανάπτυξη, είναι ο προγραμματισμός της μεθόδου εκτίμησης της αφθονίας, με χρήση δεσμεύσεων. Οι δεσμεύσεις αυτές είναι η εξασφάλιση της μη αρνητικότητας των συντελεστών αφθονίας και το άθροισμα των συντελεστών αυτών σε κάθε εικονοστοιχείο θα είναι ίσο με 1. Αυτό θα εξαλείψει τις αρνητικές τιμές των εικονοστοιχείων του χάρτη αφθονίας, δίνοντας και φυσική σημασία στο αποτέλεσμα, και θα δώσει μια ενιαία κλίμακα ανάμεσα στους χάρτες των διαφορετικών καθαρών στόχων. Έτσι πλέον οι χάρτες αυτοί θα περιέχουν πραγματικά ποσοστά συμμετοχής.

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, κατά τη διάρκεια της εκτίμησης του θορύβου με τη χρήση της μεθόδου της πολλαπλής παλινδρόμησης, το αντίστοιχο πρόγραμμα χρειάζεται να αποθηκεύσει στην μνήμη του υπολογιστή όλη την αρχική εικόνα, γιατί συμμετέχει ολόκληρη στους υπολογισμούς. Αυτό θέτει έναν περιορισμό στο μέγεθος της εικόνας που μπορεί να επεξεργαστεί το πρόγραμμα, ο οποίος ορίζεται από τη φυσική μνήμη του υπολογιστή στον οποίο εκτελείται. Έτσι λοιπόν, μια πρόταση για περαιτέρω ανάπτυξη, είναι η άρση αυτού του περιορισμού.

Τελευταία πρόταση, είναι η περαιτέρω βελτιστοποίηση των αλγορίθμων και προγραμμάτων, για την επιτάχυνση των διαδικασιών. Ακόμα και κάποιες μικρές αλλαγές που επιφέρουν αντίστοιχα μικρές βελτιώσεις στην ταχύτητα των υπολογισμών, σε μεγάλους όγκους δεδομένων, όπως συμβαίνει στα

δεδομένα υπερφασματικών δεκτών, μπορεί να έχουν σημαντική βελτίωση του συνολικού χρόνου εκτέλεσης των προγραμμάτων.

9. ΒΙΒΛΙΟΓΡΑΦΙΑ

C. Andreou, V. Karathanassi, “Using Principal Component Analysis for endmember extraction”, proc. IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Lisbon, Portugal, 6-9 June 2011.

Andreou and V. Karathanassi, "Estimation of the number of endmembers using robust outlier detection method ", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,Vol. 7, Issue 1, 2014.

C. Andreou and V. Karathanassi, "New automated method for estimating the number of endmembers in hyperspectral images", proc. IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Shanghai, China 4-7 June, 2012,

Bioucas-Dias J and Nascimento J. “Hyperspectral Subspace Identification” IEEE Transactions on Geoscience and Remote Sensing, Vol. 46, No. 8, August 2008

Borengasser M. , Hungate W. , and Watkins R.: Hyperspectral Remote Sensing Principles and Applications CRC Press (2008)

Chang C.: “Hyperspectral Data Exploitation Theory and Applications” WILEY (2007)

Chang C. and Du Q.: “Estimation of Number of Spectrally Distinct Signal Sources in Hyperspectral Imagery” IEEE Transactions on Geoscience and Remote Sensing, Vol. 42, No. 3, March 2004

Chapman B., Jost G. and Van der Pas R.: “Using OpenMP: Portable Shared Memory Parallel Programming“ The MIT Press (2008)

Green A., Berman M., Switzer P. and Craig M., “A Transformation for Ordering Multispectral Data in Terms of Image Quality with Implications for Noise Removal” IEEE Transactions on Geoscience and Remote Sensing, Vol. 26, No. 1, January 1988

Hawkins D. M.: “Identification of Outliers” Chapman and Hall, 1980

Sykas D., Karathanassi V., Topouzelis K., “Development of a New Automatic Relative Radiometric Normalization Methodology for Multispectral and Hyperspectral Images”, Journal of Remote Sensing Technology (JRST), Vol. 1, Issue 3, 2013.

Kernighan B. and Ritchie D. “The C Programming Language Second Edition” (Prentice-Hall 1988)

Keshava N. and J. F. Mustard: “Spectral unmixing”. IEEE Signal Processing Magazine, 19: pp. 44–57, 2002.

Press W., Teukolsky S., Vetterling W. and Flannery B. “Numerical Recipes - The Art of Scientific Computing - Third Edition” Cambridge Press (2007)

Smith L.: “A tutorial on Principal Components Analysis” February 26, 2002

Ανδρέου Χαρούλα: “Δυνατότητες και Περιορισμοί της Υπερφασματικής Τηλεπισκόπησης στην Ανίχνευση Ποιοτικών Χαρακτηριστικών του Οδοστρώματος”, Διπλωματική Εργασία, ΣΑΤΜ-ΕΜΠ, Αθήνα, 2008.

Ανδρέου Χαρούλα: “Development of Spectral Unmixing Methods for Hyperspectral Data Exploitation” Διδακτορική Διατριβή, ΣΑΤΜ-ΕΜΠ, Αθήνα, 2014

Αργιαλάς Δ.: “Ψηφιακή Τηλεπισκόπηση”, ΕΜΠ, Αθήνα 1998.
.

10. ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα παρουσιάζεται ο κώδικας του προγράμματος unmixing ο οποίος περιλαμβάνει όλους τους αλγόριθμους υλοποίησης του φασματικού διαχωρισμού. Εναλλακτικά ο χρήστης μπορεί να εκτελέσει τα επιμέρους προγράμματα τα οποία αντιστοιχούν σε κάθε μια από τις μεθόδους που απαρτίζουν τα στάδια του φασματικού διαχωρισμού. Ο κώδικας του κάθε ενός από αυτά τα προγράμματα, μαζί με τις οδηγίες για τη χρήση και τη μεταγλώττιση τους περιέχεται στο επισυναπτόμενο CD.

```
/*  Unmixing - Performs spectral unmixing of hyperspectral Images
    Copyright (C) 2014  Dimitris Kefalos

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as published
by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see
<http://www.gnu.org/licenses/>.*/

# include <stdio.h>
# include <stdlib.h>
# include <math.h>
# include <string.h>
# include <omp.h>
# include <time.h>

#define FP_TYPE double
FP_TYPE mySQRT(FP_TYPE);
#define myABS(x) (((x)<0)?-(x):(x))
#define mySQR(x) ((x)*(x))
#define EPSILON 1.e-19
#define ROTATE(a,i,j,k,l) {g=a[i][j];h=a[k][l];a[i][j]=g-
s*(h+g*tau);a[k][l]=h+s*(g-h*tau);}

/* Function Definitions */

void jacobi(double **, int, double *, double **, int *);
void eigsort(double *, double **, int);
int MNF(FILE*, FILE*, FILE*, int, int, int, int, int, int);
int PCA(FILE*, FILE*, int, int, int, int, FILE*, int, int);
int ODM( FILE*, FILE*, int, int, int, int*, int, int, int);
int image_add(FILE*, FILE*, FILE*, int, int, int, FILE*, int);
int SEE_E(FILE*, FILE*, FILE*, FILE*, int, int , int, int, double**,
int, int);
```

```
int Unmixing(FILE*, FILE*, int, int, int, int , char*, double**,
int);
int Matrix_inversion(FP_TYPE*, FP_TYPE*, int);
int Cholesky_decomposition(FP_TYPE *, FP_TYPE *, int );
int NND(FILE*, FILE*, int, int, int, int);
int MRTBM(FILE*, FILE*, FILE*, int, int, int, int);

int compare (const void * x, const void * y){

double da = *(const double *)x;
double db = *(const double *)y;

if (da == db)
return 0;

return (da > db) ? -1 : 1;
}

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

FILE     *fin, *report, *report2, *fsignal, *fplus;
int      rows, cols, bands, nendmembers, i, j, k, m, nthreads, opt,
nvalue, npixels;
char     ersfile[100], binfile[100], tiffile[100], sysstring[200],
txtfile[100], enddir[100], txtfile2[100];
float    *buffer;
double   **a;

if(argc!=7){
    puts(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * *");
    puts(" *
*");
    puts(" * Unmixing:   Use MNF, ODM, SEE-E and AFEM Methods
*");
    puts(" *
*");
    puts(" *         -   Input parameters (at command line) :
*");
    puts(" *                 -        Image filename
*");
    puts(" *                 -        Number of Rows
*");
    puts(" *                 -        Number of Columns
*");
    puts(" *                 -        Number of Bands
*");
    puts(" *                 -        Transformation Option :
*");
    puts(" *                          1 for MNF with NND
*");
    puts(" *                          2 for MNF with MRTBM
*");
    puts(" *                          3 for PCA instead of MNF
*");
    puts(" *                 -        Null Cell Value
*");
    puts(" *
*");
    puts(" *         -   Works on the file types supported by GDAL
*");
```

```
    puts(" *
*");
    puts(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * *");
    puts(" *
*");
    puts(" * Output : - Report :
Imagename_unmixing_report.txt   *");
    puts(" *          - Endmembers File :
Imagename_endmembers.txt        *");
    puts(" *          - Abundance Fraction Image :
Imagename_abuntance_fraction    *");
    puts(" *
*");
    puts(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * *");
    puts(" * Unmixing    by Dimitris Kefalos  (dkefalos@gmail.com)
*");
    puts(" *
*");
    puts(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * *");
    system("PAUSE");
    exit(1);
    }

/* File Manipulation */

strcpy(tiffile, argv[1]);
i=(strlen(tiffile)-4);
memcpy(binfile, tiffile, strlen(tiffile)-4);
binfile[i]='\0';
i=(strlen(binfile));
memcpy(ersfile, binfile, strlen(binfile));
ersfile[i]='\0';
sprintf(ersfile, "%s.ers", binfile);
memcpy(txtfile, binfile, strlen(binfile));
txtfile[i]='\0';
sprintf(txtfile, "%s_unmixing_report.txt", binfile);
i=(strlen(binfile));
txtfile2[i]='\0';
sprintf(txtfile2, "%s_endmembers.txt", binfile);
memcpy(enddir, binfile, strlen(binfile));
enddir[i]='\0';
sprintf(enddir, "%s_abundance_fraction", binfile);

/* Argument Handling */

rows=atoi(argv[2]);
cols=atoi(argv[3]);
bands=atoi(argv[4]);
opt=atoi(argv[5]);
nvalue=atoi(argv[6]);

if (opt!=1 && opt!=2 && opt!=3){
    puts("Please choose valid noise estimation method");
    exit(-1);
}

if (rows<0){
    puts("Please give a valid number of rows");
```

```
    exit(-1);
}

if (cols<0){
    puts("Please give a valid number of columns");
    exit(-1);
}

if (bands<0){
    puts("Please give a valid number of bands");
    exit(-1);
}

/* GDAL Call */

sprintf(sysstring, "gdal_translate -ot Float32 -of ERS %s %s",
tiffile, ersfile);
printf("\n");
printf(sysstring);
printf("\n");
if ((i=system(sysstring))!=0 && (i=system(sysstring))!=1 ){
    printf("\nGDAL did not run correctly %d", i);
    exit(-1);
    }
puts("GDAL has Executed");

/* File Opening */

if ( (fin=fopen(binfile , "rb")) == NULL){
    printf("\nCannot open %s file.\n", binfile);
    exit (-1);
    }

if ( (report=fopen(txtfile , "wt")) == NULL){
    printf("\nCannot open %s file.\n", txtfile);
    exit (-1);
    }

if ( (report2=fopen(txtfile2 , "wt")) == NULL){
    printf("\nCannot open %s file.\n", txtfile2);
    exit (-1);
    }

if ((fsignal=tmpfile()) == NULL){
    printf("\nCannot open signal file.\n");
    exit (-1);
    }

if ( (fplus=tmpfile()) == NULL){
    printf("\nCannot open PCA1 file.\n");
    exit (-1);
    }

if ( (a = malloc(bands*sizeof(double*))) == NULL){
    puts("\nBad Memory Allocation.\n");
    return(-1);
    }

if ( (buffer = malloc(cols*bands*sizeof(float))) == NULL){
    puts("\nBad Memory Allocation.\n");
    return(-1);
```

```
    }

nthreads=omp_get_max_threads();
printf("Number of Threads : %d\n\n", nthreads);

/* Masked Pixels Count */

npixels=0;
for (i=0; i<rows; i++){
    fread(buffer, cols*bands, sizeof(float), fin);
    for (j=0; j<cols; j++){
        m=0;
        for (k=0; k<bands; k++){
            if (buffer[k*cols+j]==nvalue){
                m++;
                }
            }
        if (m==bands){
            npixels++;
            }
        }
    }
rewind(fin);
printf("Masked pixels=%d\n", npixels);

/* Transformation Option */

if (opt==1 || opt ==2){
    MNF(fin, fsignal, report, rows, cols, bands, opt, nvalue,
npixels);
    }
else if (opt==3){
    puts("Executing PCA:");
    PCA(fin, report, rows, cols, bands, bands, fsignal, nvalue,
npixels);
    }

/* Number of Endmembers Estimation */

ODM(fsignal, report, rows, cols, bands, &nendmembers, opt, nvalue,
npixels);
puts("ODM Executed");

for (i=0; i< bands; i++){
    if ( (a[i] = malloc(nendmembers*sizeof(double))) == NULL){
        puts("\nBad Memory Allocation.\n");
        return(-1);
        }
    }

/* Endmember Spectal Signature Extraction */

image_add(fin, fsignal, report, rows, cols, bands, fplus, nvalue);
SEE_E(fplus, fin, report, report2, rows, cols, bands, nendmembers, a,
nvalue, npixels);
puts("\nSEE-E Executed");

/* Endmember Abundance Estimation */

Unmixing(fin, report, rows, cols, bands, nendmembers, enddir, a,
nvalue);
```

```
puts("\nUnmixing Executed");

/* Closing */

for (i=0; i<bands; i++){
    free(a[i]);
}
free(a);
free(buffer);
fflush(NULL);

if (fclose(fplus)!=0){
   puts("Error in fplus closing");
   }

if (fclose(fsignal)!=0){
   puts("Error in fsignal closing");
   }

if (fclose(report2)!=0){
   puts("Error in report2 closing");
   }

if (fclose(report)!=0){
   puts("Error in report closing");
   }

if (fclose(fin)!=0){
