Τ. Ε. Ι. ΛΑΜΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΑΞΙΝΟΜΗΣΗ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗ ΕΙΚΟΝΩΝ ΜΕ ΧΡΗΣΗ SVM

Χριστίνα Κεφαλιανού

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
ΚΑΡΚΑΝΗΣ ΣΤΑΥΡΟΣ
ΚΑΘΗΓΗΤΗΣ

Λαμία 2014

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση αυτής της πτυχιακής υλοποιήθηκε με την υποστήριξη ενός αριθμού ανθρώπων στους οποίους θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου. Πρώτα από όλους θα ήθελα να ευχαριστήσω τον καθηγητή αλλά και επιτηρητή μου στην πτυχιακή αυτή εργασία, Κο. Σταύρο Καρκάνη, ο οποίος με την καθοδήγηση, τις γνώσεις αλλά και την υποστήριξη του, με βοήθησε να ολοκληρώσω επιτυχώς την εργασία αυτή. Ακόμη, θα ήθελα να ευχαριστήσω τα παιδιά, στην ομάδα της πτυχιακής, για την ευχάριστη συνεργασία και επικοινωνία που είχαμε την τελευταίο χρόνο στα πλαίσια ολοκλήρωσης των πτυχιακών μας εργασιών.

Κεφαλιανού Χριστίνα
ΟΚΤΩΒΡΗΣ 2014

Περιεχόμενα

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία αυτή πραγματεύεται την ταξινόμηση καθώς και την αναγνώριση εικόνων με την χρήση svm(support vector machine).Η αυθεντική (και αρχική) τεχνική του svm εφευρέθηκε από τον Vladimir N. Vapnik ενώ η ισχύουσα μορφή, προτάθηκε από τον Vapnik και την Corinna Cortes το 1993 και δημοσιεύτηκε το 1995.

Αρχικά, στο πρώτο Κεφάλαιο, περιγράφεται αναλυτικά ο ήδη υπάρχων κώδικας που χρησιμοποιήθηκε μαζί με τις διαθέσιμες βάσεις δεδομένων, που δημιουργήθηκαν από επιστήμονες του MIT και δημοσιεύτηκαν στην σελίδα MIT CSAIL.(MIT Computer Science and Artificial Intelligence Laboratory).Επίσης γίνεται μία εισαγωγή και θεωρητική περιγραφή των αλγορίθμων εξαγωγής χαρακτηριστικών που χρησιμοποιήθηκαν από αυτή τη σελίδα για την πραγμάτωση των μεταγενέστερων πειραματισμών καθώς και των αποτελεσμάτων τους.

Στο δεύτερο Κεφάλαιο, γίνεται θεωρητική περιγραφή των τεχνικών που εισήχθησαν στον ήδη υπάρχοντα κώδικα, με σκοπό τη χρήση και άλλων τεχνικών των αλγορίθμων ώστε να εξετάσουμε την απόδοση του προτεινόμενου πλαισίου αναγνώρισης και ταξινόμησης των εικόνων. Οι αλγόριθμοι παρουσιάζονται και περιγράφονται θεωρητικά αλλά και σε επίπεδο κώδικα όπως υλοποιήθηκαν.

Στο τρίτο Κεφάλαιο, έχουμε την συνοπτική παρουσίαση των τελικών αποτελεσμάτων των προτεινόμενων αλγορίθμων, μετά την εκτέλεση του SVM.

Στο τέταρτο Κεφάλαιο γίνεται σύγκριση του συνόλου των αποτελεσμάτων (πειραματικών και μη).

Στο πέμπτο και τελευταίο Κεφάλαιο παρατίθεται οι μελλοντικές επεκτάσεις της εργασίας αυτής.

Η εργασία αυτή εκπονήθηκε με σκοπό την ανάπτυξη αλγορίθμων εξαγωγής χαρακτηριστικών από εικόνες μιας διεθνώς αποδεκτής βάσης δεδομένων ψηφιακών εικόνων και τη σύγκριση των αποτελεσμάτων με αυτά τα οποία επιτυγχάνονταν από τους αλγορίθμους της βιβλιογραφίας με σκοπό την καλύτερη απόδοση των αλγορίθμων ταξινόμησης και αναγνώρισης εικόνων.

Η μεθοδολογία που χρησιμοποιήθηκε εστιάσθηκε στην ανάλυση και κατανόηση των αλγορίθμων που υπήρχαν δημοσιευμένοι στη σελίδα του MIT,ενώ στη συνέχεια έγιναν διάφοροι πειραματισμοί ώστε να βρεθεί ο αλγόριθμος που η απόδοση του πάνω στις υπάρχουσες βάσεις δεδομένων θα είναι το ίδιο σε απόδοση αναγνώρισης ή καλύτερος από αυτή των ήδη υπαρχόντων αλγορίθμων.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

Εγώ, η Χριστίνα Κεφαλιανού του [redacted] δηλώνω υπεύθυνα ότι το περιεχόμενο της εργασίας που περιέχεται σε αυτό το βιβλίο είναι αποκλειστικά δικό μου. Σε όποιο σημείο περιέχονται πληροφορίες από άλλες πηγές, δηλώνω ότι αυτό επισημαίνεται στην εργασία.