   puts("Error in fin closing");
   }

return(0);
}

int MNF(FILE *fin, FILE *fsignal, FILE* report, int rows, int cols,
int bands, int opt, int nvalue, int npixels){

FILE    *fnoise, *fpca1;
int     i, j, k, l, nrot;
float   *buffer, *buffer2, *sum;
double  *Nmean, **Ncov, *Neigval, **Neigvec, **sumv, tstart, tend;

/* File Opening */

if ((fpca1=tmpfile()) == NULL){
    printf("\nCannot open signal file.\n");
    exit (-1);
    }

if ((fnoise=tmpfile()) == NULL){
    printf("\nCannot open noise file.\n");
    exit (-1);
    }

/* Memory Allocation */

if ((buffer2=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }
```

```
if ((buffer=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((sum=calloc(bands,sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ( (sumv=calloc(bands,sizeof(double*))) == NULL ){
    puts("Error in Memory Allocation.");
    exit(-1);
    }

if ((Nmean=malloc(bands*sizeof(double)))== NULL){
    puts("Error in Nmean Memory Allocation");
    exit(-1);
    }

if ( (Ncov=malloc(bands*sizeof(double*))) == NULL ){
    puts("Error in Ncov Memory Allocation.");
    exit(-1);
    }

if ( (Neigval=malloc(bands*sizeof(double))) == NULL ){
    puts("Error in Neigenval Memory Allocation.");
    exit(-1);
    }

if ( (Neigvec=malloc(bands*sizeof(double*))) == NULL ){
    puts("Error in Neigenvec Memory Allocation.");
    exit(-1);
    }

for (i=0; i<bands; i++){
    if ( (Ncov[i]=malloc(bands*sizeof(double))) == NULL ){
        puts("Error in (Ncov[i]) Memory Allocation.");
        exit(-1);
        }
    if ( (Neigvec[i]=malloc(bands*sizeof(double))) == NULL ){
        puts("Error in (Neigenvec[i]) Memory Allocation.");
        exit(-1);
        }
    if ((sumv[i]=calloc(bands,sizeof(double))) == NULL){
        puts("Error in Memory Allocation.");
        exit(-1);
        }
}

/* Noise Estimation Option */

if (opt==1){
    NND(fin, fnoise, rows, cols, bands, nvalue);
    puts("Noise calculated using NND Method");
}

if (opt==2){
    MRTBM(fin, fnoise, report, rows, cols, bands, nvalue);
    puts("Noise calculated using MRTBM Method");
}
```

```
/* Noise Sum Calculation for Every Band */

rewind(fnoise);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fnoise);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
        }
    }

/* Noise Mean Calculation for Every Band */

for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }

fprintf(report, "\nNoise Mean :\n");
for (i=0; i<bands; i++){
    fprintf(report, "Band %d : %10.5lf\n", i+1, Nmean[i]);
    }

/* sumv Calculation for Every Band */

rewind(fnoise);
tstart=omp_get_wtime();

# pragma omp parallel default(none) private(i, j, k, l) shared(rows,
cols, bands, fnoise, buffer, sumv, Nmean, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fnoise);
    printf("Calculating Image Statistics at row %d of %d\r", i+1,
rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[(j*cols)+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-
Nmean[j])*(buffer[(k*cols)+l]-Nmean[k]);
                    }
                }
            }
        }
    }
}

tend=omp_get_wtime();
printf("\nDuration of Statistics Parallel region : %2.1lf sec\n",
tend-tstart);

fprintf(report, "\nsumv:\n");
for (i=0; i<bands; i++){
```

```
    fprintf (report, "Band %d", i+1);
    for (j=0; j<bands; j++){
        fprintf(report, " %10.5lf", sumv[i][j]);
        }
    fprintf(report, "\n");
    }

/* Noise Variance-Covariance Matrix Calculation */

for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        Ncov[i][j]=sumv[i][j]/(rows*cols-npixels);
        }
    }

fprintf(report, "\nNoise Variance-Covariance Matrix:\n");
for (i=0; i<bands; i++){
    fprintf (report, "Band %d", i+1);
    for (j=0; j<bands; j++){
        fprintf(report, " %10.5lf", sqrt(fabs(Ncov[i][j])));
        }
    fprintf(report, "\n");
    }

/* Noise Eigenvectors Calculation and Sorting */

jacobi(Ncov, bands, Neigval, Neigvec, &nrot);
eigsort(Neigval, Neigvec, bands);
fflush(NULL);

fprintf(report, "\nNoise Eigenvectors:\n");
for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        fprintf(report,"%10.5f ", Neigvec[i][j]);
        }
    fprintf(report, "\n");
    }

fprintf(report,"\nNoise Eigenvalues :\n");
for (i=0; i<bands; i++){
    fprintf(report," %10.5lf\n", Neigval[i]);
    }

for (i=0; i<bands; i++){
    sum[i]=0.0;
    for (j=0; j<bands; j++){
        sumv[i][j]=0.0;
        }
    }

/* Sum Calculation for Every Band */

rewind(fin);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fin);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
```

```
        }
    }

/* Mean Calculation for Every Band */

for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }

fprintf(report, "\nData Mean :\n");
for (i=0; i<bands; i++){
    fprintf(report, "Band %d : %10.5lf\n", i+1, Nmean[i]);
    }

/* White Noise Image Creation */

rewind(fin);
tstart=omp_get_wtime();
# pragma omp parallel default(none) private (i, j, k , l)
shared(buffer, buffer2, rows, cols, bands, Neigvec, fin, fpca1,
Nmean, Ncov, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fin);
    printf("Calculating White Noise Image at row %d of %d\r", i+1,
rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            buffer2[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
                if (buffer[l*cols+k]!=nvalue){
                    buffer2[j*cols+k]+=1.*(buffer[(l*cols)+k]-
Nmean[l])*(Neigvec[l][j]);
                    }
                else {
                    buffer2[j*cols+k]=nvalue;
                    }
                }
            if (buffer2[j*cols+k]!=nvalue){

buffer2[j*cols+k]=buffer2[j*cols+k]/sqrt(fabs(Ncov[j][j]));
                }
            }
        }
    # pragma omp single
    {
    fwrite(buffer2,sizeof(float),bands*cols,fpca1);
    }
    }
}
tend=omp_get_wtime();
printf("\nDuration of Noise Whitening Parallel region : %2.1lf
sec\n", tend-tstart);

for (i=0; i<bands; i++){
    sum[i]=0.0;
    for (j=0; j<bands; j++){
```

```
        sumv[i][j]=0.0;
        }
    }

/* White Noise Image Sum Calculation for Every Band */

rewind(fpca1);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fpca1);
    for(j=0; j<bands; j++){
        for(k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[j*cols+k];
                }
            }
        }
    }

/* White Noise Image Mean Calculation for Every Band */

for (i=0; i<bands; i++){
    Nmean[i]=sum[i]/(rows*cols-npixels);
    }

fprintf(report, "\nWhite Noise Mean :\n");
for (i=0; i<bands; i++){
    fprintf(report, "Band %d : %10.5lf\n", i+1, Nmean[i]);
    }

/* White Noise Image sumv Calculation for Every Band */

rewind(fpca1);
tstart=omp_get_wtime();
# pragma omp parallel default(none) private(i, j, k, l)
shared(buffer, fpca1, Nmean, sumv, rows, cols, bands, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fpca1);
    printf("Calculating White Noise Image Statistics at row %d of
%d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[(j*cols)+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-
Nmean[j])*(buffer[(k*cols)+l]-Nmean[k]);
                    }
                }
            }
        }
    }
}

tend=omp_get_wtime();
printf("\nDuration of Statistics Parallel region : %2.1lf sec\n",
tend-tstart);
```

```
fprintf(report, "\nWhite sumv:\n");
for (i=0; i<bands; i++){
    fprintf (report, "Band %d", i+1);
    for (j=0; j<bands; j++){
        fprintf(report, " %10.5lf", sumv[i][j]);
        }
    fprintf(report, "\n");
    }

/* White Noise Image Covariance Matrix Calculation */

for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        Ncov[i][j]=sumv[i][j]/(rows*cols-npixels);
        }
    }

fprintf(report, "\nWhite Noise Variance-Covariance Matrix:\n");
for (i=0; i<bands; i++){
    fprintf (report, "Band %d", i+1);
    for (j=0; j<bands; j++){
        fprintf(report, " %15.5lf", Ncov[i][j]);
        }
    fprintf(report, "\n");
    }