Ονοματεπώνυμο: Χριστίνα Κεφαλιανού

Υπογραφή:

ΚΕΦΑΛΑΙΟ 1: ΧΡΗΣΗ ΚΑΙ ΕΠΕΞΗΓΗΣΗ ΥΠΑΡΧΟΝΤΩΝ ΑΛΓΟΡΙΘΜΩΝ

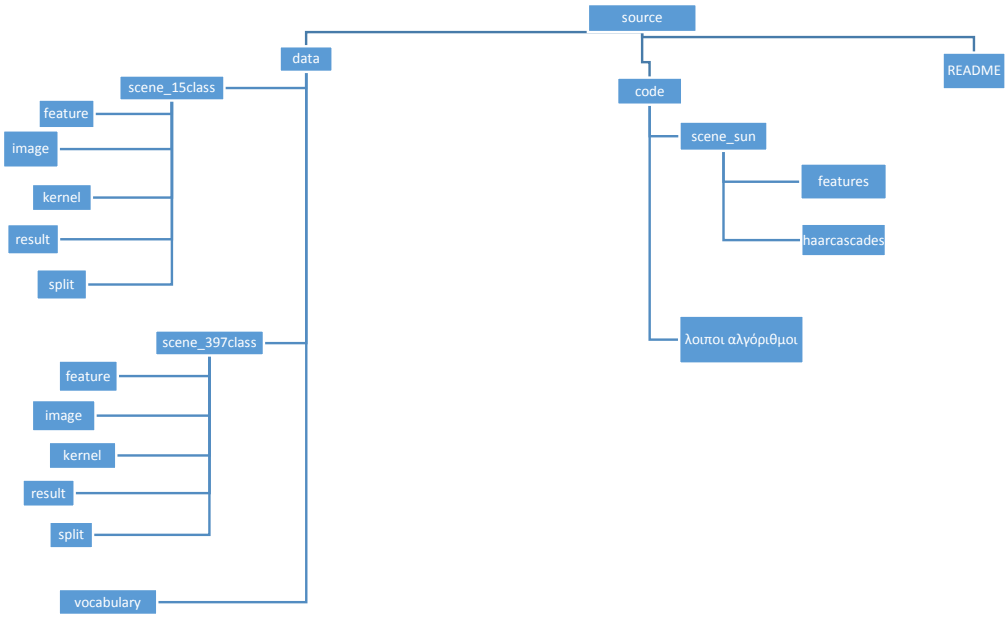
1.1 ΕΙΣΑΓΩΓΗ

1.1.1 ΟΡΓΑΝΩΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Η οργάνωση και αρχειοθέτηση του κώδικα που χρησιμοποιήθηκε από την σελίδα του MIT αποτελείται από τα δεδομένα (φάκελος data) όπου είναι ο φάκελος όπου βρίσκεται η βάση δεδομένων μας, ανάλογα με τον αριθμό των κατηγοριών των εικόνων .Σε αυτή την εργασία έχουμε τον φάκελο scene_15class (θα χρησιμοποιήσουμε δηλαδή μια βάση με 15 κατηγορίες εικόνων). Σε αυτόν τον φάκελο υπάρχουν 15 κατηγορίες εικόνων, καθώς και τα αποτελέσματα (φάκελος result), οι πυρήνες (φάκελος kernel) και τα χαρακτηριστικά (φάκελος feature). Τα 3 τελευταία διαφοροποιούνται κάθε φορά που τρέχει ο κώδικας με διαφορετικά δεδομένα (δηλαδή με διαφορετικούς πυρήνες, χαρακτηριστικά κ.ο.κ). Σε κάθε προσπάθεια εκτέλεσης του κώδικα, ό,τι αρχείο έχει δημιουργηθεί στους φακέλους result και kernel πρέπει να διαγράφεται, αλλιώς δεν είναι επιτυχής η εκτέλεση του κώδικα. Επίσης είναι απαραίτητη η παρουσία αρχείου με την ονομασία split, το οποίο είναι ένας πίνακας 1x10 που διαμοιράζει τη διαδικασία της εκπαίδευσης του συστήματος (SVM στην περίπτωσή μας) σε 10 στάδια, με κάθε στάδιο να περιέχει 100 παραδείγματα εικόνων από κάθε κατηγορία καθώς και το σύνολο των εικόνων για την δοκιμή του SVM σε κάθε στάδιο εκπαίδευσης .Στον φάκελο με ονομασία code περιέχονται οι αλγόριθμοι εξαγωγής χαρακτηριστικών, κώδικες για την εφαρμογή τους σε διαφορετικούς πυρήνες των SVM καθώς και πρόσθετες συναρτήσεις που χρησιμοποιούνται από τις βασικές συναρτήσεις, βοηθητικά, για να επιτελέσουν τον σκοπό τους.

Τέλος, στο φάκελο scene_sun αποθηκεύονται βασικοί κώδικες που χρησιμοποιήσαμε εμείς για την ολοκλήρωση της εργασίας αυτής. Ο φάκελος αυτός περιέχει έναν υποφάκελο με ονομασία features στον οποίο διατηρούνται οι αλγόριθμοι εξαγωγής των χαρακτηριστικών (εκεί διατηρούνται οι πειραματικοί αλγόριθμοι που υλοποιήθηκαν),έναν φάκελο haarcascades που εξ αρχής ήταν κενός, και παρέμεινε κενός καθόλη την διάρκεια εκτέλεσης των αλγορίθμων (δεν κράτησε δεδομένα ούτε περιείχε δεδομένα).Στο Σχήμα 1α φαίνεται σε ιεραρχική αναπαράσταση η δομή των φακέλων.

κώδικα



Σχήμα 1α : Δομή φακέλων της υλοποίησης.

1.1.2 ΟΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Η βάση δεδομένων που χρησιμοποιήθηκε από την σελίδα του MIT, απαρτίζονταν από εικόνες διαχωρισμένες σε 15 κλάσεις (κατηγορίες). Οι εικόνες αυτές είναι ψηφιακές εικόνες των 8bpp (8 bit per pixel), δηλαδή, κάθε pixel αποτελείται από 8 bit (1 byte). Επομένως, το χρωματικό εύρος κάθε pixel της κάθε εικόνας ήταν [0,255], δηλαδή grayscale εικόνες (Εικόνες στην κλίματα του γκρι).Ακόμη, στην επίσημη σελίδα του MIT στην οποία έγινε η δημοσίευση του κώδικα, βρήκαμε άλλες δύο βάσεις δεδομένων. Μία με 8 κλάσεις εικόνων , παρόμοιας δομής με αυτής των 15 κλάσεων, και μία 397 κλάσεων εικόνων , γνωστή και ως SUN397 database η οποία περιείχε 397 κλάσεις εικόνων, χωρισμένες αλφαβητικά. Φυσικά, υπάρχει πάντα η δυνατότητα ανάπτυξης μίας ανεξάρτητης βάσης δεδομένων εικόνων, από τον εκάστοτε χρήστη και χρήσης της για την εκπαίδευση του SVM.

1.1.3 ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟΝ ΚΩΔΙΚΑ

Οι αλγόριθμοι δημιουργίας και προσδιορισμού των χαρακτηριστικών των εικόνων βρίσκονται στον φάκελο codes.Για την εκπόνηση της εργασίας αυτής χρησιμοποιήθηκαν και τροποποιήθηκαν οι αλγόριθμοι που βρίσκονται στον υποφάκελο scene_sun. Η υλοποίηση του συνόλου των αλγορίθμων σε βήματα έχει ως εξής :

1. Αρχικά, πρέπει να γίνει ο υπολογισμός κάποιου χαρακτηριστικού στις εικόνες με χρήση του κώδικα compute_feature.m (βρίσκεται στον φάκελο scene_sun) ο οποίος με τις κατάλληλες τροποποιήσεις, χρησιμοποιεί κάποιον από τους αλγορίθμους και εξαγάγει τα χαρακτηριστικά τα οποία αποθηκεύονται στον υποφάκελο features της βάσης δεδομένων (φάκελος data) που χρησιμοποιήθηκε.
2. Στη συνέχεια, γίνεται εκτέλεση του αλγορίθμου ταξινόμησης με βάση τα χαρακτηριστικά που προσδιορίστηκαν στο προηγούμενο βήμα με χρήση του κώδικα run_kernel_svm.m στο οποίο γίνεται η επιλογή του πυρήνα και του χαρακτηριστικού που θα τρέξει πάνω σε αυτόν τον πυρήνα.
3. Τα τελικά αποτελέσματα αποθηκεύονται στον υποφάκελο results (ο οποίος βρίσκεται στον φάκελο data),της εκάστοτε βάσης που έχει χρησιμοποιηθεί καθώς και στον εκάστοτε φάκελο kernel.
4. Στο τέλος κάθε εκτέλεσης εμφανίζεται παράθυρο αποτελεσμάτων στην περιοχή εντολών του Matlab και ένα συγκεντρωτικό διάγραμμα, στο οποίο εμφανίζονται τα αποτελέσματα για κάθε ένα από τα 10 τμήματα του διαχωρισμού της εκπαίδευσης.

1.2 ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ

1.2.1 ΕΙΚΟΝΑ:ΟΡΙΣΜΟΣ: ΤΙ ΕΙΝΑΙ Η ΕΙΚΟΝΑ

Ως εικόνα ορίζεται η αποτύπωση ή απομνημόνευση της εμφάνισης ενός ή περισσοτέρων αντικειμένων σε ένα χαρτί, στην ανθρώπινη μνήμη ,στην οθόνη ενός υπολογιστή ή φωτογραφικής μηχανής, συνηθέστερα σε δύο ή τρεις διαστάσεις. Όμως με την λέξη εικόνα δεν νοείται μόνο η απεικόνιση ενός φυσικού ή μη περιβάλλοντος, άλλα είναι ένα μέσο με το οποίο μπορούμε να αποτυπώσουμε και να συγκρατήσουμε πληροφορίες διαφόρων ειδών.[1] Έτσι, έχουμε εικόνες που απεικονίζουν την φύση αλλά και εικόνες ιατρικών δεδομένων (τομογραφίες, ακτινογραφίες),εικόνες τηλεπισκόπησης κ.ο.κ Όλες αυτές οι πληροφορίες μπορούν να ψηφιοποιηθούν και να επεξεργασθούν ως ψηφιακές εικόνες. Ψηφιακή Εικόνα λοιπόν είναι η κωδικοποίηση μίας αναλογικής εικόνας με σκοπό την εκτύπωση, αποθήκευση ,βελτιστοποίηση, ανάλυση και μετάδοση της. Είναι ένας πίνακας δύο ή τριών διαστάσεων, ο οποίος μεταφέρει πληροφορίες όπως στα παραπάνω παραδείγματα, με την διαφορά ότι οι πληροφορίες αυτές είναι επεξεργάσιμες , σε αντίθεση με τις αναλογικές εικόνες. Ο επιστημονικός κλάδος που ασχολείται με τις τεχνικές αυτές ονομάζεται Ψηφιακή Επεξεργασία Εικόνας (ΨΕΕ). Ακόμη μία ψηφιακή εικόνα μπορεί να είναι έγχρωμη (RGB,HSV κ.ο.κ), στην κλίμακα του γκρι (grayscale), δύο αποχρώσεων (bitmap) και αντίστοιχα ορίζεται και το βάθος των εικονοστοιχείων (pixel) της.[2]