/* White Noise Image Eigenvectors and Sorting */

jacobi(Ncov, bands, Neigval, Neigvec, &nrot);
eigsort(Neigval, Neigvec, bands);

fprintf(report, "\nWhite Noise Eigenvectors:\n");
for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        fprintf(report,"%15.5f ", Neigvec[i][j]);
        }
    fprintf(report, "\n");
    }

fflush(NULL);

/* MNF Image Creation */

tstart=omp_get_wtime();
rewind(fpca1);
# pragma omp parallel default(none) private(i, j, k, l)
shared(buffer, buffer2, rows, cols, bands, fpca1, fsignal, Neigvec,
Nmean, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fpca1);
    printf("Calculating Pr. Components of Wh. Noise Image at row %d
of %d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            buffer2[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
```

```
                if (buffer[l*cols+k]!=nvalue){
                    buffer2[j*cols+k]+=1.*(buffer[(l*cols)+k]-
Nmean[l])*(Neigvec[l][j]);
                    }
                else {
                    buffer2[j*cols+k]=nvalue;
                    }
                }
            }
        }
    # pragma omp single
    {
    fwrite(buffer2,sizeof(float),bands*cols,fsignal);
    }
    }
}
tend=omp_get_wtime();
printf("\nDuration of Wh. Noise PCA Parallel region : %2.1lf sec",
tend-tstart);
fflush(NULL);

/* Closing MNF */

for(i=0; i<bands; i++){
    free(sumv[i]);
    free(Neigvec[i]);
    free(Ncov[i]);
    }
free(Neigvec);
free(Neigval);
free(Ncov);
free(Nmean);
free(sumv);
free(sum);
free(buffer);
free(buffer2);
if (fclose(fpca1)!=0){
   puts("Error in fpca1 closing");
   }
if (fclose(fnoise)!=0){
   puts("Error in fnoise closing");
   }

puts("\nMNF Executed");
return (0);
}

int ODM (FILE* fsignal, FILE* report, int rows, int cols, int bands,
int* nendmembers, int opt, int nvalue, int npixels){

int       i, j, k, p;
double    *sum, *mean, *var, *dist, threshold, varmax, varmin,
*dvar2;
float     *buffer;

/* Memory Allocation */

if ( (buffer=malloc(bands*cols*sizeof(float))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }
```

```
if ( (sum=calloc(bands,sizeof(double))) == NULL ){
    puts("Error in (sum) Memory Allocation.");
    exit(-1);
    }

if ( (mean=calloc(bands,sizeof(double))) == NULL ){
    puts("Error in (mean) Memory Allocation.");
    exit(-1);
    }

if ( (var=calloc(bands,sizeof(double))) == NULL ){
    puts("Error in (var) Memory Allocation.");
    exit(-1);
    }

if ( (dvar2=calloc((bands),sizeof(double))) == NULL ){
    puts("Error in (dvar2) Memory Allocation.");
    exit(-1);
    }

if ( (dist=calloc(bands-1,sizeof(double))) == NULL ){
    puts("Error in (dist) Memory Allocation.");
    exit(-1);
    }

/* Sum Calculation for Every Band */

rewind(fsignal);
for (i=0; i<rows; i++){
    fread (buffer, sizeof(float), cols*bands, fsignal);
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[(j*cols)+k];
                }
            }
        }
    }

fprintf(report,"\nSum :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, sum[i]);
    }

/* Mean Calculation for Every PC */

for (i=0; i<bands; i++){
    mean[i]=sum[i]/(rows*cols-npixels);
    }

fprintf(report,"\nMean :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, mean[i]);
    }

/* Variance Calculation of signal image PCs */

rewind(fsignal);
# pragma omp parallel default(none) private(i, j, k) shared(buffer,
mean, var, rows, cols, bands, fsignal, nvalue)
```

```
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fsignal);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                var[j]+=(buffer[(j*cols)+k]-
mean[j])*(buffer[(j*cols)+k]-mean[j]);
                }
            }
        }
    }
}

for (i=0; i<bands; i++){
    var[i]=sqrt(var[i]/(rows*cols-npixels));
    }

fprintf(report,"\nVariance :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, var[i]);
    }

/* Std Deviation Normalization Option */

if (opt==1 || opt==2){
    varmax=0;
    varmin=5000;
    for (i=0; i<bands; i++){
        if (var[i]>varmax){
            varmax=var[i];
            }
        if (var[i]<varmin){
            varmin=var[i];
            }
        }
    fprintf(report, "varmax = %10.5lf, varmin= %5.5lf\n", varmax,
varmin);
    for (i=0; i<bands; i++){
        var[i]=(var[i]-varmin)/(varmax-varmin);
        }
    fprintf(report,"\nNormalised Variance :\n");
    for (i=0; i<bands; i++){
        fprintf(report,"Band %d : %10.5lf\n", i+1, var[i]);
        }
    }

/* Sorting Standard Deviations in Descending Order */

for (i=0; i<bands; i++){
    dvar2[i]=var[bands-i-1];
    }

fprintf(report,"\nSorted Normalised dVariance :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, dvar2[i]);
    }
```

```
/* Eucleidian Distance Calculation */

for (i=0; i<bands-1; i++){
    dist[i]=sqrt((dvar2[i]-dvar2[i+1])*(dvar2[i]-dvar2[i+1])+1);
    }

fprintf(report,"\ndist :\n");
for (i=0; i<bands-1; i++){
    fprintf(report,"Band %d : %10.10lf\n", i, dist[i]);
    }

/* Threshold Calculation */

threshold=dist[bands*1/4]+1.5*(fabs(dist[bands*1/4]-
dist[bands*3/4]));

fprintf(report,"\nthreshold= %10.10lf, Q3=Q[%d]=%10.10lf,
Q1=Q[%d]=%10.10lf", threshold, bands*3/4, dist[bands*3/4], bands*1/4,
dist[bands*1/4]);

/* Number Of Endmambers Calculation */

p=0;
for (i=0; i<bands-1; i++){
    if (dist[bands-2-i]<threshold){
        p=i+1;
        break;
        }
    }
*nendmembers=p;
printf("\nNumber of endmembers = %d\n", *nendmembers);
fprintf(report, "\nNumber of endmembers = %d\n", *nendmembers);

/* Closing ODM */

fflush(NULL);
free(dist);
free(dvar2);
free(var);
free(mean);
free(sum);
free(buffer);
return (0);
}

int image_add(FILE* fin, FILE* fsignal, FILE* report, int rows, int
cols, int bands, FILE* fplus, int nvalue){

FILE    *fpca2;
int     maxrow, maxcol, i, j;
float   *buffer, *buffer2, *buffer3, *mpv, smax;

/* Memory Allocacion */

if ((fpca2=tmpfile()) == NULL){
    printf("\nCannot open pca2 file.\n");
    exit (-1);
    }

if ( (buffer = malloc(cols*sizeof(float))) == NULL){
```

```
    puts("\nBad Memory Allocation.\n");
    return(-1);
    }

if ( (buffer2 = malloc(cols*bands*sizeof(float))) == NULL){
    puts("\nBad Memory Allocation.\n");
    return(-1);
    }

if ( (buffer3 = malloc(3*cols*bands*sizeof(float))) == NULL){
    puts("\nBad Memory Allocation.\n");
    return(-1);
    }

if ( (mpv = malloc(bands*sizeof(float))) == NULL){
    puts("\nBad Memory Allocation.\n");
    return(-1);
    }

/* Maximum Projected Value */

smax=-500000.0;
rewind(fsignal);
for (i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols, fsignal);
    for (j=0; j<cols; j++){
        if (buffer[j]!=nvalue){
            if (smax<buffer[j]){
                smax=buffer[j];
                maxrow=i;
                maxcol=j;
                }
            }
        }
    fseek(fsignal, cols*(bands-1)*sizeof(float), SEEK_CUR);
    }

fprintf(report, "\nsmax= %5.5lf maxrow= %d maxcol= %d", smax, maxrow,
maxcol);

/* Maximum Projected Value Signature */

fseek(fin, maxrow*cols*bands*sizeof(float), SEEK_SET);
fread(buffer2, sizeof(float), cols*bands, fin);
for(i=0; i<bands; i++){
    mpv[i]=buffer2[i*cols+maxcol];
    }

/* Image Add Preparation */

for (i=0; i<bands; i++){
    for (j=0; j<cols*3; j++){
        buffer3[i*cols*3+j]=mpv[i];
        }
    }

/* Image Add Writing */

rewind(fin);
for (i=0 ; i<rows; i++){
    for (j=0; j<bands; j++){
```

```
        fread(buffer2, sizeof(float), cols, fin);
        fwrite(buffer2, sizeof(float), cols, fplus);
        fwrite(buffer3+j*cols*3, sizeof(float), cols*3, fplus);
        }
    }
puts("Image add Finished\n");

/* Closing Image Add */

free(mpv);
free(buffer3);
free(buffer2);
free(buffer);
if (fclose(fpca2)!=0){
   puts("Error in fpca2 closing");
   }
return (0);
}

int SEE_E(FILE* fplus, FILE* fin, FILE* report, FILE* report2, int
rows, int cols, int bands, int nendmembers, double **a, int nvalue,
int npixels){

FILE     *fpca4;
int      *coordsr, *coordsc, *endm, c, i, j, k, l, *tempint;
float    *hmin, *hmax, *buffer, *buffer2, **ssignature, threshold,
**dif;
double   *sum, **pro, **sqr, **sad, *ssad, ssadmin, ssadmax;

/* File Opening */

if ((fpca4=tmpfile()) == NULL){
    printf("\nCannot open pca4 file.\n");
    exit (-1);
    }
fflush(NULL);

/* Memory Allocation */

if ( (hmin = malloc((nendmembers-1)*sizeof(float))) == NULL){
    puts("\nBad hmin Memory Allocation.\n");
    return(-1);
    }

if ( (hmax = malloc((nendmembers-1)*sizeof(float))) == NULL){
    puts("\nBad hmax Memory Allocation.\n");
    return(-1);
    }

if ( (coordsr = malloc(2*(nendmembers-1)*sizeof(int))) == NULL){
    puts("\nBad coordsr Memory Allocation.\n");
    return(-1);
    }

if ( (coordsc = malloc(2*(nendmembers-1)*sizeof(int))) == NULL){
    puts("\nBad coordsc Memory Allocation.\n");
    return(-1);
    }

if ( (ssignature = malloc(2*(nendmembers-1)*sizeof(float*))) ==
NULL){
```

```
    puts("\nBad ssingnature Memory Allocation.\n");
    return(-1);
    }

if ( (sum = calloc(2*(nendmembers-1),sizeof(double))) == NULL){
    puts("\nBad sum Memory Allocation.\n");
    return(-1);
    }

if ( (pro = calloc(2*(nendmembers-1),sizeof(double*))) == NULL){
    puts("\nBad pro Memory Allocation.\n");
    return(-1);
    }

if ( (sqr = calloc(2*(nendmembers-1),sizeof(double*))) == NULL){
    puts("\nBad sqr Memory Allocation.\n");
    return(-1);
    }

if ( (buffer = malloc(cols*4*(nendmembers-1)*sizeof(float))) ==
NULL){
    puts("\nBad buffer Memory Allocation.\n");
    return(-1);
    }

if ( (buffer2 = malloc(cols*bands*sizeof(float))) == NULL){
    puts("\nBad buffer2 Memory Allocation.\n");
    return(-1);
    }

if ( (ssad = calloc(2*(nendmembers-1),sizeof(double))) == NULL){
    puts("\nBad ssad Memory Allocation.\n");
    return(-1);
    }

if ( (sad = calloc(2*(nendmembers-1),sizeof(double*))) == NULL){
    puts("\nBad sad Memory Allocation.\n");
    return(-1);
    }

if ( (dif = calloc(2*(nendmembers-1),sizeof(float*))) == NULL){
    puts("\nBad dif Memory Allocation.\n");
    return(-1);
    }

if ( (endm = calloc(2*(nendmembers-1),sizeof(int))) == NULL){
    puts("\nBad end Memory Allocation.\n");
    return(-1);
    }

if ( (tempint = calloc(2*(nendmembers-1),sizeof(int))) == NULL){
    puts("\nBad tempint Memory Allocation.\n");
    return(-1);
    }

for (i=0; i<(2*(nendmembers-1)); i++){
    if ( (ssignature[i] = malloc(bands*sizeof(float))) == NULL){
        puts("\nBad ssignature[i] Memory Allocation.\n");
        return(-1);
        }
```

```
    if ( (pro[i] = calloc(2*(nendmembers-1),sizeof(double))) ==
NULL){
        puts("\nBad pro[i] Memory Allocation.\n");
        return(-1);
        }

    if ( (sqr[i] = calloc(2*(nendmembers-1),sizeof(double))) ==
NULL){
        puts("\nBad sqr[i] Memory Allocation.\n");
        return(-1);
        }
     if ( (sad[i] = calloc(2*(nendmembers-1),sizeof(double))) ==
NULL){
        puts("\nBad sad[i] Memory Allocation.\n");
        return(-1);
        }
    if ( (dif[i] = calloc(2*(nendmembers-1),sizeof(float))) == NULL){
        puts("\nBad dif[i] Memory Allocation.\n");
        return(-1);
        }
    }

/* PCA on Enchanced Image */

fflush(NULL);
puts("PCA on Enchanced Image");
PCA(fplus, report, rows, cols*4, bands, nendmembers-1, fpca4, nvalue,
npixels);
fflush(NULL);

/* Min-Max Calculation for Every Band */

for(i=0; i<(nendmembers-1); i++){
    hmin[i]=500000.;
    hmax[i]=-500000.;
    }

//fseek(fpca4, cols*(nendmembers-1)*4*sizeof(float), SEEK_SET);
rewind(fpca4);
for (i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*(nendmembers-1)*4, fpca4);
    for (j=0; j<(nendmembers-1); j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols*4+k]!=nvalue){
                if (buffer[j*cols*4+k]<hmin[j]){
                    hmin[j]=buffer[j*cols*4+k];
                    coordsr[j]=i;
                    coordsc[j]=k;
                    }
                if (buffer[j*cols*4+k]>hmax[j]){
                    hmax[j]=buffer[j*cols*4+k];
                    coordsr[(nendmembers-1)+j]=i;
                    coordsc[(nendmembers-1)+j]=k;
                    }
                }
            }
        }
    }

for (i=0; i<nendmembers-1; i++){
    fprintf(report, " %10.5lf  %10.5lf\n", hmin[i], hmax[i]);
```

```
    }

/* Endmember Canditate Signature Reading */

rewind(fin);
for (i=0; i<2*(nendmembers-1); i++){
    fseek(fin, coordsr[i]*cols*bands*sizeof(float), SEEK_SET);
    fread(buffer2, sizeof(float), cols*bands, fin);
    for (j=0; j<bands; j++){
        ssignature[i][j]=buffer2[j*cols+coordsc[i]];
        }
    }

fprintf(report, "\nEndmember Candidates:");
for (i=0; i<(nendmembers-1); i++){
    fprintf(report, "\n%d: %10.5lf row=%d, col=%d :", i+1, hmin[i],
coordsr[i], coordsc[i]);
    for(j=0; j<bands; j++){
        fprintf(report, " %f", ssignature[i][j]);
        }
    }
for (i=(nendmembers-1); i<2*(nendmembers-1); i++){
    fprintf(report, "\n%d: %10.5lf row=%d, col=%d :", i+1, hmax[i-
(nendmembers-1)], coordsr[i], coordsc[i]);
    for(j=0; j<bands; j++){
        fprintf(report, " %f", ssignature[i][j]);
        }
    }

/* SAD Calculation for Every Endmember */

for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        for (k=0; k<bands; k++){
            if (i!=j){
                pro[i][j]+=ssignature[i][k]*ssignature[j][k];
                sqr[i][j]+=ssignature[j][k]*ssignature[j][k];
                }
            }
        }
    for (l=0; l<bands; l++){
        sum[i]+=ssignature[i][l]*ssignature[i][l];
        }
    }

for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        if (i!=j){

sad[i][j]=acos((pro[i][j])/((sqrt(sum[i]))*(sqrt(sqr[i][j]))));
            }
        }
    }
fprintf(report, "\nSADs:\n");
for (i=0; i<2*(nendmembers-1); i++){
    fprintf(report, "%d: ", i+1);
    for (j=0; j<2*(nendmembers-1); j++){
        fprintf(report, " %10.5lf", sad[i][j]);
        }
    fprintf(report, "\n");
    }
```

```
/* SAD Adding */

for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        if (i!=j){
            ssad[i]+=sad[i][j];
            }
        }
    }

fprintf(report, "\nsSADs:\n");
for (i=0; i<2*(nendmembers-1); i++){
    fprintf(report, "ssad[%d] = %15.5lf\n", i, ssad[i]);
    }

/* SAD Normalization */

ssadmax=0.;
ssadmin=5000.;
for (i=0; i<2*(nendmembers-1); i++){
    if (ssadmax<ssad[i]){
        ssadmax=ssad[i];
        }
    if (ssadmin>ssad[i]){
        ssadmin=ssad[i];
        }
    }

for (i=0; i<2*(nendmembers-1); i++){
    ssad[i]= (ssad[i]-ssadmin)/(ssadmax-ssadmin);
    }

fprintf(report, "\nNormalized sSADs:\n");
for (i=0; i<2*(nendmembers-1); i++){
    fprintf(report, "ssad[%d] = %15.5lf\n", i, ssad[i]);
    }

/* SAD Differencing */

for (i=0; i<2*(nendmembers-1); i++){
    for (j=i+1; j<2*(nendmembers-1); j++){
        dif[i][j]=fabs(ssad[i]-ssad[j]);
        }
    }

fprintf(report, " \nDifT:\n");
for (i=0; i<2*(nendmembers-1); i++){
    for (j=0; j<2*(nendmembers-1); j++){
        fprintf(report, "dif[%d][%d]= %10.5lf     ", j, i,
dif[j][i]);
        }
    fprintf(report, " \n");
    }

/* Endmember Selection */

fflush(NULL);
threshold=0.0;
c=1000;
while (threshold<0.1 && c>nendmembers){
```

```
    c=0;
    threshold= threshold+0.00001;
    for (i=0; i<2*(nendmembers-1); i++){
        endm[i]=0;
        for (j=i+1; j<2*(nendmembers-1); j++){
            if (dif[i][j]<threshold){
                tempint[i]++;
                }
            }
        if (tempint[i]==0){
            c++;
            endm[i]=1;
            }
        }
    }

printf("Extracted %d Endmembers\n", c);
k=0;
fprintf(report, "\nEndmembers :\n");
for (i=0; i<2*(nendmembers-1); i++){
    if (endm[i]!=0){
        fprintf(report, "\nEndmember %d: at row %d, col %d:\t", i+1,
coordsr[i]+1, coordsc[i]+1);
        for (j=0; j<bands; j++){
            fprintf(report, "\t%5.5lf", ssignature[i][j]);