Ως εικονοστοιχείο (pixel = picture element), είναι ένα «σημείο» μιας εικόνας που εμφανίζεται στην οθόνη ενός υπολογιστικού συστήματος, δηλαδή για το υπολογιστικό σύστημα, ένα δείγμα πληροφορίας. Μία εικόνα στον υπολογιστή εμφανίζεται ως ένα «ψηφιδωτό» και το εικονοστοιχείο αποτελεί μία ψηφίδα από αυτό το ψηφιδωτό. Σε μια οθόνη υπολογιστή, μια εικόνα αναπαρίσταται με υποδιαίρεση της οθόνης σε ένα δυσδιάστατο πίνακα, του οποίου το εκάστοτε «κελί», αποτελεί και ένα εικονοστοιχείο. Ο αριθμός αυτών των υποδιαιρέσεων είναι αρκετά μεγάλος και για το λόγο αυτό η λεπτομέρεια αυτή δεν διακρίνεται με το ανθρώπινο μάτι. [3] (Σχήμα 1β)

Σχήμα 1β Παράδειγμα εικονοστοιχείων εικόνας

1.2.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

1.2.2.1 ΟΡΙΣΜΟΣ: ΤΙ ΕΙΝΑΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Ένα σύστημα τεχνητής νοημοσύνης το οποίο έχει ως σκοπό την αναγνώριση και την λήψη αποφάσεων, στηρίζεται συνήθως στην αναγνώριση προτύπων. Όταν δε το σύστημα αυτό σχετίζεται με εφαρμογές Ψηφιακής Επεξεργασίας και Ανάλυσης Εικόνων, τα πρότυπα αυτά καλούνται χαρακτηριστικά (features). Γενικά, ως χαρακτηριστικό, μπορεί να θεωρηθεί ένα οποιοδήποτε μετρήσιμο μέγεθος που εξάγεται από μία εικόνα. Η χρήση τους είναι απαραίτητη για την περιγραφή αντικειμένων που περιέχονται σε εικόνες, και η επιλογή τους είναι ανάλογη των απαιτήσεων της εκάστοτε εφαρμογής. Τα χαρακτηριστικά που επιλέγονται πρέπει να διαχωρίζουν επαρκώς μονοσήμαντα τα αντικείμενα ή τις περιοχές μιας εικόνας, διότι στη περιγραφή αντικειμένων με χαρακτηριστικά, η αναγνώριση των αντικειμένων αυτών, ισοδυναμεί με την μέτρηση της ομοιότητας μεταξύ των χαρακτηριστικών τους.

Η κατηγοριοποίηση των χαρακτηριστικών μπορεί να γίνει με διάφορους τρόπους. Μπορεί να κατηγοριοποιηθούν σε

- μετρήσιμα,
- συμβολικά,
- ΝΑΙ ή ΟΧΙ,
- δυαδικά,
- σαφή ή ασαφή.

Ακόμη, μπορούν να διαχωρισθούν με βάση το πεδίο ορισμού τους ως:

- Χαρακτηριστικά χώρου,
- Χαρακτηριστικά από μετασχηματισμό

Τα χαρακτηριστικά χώρου μπορούν να διακριθούν σε υποκατηγορίες όπως:

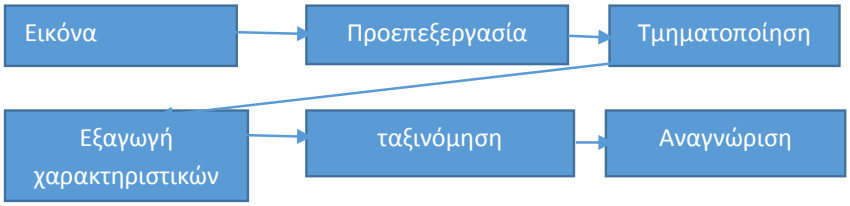
- Γεωμετρικά χαρακτηριστικά,
- Στατιστικά χαρακτηριστικά (π.χ. ροπές)
- Χαρακτηριστικά υφής.

Σε ένα σύστημα αναγνώρισης και ταξινόμησης εικόνων που βασίζεται σε χαρακτηριστικά, μπορούμε χρησιμοποιήσουμε και συνδυασμούς δύο η περισσοτέρων τύπων χαρακτηριστικών. Φυσικά, προκύπτουν περιορισμοί οι οποίοι εξαρτώνται από το είδος του ταξινομητή καθώς και από τις απαιτήσεις της κάθε εφαρμογής.

Γενικά, ένα σύνολο από χαρακτηριστικά θεωρείται αποδεκτό όταν πληροί τις ακόλουθες ιδιότητες:

- **Διακριτικότητα.** Συγκεκριμένα, πρέπει τα χαρακτηριστικά να είναι ικανά να διαχωρίζουν τα αντικείμενα μεταξύ τους, δηλαδή, για διαφορετικά αντικείμενα να καταλήγουμε σε διαφορετικό σημείο του χώρου των χαρακτηριστικών
- **Αξιοπιστία**. Θεωρώντας δύο η περισσότερα παρόμοια αντικείμενα, θα πρέπει οι. τιμές που προκύπτουν να είναι ικανοποιητικά κοντινές, ώστε με την χρήση των χαρακτηριστικών να προκύπτει ότι όντως αυτά τα αντικείμενα είναι παρόμοια.
- **Μικρό υπολογιστικό κόστος.** Ανάλογα με την εφαρμογή πρέπει να έχουμε όσο το δυνατόν μικρότερο υπολογιστικό κόστος.
- **Καλή προσαρμογή με την διαδικασία ταξινόμησης**. Σε συστήματα αναγνώρισης , μετά την εξαγωγή χαρακτηριστικών, έχουμε το στάδιο της ταξινόμησης. Η μορφή των χαρακτηριστικών, που εξάγονται , (πχ σε πίνακες) οφείλουν να ταιριάζουν, δηλαδή να είναι συμβατές με την μέθοδο της ταξινόμησης.
- Τέλος, είναι σημαντικό τα χαρακτηριστικά να είναι **ανεξάρτητα της κλιμάκωσης, της περιστροφής, της μετατόπισης** αλλά και **να μην είναι ευαίσθητα στον θόρυβο**.

Στο Σχήμα 1γ παρουσιάζεται ένα τυπικό σχήμα αναγνώρισης και ταξινόμησης αντικειμένων.[4]



Παρατηρούμε, ότι πριν από το στάδιο της εξαγωγής των χαρακτηριστικών, έχουμε το στάδιο της προεπεξεργασίας, το οποίο περιλαμβάνει τεχνικές βελτιστοποίησης της εικόνας και οποιαδήποτε άλλη απαραίτητη εφαρμογή για την επεξεργασία της εικόνας. Η Τμηματοποίηση περιλαμβάνει τεχνικές προσαρμοσμένες στον τρόπο παρουσίασης-παράστασης των αντικειμένων της εικόνας.

1.2.2.2 ΒΑΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΕΞΑΓΩΓΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Δεδομένης της παραπάνω περιγραφής και κατηγοριοποίησης των χαρακτηριστικών, θεωρείται σκόπιμη η περαιτέρω αναφορά και περιγραφή σε συγκεκριμένα χαρακτηριστικά, τα οποία περιγράφονται στο [5] και στο οποίο στηρίχθηκε η εργασία αυτή.

GIST: Ο αλγόριθμος GIST υπολογίζει την ενέργεια εξόδου μέσα από ένα σύνολο 24 φίλτρων. Τα φίλτρα αυτά μοιάζουν με φίλτρα Gabor (τα φίλτρα Gabor,στην ΨΕΕ είναι φίλτρα ανίχνευσης ακμών) συντονισμένα σε 8 κατευθύνσεις σε 4 διαφορετικές κλίμακες. [6]

HISTOGRAM OF ORIENTED GRADIENTS (HOG): Το HOG είναι ένας αλγόριθμος που χρησιμοποιείται στην ΨΕΕ για την ανίχνευση αντικειμένων σε εικόνες. Η τεχνική αυτή μετράει περιπτώσεις προσανατολισμένης κλίσης σε

τοπικά τμήματα μιας εικόνας. Η μέθοδος αυτή είναι παρόμοια με εκείνες του προσανατολισμού ακμών σε ιστογράμματα, των SIFT (περιγράφονται παρακάτω) καθώς και των shape context (το shape context περιγράφει το σχήμα ενός αντικειμένου ,περιμετρικά. Ουσιαστικά, παίρνει n σημεία από την περιφέρεια του αντικειμένου που θέλουμε και εξαγάγει έτσι το σχήμα του) αλλά διαφέρουν στο γεγονός ότι υπολογίζονται σε ένα πυκνό πλέγμα από ομοιόμορφα κατανεμημένα κελιά και χρησιμοποιούν επικαλυπτόμενη τοπική κανονικοποίηση αντίθεσης για βελτιωμένη ακρίβεια [7]

SCALE-INVARIANT FEATURE TRANSFORM (SIFT) : είναι ένας αλγόριθμος που ανιχνεύει και περιγράφει τοπικά χαρακτηριστικά σε εικόνες. [8]. Ως τοπικά χαρακτηριστικά ορίζονται τα χαρακτηριστικά αυτά που επιτρέπουν σε μια εφαρμογή να βρει τοπικές δομές μιας εικόνας, που επαναλαμβάνονται ,να τις κωδικοποιήσει σε μία παρουσίαση που είναι αναλλοίωτη σε μια σειρά από μετασχηματισμούς εικόνας, όπως μετάφραση, περιστροφή, κλιμάκωση κ.ο.κ. Τα προκύπτοντα χαρακτηριστικά, στη συνέχεια αποτελούν τη βάση για την αναγνώριση συγκεκριμένων αντικειμένων. [8]

SELF-SIMILARITY (SSIM): Οι περιγραφείς SSIM είναι ένα σημαντικό είδος εξαγωγής τοπικών χαρακτηριστικών εικόνων και βίντεο. Συνήθως χρησιμοποιούνται για ανίχνευση, ταυτοποίηση και αναγνώριση. Ουσιαστικά γίνεται ανίχνευση ομοιοτήτων αντικειμένων στα πλαίσια μιας η περισσοτέρων εικόνων.