            a[j][k]=ssignature[i][j];
            }
        k++;
        }
    }

fprintf(report2, "Endmembers :\n");
for (i=0; i<nendmembers; i++){
    for (j=0; j<bands; j++){
        fprintf(report2, "%f ", a[j][i]);
        }
    fprintf(report2, "\n");
    }

/* SEE Closing */

if (fclose(fpca4)!=0){
   puts("Error in fpca4 closing");
   }

for (i=0; i<(2*(nendmembers-1)); i++){
    free(dif[i]);
    free(sad[i]);
    free(sqr[i]);
    free(pro[i]);
    free(ssignature[i]);
    }
free(tempint);
free(endm);
free(dif);
free(sad);
free(ssad);
free(buffer2);
free(buffer);
free(sqr);
free(pro);
```

```
free(sum);
free(ssignature);
free(coordsc);
free(coordsr);
free(hmax);
free(hmin);

return (0);
}

int Unmixing(FILE* fin, FILE* report, int rows, int cols, int bands,
int nendmembers, char* enddir, double **a, int nvalue){

FILE     *fend;
int      i, j, k, l, m;
float    *buffer, *buffer2, *b, c;
double   **N,**ata, **ata_inv, tmpd, *atal, *ata_invl;

/* Memory Allocation */

if ( (buffer=malloc(bands*cols*sizeof(float))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (buffer2=malloc(nendmembers*cols*sizeof(float))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (b=malloc(bands*sizeof(float))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (ata=calloc(nendmembers,sizeof(double*))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (ata_inv=calloc(nendmembers,sizeof(double*))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (atal=calloc(nendmembers*nendmembers,sizeof(double))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (ata_invl=calloc(nendmembers*nendmembers,sizeof(double))) ==
NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (N=calloc(nendmembers,sizeof(double*))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }
```

```
for (i=0; i<nendmembers; i++){
    if ( (ata[i]=calloc(nendmembers,sizeof(double))) == NULL ){
        puts("Error in (buffer) Memory Allocation.");
        exit(-1);
        }

    if ( (ata_inv[i]=calloc(nendmembers,sizeof(double))) == NULL ){
        puts("Error in (buffer) Memory Allocation.");
        exit(-1);
        }

    if ( (N[i]=calloc(bands,sizeof(double))) == NULL ){
        puts("Error in (buffer) Memory Allocation.");
        exit(-1);
        }
    }

fprintf(report, "\n\n %s\n", enddir);

if ( (fend=fopen(enddir,"wb+")) == NULL){
    printf("\nCannot create endmembers file.\n");
    exit (-1);
    }

for (i=0; i<bands; i++){
    for (j=0; j<nendmembers; j++){
        fprintf(report, " %f" ,(a[i][j]));
        }
    fprintf(report, "\n");
    }

/* aTa Calculation */

for (i=0; i<nendmembers ; i++){
    for (j=0; j<nendmembers ; j++){
        for (k=0; k<bands; k++){
            ata[i][j]+=a[k][i]*a[k][j];
            }
        }
    }

fprintf(report, "\nata\n");
for (i=0; i<nendmembers; i++){
    for (j=0; j<nendmembers; j++){
        fprintf(report, " %f" , ata[i][j]);
        }
    fprintf(report, "\n");
    }

/* aTa Inversion */

for (i=0; i< nendmembers; i++){
    for (j=0; j<nendmembers; j++){
        atal[i*nendmembers+j]=ata[i][j];
        }
    }

Matrix_inversion(atal,ata_invl,nendmembers);

for (i=0; i< nendmembers; i++){
    for (j=0; j<nendmembers; j++){
```

```
            ata_inv[i][j]=ata_invl[i*nendmembers+j];
            }
        }

    fprintf(report, "\nata_inv\n");
    for (i=0; i<nendmembers; i++){
        for (j=0; j<nendmembers; j++){
            fprintf(report, " %lf" , ata_inv[i][j]);
            }
        fprintf(report, "\n");
        }
    fflush(NULL);

    fprintf(report, "\n");
    fprintf(report, "\n");

    /* N Matrix Calculation */

    for (i=0; i<nendmembers; i++){
        for (j=0; j<bands; j++){
            tmpd=0.0;
            for (k=0; k<nendmembers; k++){
                tmpd+=ata_inv[i][k]*a[j][k];
                }
            N[i][j]=tmpd;
            }
        }

    fprintf(report, "\nN :\n");
    for(i=0; i<nendmembers; i++){
        for(j=0; j<bands; j++){
            fprintf(report, " %lf", N[i][j]);
            }
        fprintf(report, "\n");
        }

    /* Abundance Fraction Calculation */

    fprintf(report, " \nimage: \n");
    rewind(fin);
    for (i=0; i<rows; i++){
        fread(buffer, sizeof(float), cols*bands, fin);
        printf("Unmixing at row %d of %d\r", i+1, rows);
        for (j=0; j<cols; j++){
            for (k=0; k<bands; k++){
                b[k]=buffer[k*cols+j];
                }
            for (l=0; l<nendmembers; l++){
                c=0.0;
                if (b[0]==nvalue){
                    buffer2[l*cols+j]=nvalue;
                    }
                else {
                    for (m=0; m<bands; m++){
                        c+=N[l][m]*b[m];
                        }
                    buffer2[l*cols+j]=c;
                    }
                }
            }
        fwrite(buffer2, sizeof(float), nendmembers*cols, fend);
```

```
    for (j=0; j<cols*nendmembers; j=j+30){
        fprintf(report, " %f", buffer2[j]);
        }
    }

/* Unmixing Closing */

for (i=0; i<nendmembers; i++){
    free(N[i]);
    free(ata_inv[i]);
    free(ata[i]);
    }
free(N);
free(ata_invl);
free(atal);
free(ata_inv);
free(ata);
free(b);
free(buffer2);
free(buffer);

return (0);
}

int NND(FILE* fin, FILE* fnoise, int rows, int cols, int bands, int
nvalue){

int     i, j, k;
float   *buffer, *buffer2, *noise1, dif[2];

/* Memory Allocation */

if ((buffer2=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((buffer=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((noise1=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

/* Noise Estimation of the first row */

for (i=0; i<bands*cols; i++){
    noise1[i]=0.0;
    }
fwrite(noise1 , sizeof(float), cols*bands, fnoise);

// Noise Estimation of the rest

rewind(fin);
fread(buffer, sizeof(float), cols*bands,fin);

for (i=1; i<rows; i++){
    fread(buffer2, sizeof(float), cols*bands, fin);
```

```
    for (j=0; j<bands; j++){
        for (k=1; k<cols-1; k++){
            if (buffer[j*cols+k]== nvalue){
                noise1[j*cols+k]=nvalue;
                }
            else if (buffer[j*cols+k]!=nvalue){
                dif[0]=buffer2[j*cols+k]-buffer2[j*cols+k+1];
                dif[1]=buffer2[j*cols+k]-buffer[j*cols+k];
                noise1[j*cols+k]=(dif[0]+dif[1])/2;
                }
            }
        }
    fwrite(noise1, sizeof(float), cols*bands, fnoise);
    memcpy(buffer, buffer2, cols*bands*sizeof(float));
    }

/* NND Closing */

free(noise1);
free(buffer);
free(buffer2);

return (0);
}

int MRTBM(FILE* fin, FILE* fnoise, FILE* report, int rows, int cols,
int bands, int nvalue){

FILE    *fnoise2;
int     i, j, k, l;
float   *noise2, *buffer, *buffer2, *buffer3, **Z, *z;
double  **zTz, *zTzl, *zTzl_inv, **zTz_inv, **R, *b, *c, tstart,
tend, tmp;

/* Memory Allocation */

if ((fnoise2=tmpfile()) == NULL){
    printf("\nCannot open noise2 file.\n");
    exit (-1);
    }

if ((noise2=malloc(cols*rows*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((buffer=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((buffer2=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }

if ((buffer3=malloc(cols*bands*sizeof(float)))==NULL){
    puts("Error in Memory Allocation");
    exit(1);
    }
```

```
if ( (zTz=calloc(bands,sizeof(double*))) == NULL ){
    puts("Error in zTz Memory Allocation.");
    exit(-1);
    }

if ( (zTzl=malloc(bands*bands*sizeof(double))) == NULL ){
     puts("Error in zTzl Memory Allocation.");
     exit(-1);
     }

if ( (zTzl_inv=malloc(bands*bands*sizeof(double))) == NULL ){
     puts("Error in zTzl_inv Memory Allocation.");
     exit(-1);
     }

if ( (zTz_inv=malloc(bands*sizeof(double*))) == NULL ){
     puts("Error in zTz_inv Memory Allocation.");
     exit(-1);
     }

if ( (R=calloc((bands-1),sizeof(double*))) == NULL ){
     puts("Error in R Memory Allocation.");
     exit(-1);