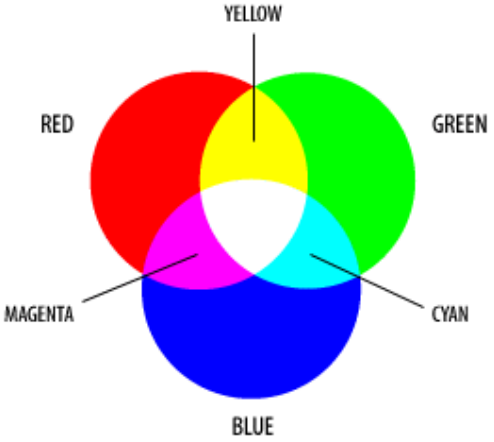
LOCAL BINARY PATTERNS (LBP): Τα LBP είναι τα χαρακτηριστικά στα οποία βασίστηκε η πτυχιακή και οι αλγόριθμοι που αναπτύχθηκαν μετέπειτα. Στοχεύουν στην αποτύπωση και κωδικοποίηση τοπικών χαρακτηριστικών των εικόνων και η κατανομή τους χαρακτηρίζει τις εικόνες. Φυσικά γίνεται αναλυτική επεξήγηση και παρουσίαση αυτού του τύπου των χαρακτηριστικών στο Κεφάλαιο 1.3 [9]-[10]

TINY IMAGES: Ουσιαστικά, ο αλγόριθμος αυτός είναι ο πιο απλοποιημένος αλγόριθμος προσδιορισμού χαρακτηριστικών για αναγνώριση και ταξινόμηση εικόνων. Ο αλγόριθμος αυτός μειώνει το μέγεθος των εικόνων (32χ32 64χ64 κ.ο.κ) και συγκρίνει τις εικόνες στο επίπεδο του χρωματικού χώρου, δηλαδή σε επίπεδο RGB,CIE κ.ο.κ (αυτές οι έννοιες εξηγούνται αναλυτικά παρακάτω). Αυτό καθιστά τον αλγόριθμο αυτό πιο εφικτό υπολογιστικά.

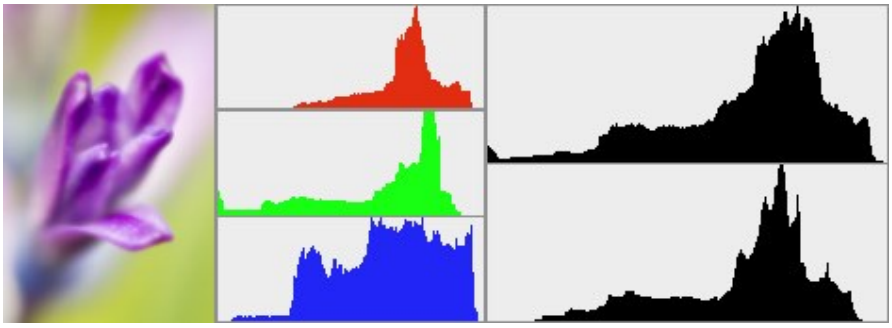
LINE FEATURES: Ανιχνεύονται οι ακμές, με τον ανιχνευτή ακμών Canny. Ο ανιχνευτής ακμών Canny είναι ένας μια τεχνική ανίχνευσης ακμών που χρησιμοποιεί έναν αλγόριθμο πολλαπλών σταδίων για να ανιχνεύσει ένα ευρύ φάσμα των ακμών σε εικόνες [11]. Για κάθε εικόνα δημιουργούνται δύο ιστογράμματα, που στηρίζονται στα στατιστικά των ανιχνευμένων ακμών. Το ένα ιστόγραμμα αντιστοιχεί στις γωνίες των ακμών και το άλλο στο μήκος τους.

COLOR HISTOGRAMS: Στις οθόνες των υπολογιστών κατά κύριο λόγο έχουμε την χρήση του χρωματικού μοντέλου RGB, όπως φαίνεται στο Σχήμα 1δ. Κάθε εικονοστοιχείο αποκτά χρώμα ως συνδυασμό τριών χρωματικών συνιστωσών (καναλιών),Red (κόκκινο) - Green (πράσινο) - Blue (μπλε) – RGB. Λαμβάνοντας υπόψιν, την συχνότητα εμφάνισης κάθε απόχρωσης στο εκάστοτε κανάλι, μπορούμε να σχηματίσουμε μία γραφική παράσταση, η οποία αποτελεί το ιστόγραμμα πιθανότητας κάθε χρωματικής συνιστώσας όπως παρατίθεται στο Σχήμα 1ε. Ο προσδιορισμός των πιθανοτήτων είναι απλή προσέγγιση αφού είναι γνωστός και ο συνολικός αριθμός των εικονοστοιχείων της εικόνας που προκύπτει από τις διαστάσεις της. Τα Χρωματικά

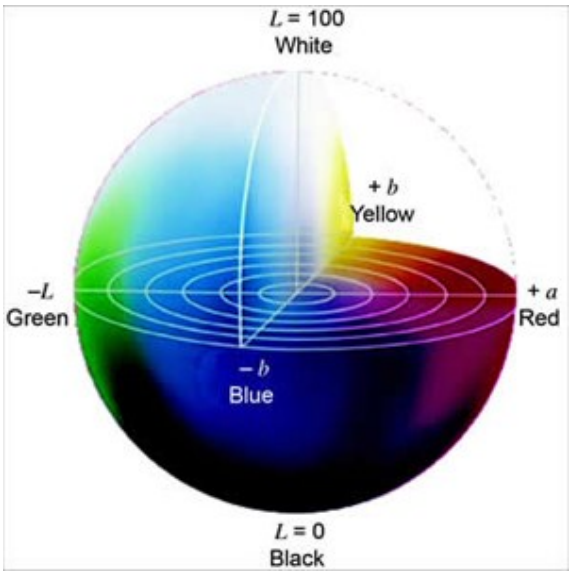
Ιστογράμματα (Color Histograms) είναι ευρέως διαδεδομένα για την χρήση τους στην ανάκτηση εικόνων.
Στην έρευνα που διεξήγαγε το MIT και στην οποία στηρίχθηκε τμήμα της εργασίας αυτής, χρησιμοποιήθηκε ο χρωματικός μετασχηματισμός CIE L*a*b, συνδυάζοντας ιστογράμματα του χώρου αυτού. Σύμφωνα με το μετασχηματισμό CIE L*a*b , η έννοια του χρώματος μπορεί να διαιρεθεί σε δύο μέρη: την φωτεινότητα (brightness) και τη χρωματικότητα (chromaticity). Για παράδειγμα το λευκό χρώμα, θεωρείται ένα φωτεινό χρώμα, το γκρι, μπορεί να θεωρηθεί μια λιγότερο φωτεινή έκδοση του λευκού χρώματος, συνεχίζοντας με αυτόν τον τρόπο χαρακτηρίζονται όλα τα χρώματα. Ο μετασχηματισμός που υλοποιείται έτσι είναι χρωματικό μοντέλο το οποίο ονομάστηκε CIE-XYZ. Ανεξάρτητα από τον χρωματικό μετασχηματισμό, ένα από τα επιθυμητά χαρακτηριστικά του χρωματικού συστάματος που προκύπτει είναι η δυνατότητα του συστήματος να καταγράφει / απεικονίζει ώστε να γίνεται αντιληπτή και η ελάχιστη χρωματική μεταβολή. Οι χρωματικοί χώροι που ικανοποιούν την τελευταία ιδιότητα ονομάζονται ομοιόμορφοι χρωματικοί χώροι. Οι χώροι RGB καθώς και CIE-XYZ δεν ικανοποιούσαν την συνθήκη αυτή. Το 1976 η CIE (*Comission Internationale de l'Eclairage* (International Commission on Illumination) καθιέρωσε τους ομοιόμορφους χρωματικούς χώρους CIE L*u*v και CIE L*a*b. Η παράμετρος L είναι η παράμετρος που καθορίζει την αντιληπτή φωτεινότητα, ενώ τα a,b την χρωματικότητα που προκύπτει από την ανάμειξη κόκκινου-πράσινου και κίτρινου-μπλε αντίστοιχα. Όπως φαίνεται και στο Σχήμα 1στ [12].



Σχήμα 1δ RGB μοντέλο.



Σχήμα 1ε Χρωματικό Ιστόγραμμα.

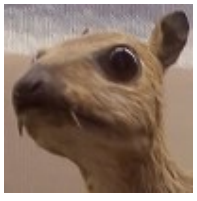


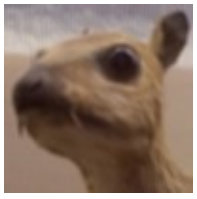
Σχήμα 1στ χρωματικό μοντέλο CIE L*a*b.

1.2.3 ΠΥΡΗΝΕΣ

1.2.3.1 ΟΡΙΣΜΟΣ: ΤΙ ΕΙΝΑΙ Ο ΠΥΡΗΝΑΣ

Στην Ψηφιακή Επεξεργασία Εικόνας, ως πυρήνας, έννοια που αναφέρεται και ως πίνακας συνέλιξης ή μάσκα ,ορίζεται ένας πίνακας, ο οποίος χρησιμοποιείται για το θόλωμα (blurring), για την δημιουργία ανάγλυφου (embossing), για την ανίχνευση ακμών (edge detection), όξυνση (sharpening) και πολλών άλλων αλγορίθμων προ-επεξεργασίας. Συνήθως αυτό υλοποιείται με την πράξη της συνέλιξης ανάμεσα στον πυρήνα και την εικόνα όπως φαίνεται στο Σχήμα 1ζ.

Original	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge-Detect	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	

	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Blur*	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Σχήμα 1ζ. Παραδείγματα συνέλιξης πυρήνα με εικόνα.

Μία ακόμη έννοια που συναντάμε στους πυρήνες είναι η κανονικοποίηση (Normalization). Ουσιαστικά διαιρούμε κάθε στοιχείο του πυρήνα, με το άθροισμα της απόλυτης τιμής των στοιχείων αυτών. Έτσι, εξασφαλίζουμε ότι τα εικονοστοιχεία της εικόνας εξόδου είναι του ίδιου σχετικού πλάτους με εκείνα της εικόνα εισόδου.
Σχετικά με τα SVM ,κάνουμε χρήση πυρήνων όταν τα στοιχεία εισόδου δεν μπορούν να διαχωριστούν γραμμικά. Επομένως θέλουμε να μεταβούμε από τον αρχικό χώρο X σε έναν άλλο, πιο μεγάλο όπου τα στοιχεία θα χωρίζονται γραμμικά. Ο γενικός ορισμός ενός πυρήνα είναι:

$K(x,y)=\varphi(x)^T\varphi(y)$,

,όπου φ(x) είναι μια συνάρτηση που μετασχηματίζει τον αρχικό χώρο σε έναν άλλο μεγαλύτερο. [13]

1.2.3.2 ΒΑΣΙΚΟΙ ΠΥΡΗΝΕΣ

Κατά την εκπόνηση της πτυχιακής βρεθήκαμε μπροστά σε πολλών ειδών πυρήνες. Κατά την εκτέλεση, κάθε είδους πειραματικού κώδικα, χρησιμοποιήθηκαν σχεδόν όλοι οι πυρήνες που εμπεριέχονταν στον κώδικα του MIT, ώστε να βρεθεί εκείνος, που σε συνδυασμό με τον κατάλληλο αλγόριθμο, θα δώσει ,όσο δύναται, καλύτερα αποτελέσματα. Αυτοί είναι οι: rbf,echi2,kchi,kchi2,kl1,kl2

1.2.4 ΠΛΗΡΟΦΟΡΙΕΣ ΣΧΕΤΙΚΑ ΜΕ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΟΥ ΚΩΔΙΚΑ

1.2.4.1 ΔΗΜΙΟΥΡΓΙΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ COMPUTE_FEATURE

1.2.4.1.1ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ

Πριν την αναλυτική περιγραφή των βασικών συναρτήσεων εκτέλεσης του κώδικα, κρίνεται σκόπιμη η περιγραφή ορισμένων βασικών εννοιών, οι οποίες αναφέρονται στα παρακάτω Κεφάλαια.