     }

if ( (b=calloc((bands-1),sizeof(double))) == NULL ){
     puts("Error in b Memory Allocation.");
     exit(-1);
     }

if ( (c=calloc(rows*cols,sizeof(double))) == NULL ){
     puts("Error in c Memory Allocation.");
     exit(-1);
     }

if ( (z=calloc(rows*cols,sizeof(float))) == NULL ){
     puts("Error in z Memory Allocation.");
     exit(-1);
     }

if ( (Z=malloc(rows*cols*sizeof(float*))) == NULL ){
     puts("Error in c Memory Allocation.");
     exit(-1);
     }

for (i=0; i<bands; i++){
    if ( (zTz[i]=calloc(bands,sizeof(double))) == NULL ){
       puts("Error in zTz[i] Memory Allocation.");
       exit(-1);
       }
    if ( (zTz_inv[i]=malloc(bands*sizeof(double))) == NULL ){
       puts("Error in zTz_inv[i] Memory Allocation.");
       exit(-1);
       }
    }

for( i=0; i<rows*cols; i++){
    if ( (Z[i]=malloc((bands-1)*sizeof(float))) == NULL ){
        puts("Error in Z[i] Memory Allocation.");
        exit(-1);
        }
```

```
    }
for (i=0; i<bands-1; i++){
    if ( (R[i]=calloc(rows*cols,sizeof(double))) == NULL ){
       puts("Error in R[i] Memory Allocation.");
       exit(-1);
        }
}

tstart=omp_get_wtime();
for (i=0; i<bands; i++){
    printf("Calcul. Noise using Mul. Regression at band %d of %d",
i+1, bands);

/* Z Matrix Formation */

    for (j=0; j<i; j++){
        fseek(fin, j*cols*sizeof(float), SEEK_SET);
        for (k=0; k<rows; k++){
            fread(buffer, sizeof(float), cols, fin);
            for (l=0; l<cols; l++){
                Z[k*cols+l][j]=buffer[l];
                }
            fseek(fin, (bands-1)*cols*sizeof(float), SEEK_CUR);
            }
        }

    for (j=i; j<bands-1; j++){
        fseek(fin, (j+1)*cols*sizeof(float), SEEK_SET);
        for (k=0; k<rows; k++){
            fread(buffer, sizeof(float), cols, fin);
            for (l=0; l<cols; l++){
                Z[k*cols+l][j]=buffer[l];
                }
            fseek(fin, (bands-1)*cols*sizeof(float), SEEK_CUR);
            }
        }

    fseek(fin, i*cols*sizeof(float), SEEK_SET);
    for (j=0; j<rows; j++){
        fread(buffer, sizeof(float), cols, fin);
        for (k=0; k<cols; k++){
            z[j*cols+k]=buffer[k];
            }
        fseek(fin, (bands-1)*cols*sizeof(float), SEEK_CUR);
        }

    for (j=0; j<bands-1; j++){
        for (k=0; k<bands-1; k++){
            zTz[j][k]=0.0;
        }
    }

/* zTz Matrix Calculation */

//# pragma omp parallel for default(none) private(j, k, l, tmp)
shared(rows, cols, bands, zTz, Z)
    for (j=0; j<bands-1; j++){
        for (k=0; k<bands-1; k++){
            tmp=0.0;
            for (l=0; l<rows*cols; l++){
                tmp+=Z[l][k]*Z[l][j];
```

```
                }
            zTz[j][k]=tmp;
            }
        }

/* zTz Matrix Inversion */

    for (j=0; j<bands-1; j++){
        for (k=0; k<bands-1; k++){
            zTzl[j*(bands-1)+k]=zTz[j][k];
            }
        }

    Matrix_inversion(zTzl, zTzl_inv, bands-1);

    for (j=0; j<bands-1; j++){
        for (k=0; k<bands-1; k++){
            zTz_inv[j][k]=zTzl_inv[j*(bands-1)+k];
            }
        }

/* R Matrix Calculation */

    for (j=0; j<bands-1; j++){
        for (k=0; k<rows*cols; k++){
            tmp=0.0;
            for (l=0; l<bands-1; l++){
                tmp+=zTz_inv[j][l]*Z[k][l];
                }
            R[j][k]=tmp;
            }
        }

/* Regression Factors Calculation */

    for (j=0; j<bands-1; j++){
        tmp=0.0;
        for (k=0; k<rows*cols; k++){
            tmp+=R[j][k]*z[k];
            }
        b[j]=tmp;
        }

/* c Matrix Calculation */

    for (j=0; j<rows*cols; j++){
        tmp=0.0;
        for (k=0; k<bands-1; k++){
            tmp+=Z[j][k]*b[k];
            }
        c[j]=tmp;
        }

/* Noise Calculation */

    for (j=0; j<rows*cols; j++){
        if (z[j]!=nvalue){
            noise2[j]=(float)(z[j]-c[j]);
            }
        else {
            noise2[j]=nvalue;
```

```
            }
        }

fprintf(report, "\nnoise=\n");
for (j=0; j<400; j++){
    fprintf(report, "%f ", noise2[j]);
    }
fprintf(report, "\n");

/* Noise2 Image Writing */

    if (fseek(fnoise2, i*cols*sizeof(float), SEEK_SET) != 0){
       puts ("Bad File Rewind");
       }

    for (j=0; j<rows; j++){
        fwrite(noise2+j*cols, sizeof(float), cols, fnoise2);
        if (fseek ( fnoise2, cols*(bands-1)*sizeof(float), SEEK_CUR)
!= 0){
           puts( "Bad File Control");
           }
        fflush(NULL);
        }

tend=omp_get_wtime();
printf(" Duration: %2.2lf min\r", (tend-tstart)/60.0);
    }

rewind(fin);
rewind(fnoise2);
for(i=0; i<rows; i++){
    fread(buffer, sizeof(float), cols*bands, fin);
    fread(buffer2, sizeof(float), cols*bands, fnoise2);
    for(j=0; j<cols*bands; j++){
        buffer3[j]=buffer2[j]-buffer[j];
        }
    fwrite(buffer3, sizeof(float), cols*bands, fnoise);
    }

printf("\n");
/* MRTBM Closing */

for(i=0; i<bands-1; i++){
    free(R[i]);
    }
for(i=0; i<rows*cols; i++){
    free(Z[i]);
    }
for(i=0; i<bands; i++){
    free(zTz_inv[i]);
    free(zTz[i]);
    }
free(Z);
free(z);
free(c);
free(b);
free(R);
free(zTz_inv);
free(zTzl_inv);
free(zTzl);
free(zTz);
```

```
free(buffer);
free(noise2);

return(0);
}

int PCA(FILE* fin, FILE* report, int rows, int cols, int bands, int
noc, FILE * fpca, int nvalue, int npixels){

int             i, j, k, l, nrot;
double          *sum, **sumv, **cov, *mean, *eigval, **eigvec,
tstart, tend;
float           *pc, *buffer;

if (noc==-1){
    noc=bands;
    }

/* Memory Allocation */

if ( (buffer=malloc(bands*cols*sizeof(float))) == NULL ){
    puts("Error in (buffer) Memory Allocation.");
    exit(-1);
    }

if ( (sum=calloc(bands,sizeof(double))) == NULL ){
    puts("Error in (sum) Memory Allocation.");
    exit(-1);
    }

if ( (sumv=calloc(bands,sizeof(double*))) == NULL ){
    puts("Error in (sumv) Memory Allocation.");
    exit(-1);
    }

if ( (mean=calloc(bands,sizeof(double))) == NULL ){
    puts("Error in (mean) Memory Allocation.");
    exit(-1);
    }

if ( (cov=malloc(bands*sizeof(double*))) == NULL ){
    puts("Error in (cov) Memory Allocation.");
    exit(-1);
    }

if ( (eigval=malloc(bands*sizeof(double))) == NULL ){
    puts("Error in (eigval) Memory Allocation.");
    exit(-1);
    }

if ( (eigvec=malloc(bands*sizeof(double*))) == NULL ){
    puts("Error in (eigvec) Memory Allocation.");
    exit(-1);
    }

if ( (pc=calloc(noc*cols,sizeof(float))) == NULL ){
    puts("Error in (pc) Memory Allocation.");
    exit(-1);
    }

for (i=0; i<bands; i++){
```

```
    if ((sumv[i]=calloc(bands,sizeof(double))) == NULL){
        puts("Error in (sumv[i]) Memory Allocation.");
        exit(-1);
        }
    if ((cov[i]=malloc(bands*sizeof(double))) == NULL){
        puts("Error in (cov[i]) Memory Allocation.");
        exit(-1);
        }
    if ( (eigvec[i]=malloc(bands*sizeof(double))) == NULL ){
        puts("Error in (eigvec[i]) Memory Allocation.");
        exit(-1);
        }
    }

rewind(fin);

/* Sum Calculation for Every Band*/

for (i=0; i<rows; i++){
    fread (buffer, sizeof(float), cols*bands, fin);
    for (j=0; j<bands; j++){
        for (k=0; k<cols; k++){
            if (buffer[j*cols+k]!=nvalue){
                sum[j]+=buffer[(j*cols)+k];
                }
            }
        }
    }

fprintf(report,"\nPCA :\n");
fprintf(report,"\nSum :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, sum[i]);
    }

/* Mean Calculation for Every Band */

for (i=0; i<bands; i++){
    mean[i]=sum[i]/(rows*cols-npixels);
    }

fprintf(report,"\nMean :\n");
for (i=0; i<bands; i++){
    fprintf(report,"Band %d : %10.5lf\n", i+1, mean[i]);
    }

rewind(fin);
/* sumv Calculation for Every Band*/

tstart=omp_get_wtime();
# pragma omp parallel default(none) private(i, j, k, l)
shared(buffer, fin, mean, sumv, rows, cols, bands, nvalue)
{
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread (buffer, sizeof(float), cols*bands, fin);
    printf("Calculating Statistics at row %d of %d\r", i+1, rows);
    }
    # pragma omp for schedule(dynamic)
    for (j=0; j<bands; j++){
```

```
        for (k=0; k<bands; k++){
            for (l=0; l<cols; l++){
                if (buffer[j*cols+l]!=nvalue){
                    sumv[j][k]+=(buffer[(j*cols)+l]-
mean[j])*(buffer[(k*cols)+l]-mean[k]);
                }
            }
        }
    }
}

tend=omp_get_wtime();
printf("\nDuration of Statistics Parallel region : %2.1lf sec\n",
tend-tstart);

/* Variance-Covariance Matrix Calculation */

for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        cov[i][j]=sumv[i][j]/(rows*cols-npixels);
        }
    }

fprintf(report,"\nVariance-Covariance Matrix:\n");
for (i=0; i<bands; i++){
    fprintf(report, "Band %d", i+1);
    for (j=0; j<bands; j++){
        fprintf(report," %15.5lf", cov[i][j]);
        }
    fprintf(report,"\n");
    }
fflush(NULL);
/* Eigenvalues and Eigenvectors Calculation and Sorting */

jacobi(cov, bands, eigval, eigvec, &nrot);
eigsort(eigval, eigvec, bands);

//fprintf(report,"\nNumber of Jacobi rotations : %d\n", nrot);

fprintf(report,"\nEigenvalues :\n");
for (i=0; i<bands; i++){
    fprintf(report," %10.5lf\n", eigval[i]);
    }

fprintf(report,"\nEigenvectors :\n");
for (i=0; i<bands; i++){
    for (j=0; j<bands; j++){
        fprintf(report," %10.5lf", eigvec[i][j]);
        }
    fprintf(report,"\n");
    }


/* PC Calculation */

rewind(fin);
tstart=omp_get_wtime();
# pragma omp parallel default (none) private(i, j, k, l) shared(fin,
buffer, rows, cols, bands, noc, pc, eigvec, fpca, mean, nvalue)
{
```

```
for (i=0; i<rows; i++){
    # pragma omp single
    {
    fread(buffer, sizeof(float), cols*bands, fin);
    printf("Calculating Pr. Components at row %d of %d\r", i+1,
rows);
    }
    # pragma omp for schedule (dynamic)
    for (j=0; j<noc; j++){
        for (k=0; k<cols; k++){
            pc[j*cols+k]=0.0;
            for(l=0; l<bands; l++){
                if (buffer[l*cols+k]!=nvalue){
                    pc[j*cols+k]+=1.*(buffer[(l*cols)+k]-
mean[j])*(eigvec[l][j]);
                    }
                else {
                    pc[j*cols+k]=nvalue;
                    }
                }
            }
        }
    # pragma omp single
    {
    fwrite(pc, sizeof(float), noc*cols, fpca);
    }
    }
}

tend=omp_get_wtime();
printf("\nDuration of PCA Parallel region : %2.1lf sec\n", tend-
tstart);

/* Closing */

fflush(NULL);
for (i=0; i<bands; i++){
    free(sumv[i]);
    free(cov[i]);
    free(eigvec[i]);
    }
free(pc);
free(eigvec);
free(eigval);
free(cov);
free(mean);
free(sumv);
free(sum);
free(buffer);

return(0);
}
/* Source: http://users.ntua.gr/chiossif/ */
int Matrix_inversion(FP_TYPE *ata,FP_TYPE *ata_inv,int n) {
    FP_TYPE *tmpa, *b;
    int *l;
    int r_value;
    register int i, j, k;

    tmpa=malloc(n*n*sizeof(FP_TYPE));
    b=malloc(n*sizeof(FP_TYPE));
```

```
    l=malloc(n*sizeof(int));
    r_value=1;
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            for (k=0; k<n; k++)
                tmpa[j*n+k]=ata[j*n+k];
            b[j]=0.0;
        }
        b[i]=1.0;
        r_value*=Cholesky_decomposition(tmpa, b, n);
        for (j=0; j<n; j++)
            ata_inv[j*n+i]=b[j];
    }
    free(l);
    free(b);
    free(tmpa);
    return r_value;
}
/* Source: http://users.ntua.gr/chiossif/ */
int Cholesky_decomposition(FP_TYPE *a, FP_TYPE *b, int n) {
    FP_TYPE s;
    register int i, j, k;

    for (i=0; i<n; i++) {
        s=0.0;
        if (i)
            for (j=0; j<i; j++)
                s+=mySQR(a[i*n+j]);
        a[i*n+i]=mySQRT(a[i*n+i]-s);
        if (myABS(a[i*n+i])<EPSILON)
            return 0;
        if (i<n-1)
            for (j=i+1; j<n; j++) {
                s=0.0;
                if (i)
                    for (k=0; k<i; k++)
                        s+=a[i*n+k]*a[j*n+k];
                a[j*n+i]=(a[j*n+i]-s)/a[i*n+i];
            }
    }
    b[0]/=a[0*n+0];
    for (k=1; k<n; k++) {
        s=0.0;
        for (j=0; j<k; j++)
            s+=a[k*n+j]*b[j];
        b[k]=(b[k]-s)/a[k*n+k];
    }
    b[n-1]/=a[(n-1)*n+n-1];
    for (i=0; i<n-1; i++) {
        s=0.0;
        for (j=n-i-1; j<n; j++)
            s+=a[j*n+n-i-2]*b[j];
        b[n-i-2]=(b[n-i-2]-s)/a[(n-i-2)*n+n-i-2];
    }
    return 1;
}

FP_TYPE mySQRT(FP_TYPE x) {
    FP_TYPE px,a;

    a=px=x;
```

```
    x/=2.0;
    while (myABS(x-px)>EPSILON) {
        px=x;
        x=(px+a/px)/2;
    }
    return x;
}

/* Jacobi function computes all eigenvalues and eigenvectors of a
real symmetric matrix
a[1..n][1..n]. On output, elements of a above the diagonal are
destroyed.
d[1..n] returns the eigenvalues of a.   v[1..n][1..n] is a matrix
whose
columns contain, on output, the normalized eigenvectors of a.
nrot returns the number of Jacobi rotations that were required.
Source: http://users.ntua.gr/chiossif/
 */

void jacobi(double **a, int n, double *d, double **v, int *nrot){

int j,iq,ip,i;
double tresh,theta,tau,t,sm,s,h,g,c,*b,*z;

if ((b=(double *)malloc(n*sizeof(double)))==NULL){
    puts("Jacobi reports: Error in Memory Allocation");
    exit(1);
    }

if ((z=(double *)malloc(n*sizeof(double)))==NULL){
    puts("Jacobi reports: Error in Memory Allocation");
    exit(1);
    }
for(ip=0;ip<n;ip++){
    for(iq=0;iq<n;iq++)
        v[ip][iq]=0.0;
    v[ip][ip]=1.0;
    }

for(ip=0;ip<n;ip++){
    d[ip]=b[ip]=a[ip][ip];
    z[ip]=0.0;
    }

*nrot=0;

for(i=1;i<=50;i++){
    sm=0.0;
    for(ip=0;ip<n-1;ip++){
        for(iq=ip+1;iq<n;iq++)
            sm+=fabs(a[ip][iq]);
        }
    if(fabs(sm)<1.0e-14){
        free(z);
        free(b);
        return;
        }
    if(i<4)
        tresh=0.2*sm/(n*n);
    else
        tresh=0.0;
```

```
    for(ip=0;ip<n-1;ip++){
        for(iq=ip+1;iq<n;iq++){
            g=100.0*fabs(a[ip][iq]);
            if
(i>4&&(fabs(d[ip])+g)==fabs(d[ip])&&(fabs(d[iq])+g)==fabs(d[iq]))
                a[ip][iq]=0.0;
            else if(fabs(a[ip][iq])>tresh){
                h=d[iq]-d[ip];
                if((fabs(h)+g)==fabs(h))
                    t=(a[ip][iq])/h;
                else{
                    theta=0.5*h/(a[ip][iq]);
                    t=1.0/(fabs(theta)+sqrt(1.0+theta*theta));
                    if (theta<0.0)
                        t=-t;
                    }
                c=1.0/sqrt(1+t*t);
                s=t*c;
                tau=s/(1.0+c);
                h=t*a[ip][iq];
                z[ip]-=h;
                z[iq]+=h;
                d[ip]-=h;
                d[iq]+=h;
                a[ip][iq]=0.0;
                for(j=0;j<=ip-1;j++){
                    ROTATE(a,j,ip,j,iq)
                    }
                for(j=ip+1;j<=iq-1;j++){
                    ROTATE(a,ip,j,j,iq)
                    }
                for(j=iq+1;j<n;j++){
                    ROTATE(a,ip,j,iq,j)
                    }
                for(j=0;j<n;j++){
                    ROTATE(v,j,ip,j,iq)
                    }
                ++(*nrot);
                }
            }
        }
    for(ip=0;ip<n;ip++){
        b[ip]+=z[ip];
        d[ip]=b[ip];
        z[ip]=0.0;
        }
    }
}

/* Given the eigenvalues d[1..n] and eigenvectors v[1..n][1..n] as
output
from jacobi routine, eigensort function sorts the eigenvalues into
descending order,
and rearranges the columns of v correspondingly. The method is
straight insertion.
Source: http://users.ntua.gr/chiossif/
*/

void eigsort(double d[],double **v,int n){

int k,j,i;
```

```
double p;

for(i=0;i<n-1;i++){
    p=d[k=i];
    for(j=i+1;j<n;j++)
        if(d[j]>=p)
            p=d[k=j];
    if(k!=i){
        d[k]=d[i];
        d[i]=p;
        for(j=0;j<n;j++){
            p=v[j][i];
            v[j][i]=v[j][k];
            v[j][k]=p;
            }
        }
    }
}
```