Αρχικά, η βασική έννοια που αναφέρεται στα παρακάτω Κεφάλαια, είναι εκείνη των χαρακτηριστικών(features).Ως χαρακτηριστικό, ορίζεται ένα στοιχείο, το οποίο προκύπτει μετά από μία ειδική επεξεργασία που γίνεται πάνω σε μία εικόνα(δηλαδή ενός πίνακα nxm διαστάσεων). Για παράδειγμα, η κύρια διαγώνιος μιας εικόνας μπορεί να αποτελεί χαρακτηριστικό της.

Ακόμη, γίνεται λόγος για συναρτήσεις. Μία συνάρτηση στο Matlab είναι ό,τι ακριβώς σε όλες τις γλώσσες προγραμματισμού. Δηλαδή ένα κομμάτι κώδικα που επιτελεί μία συγκεκριμένη διεργασία.

Εν συνεχεία βλέπουμε, την έννοια των «παραδειγμάτων εκπαίδευσης» και «παραδειγμάτων ελέγχου». Ουσιαστικά τα παραδείγματα εκπαίδευσης είναι τα χαρακτηριστικά επιλεγμένων εικόνων από κάθε κατηγορία που χρησιμοποιούνται για να εκπαιδευτεί το SVM ενώ τα παραδείγματα ελέγχου αποτελούνται από όλες τις εικόνες, τις οποίες περνάμε στο SVM μετά από την εκπαίδευση του για να ελέγξουμε την ακρίβεια αναγνώρισης.

1.2.4.1.2 ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ COMPUTE FEATURE

Για την σωστή ολοκληρωμένη εκτέλεση/μετατροπή του κώδικα, κρίνεται απαραίτητη η περιγραφή της λειτουργίας του[1] :

1. Η πρώτη συνάρτηση η οποία καλείται ονομάζεται : «*compute_feature*» , η οποία αποσκοπεί στη δημιουργία της δομής των χαρακτηριστικών τα οποία

[1] Το περιβάλλον που χρησιμοποιήθηκε είχε υλοποιηθεί σε unix-περιβάλλον με αποτέλεσμα, δεδομένου ότι στην εργασία αυτή χρησιμοποιήθηκε λειτουργικό windows, να είναι απαραίτητη η των slash (/) σε backslash (\).

αποθηκεύονται στον φάκελο : «*source/data\class_15scene(ή την εκάστοτε βάση δεδομένων που χρησιμοποιείται)\features*».

2. Στη συνέχεια γίνεται η επιλογή της βάσης των ψηφιακών εικόνων, στην οποία θα διενεργηθεί η επεξεργασία για να εξαχθούν τα χαρακτηριστικά. Η συνάρτηση διαθέτει στους χρήστες 3 κατηγορίες βάσεων δεδομένων: α) την βάση δεδομένων 8 κλάσεων (κατηγοριών) εικόνων, β) την βάση δεδομένων 15 κλάσεων (κατηγοριών) εικόνων και τέλος γ) την βάση δεδομένων SUN 397 κλάσεων (κατηγοριών) εικόνων. Η επιλογή της βάσης των εικόνων γίνεται με αλλαγή από σχόλια σε γραμμές εντολών των αντίστοιχων δηλώσεων στο αρχείο της συνάρτησης ("*compute feature*").
3. Τέλος πρέπει να γίνει έλεγχος για τις διαδρομές των εικόνων , την αποθήκευση χαρακτηριστικών και του αρχείου split –στο οποίο θα αναφερθούμε στη συνέχεια- είναι σωστά.

Τα παραπάνω φαίνονται στη συνέχεια, όπου X η βάση εικόνων που έχουμε επιλέξει.

```
conf.imagePath = '..\..\data\scene_Xclass\image\';
conf.featPath  = '..\..\data\scene_Xclass\feature\';
conf.splitFile = '..\..\data\scene_Xclass\split10.mat';
```

Στη συνέχεια γίνεται καθορισμός του αριθμού των εικόνων που θα χρησιμοποιηθούν για εκπαίδευση (training) καθώς και για ταξινόμηση με χρήση των SVM. Οι εντολές αυτές κάνουν μια αρχικοποίηση των μεταβλητών που καθορίζουν τους αριθμούς αυτούς (πλήθος εικόνων).

```
conf.max_num_train = inf;
conf.max_num_test = inf;
```

Ακολουθεί καθορισμός του αλγορίθμου εξαγωγής χαρακτηριστικών, στο πεδίο που φαίνεται παρακάτω. , διαγράφοντας το σχόλια ανάλογα , και κρατώντας το χαρακτηριστικό της επιλογής μας. Στο παράθυρο που ακολουθεί, έχει ορισθεί ως αλγόριθμος εξαγωγής χαρακτηριστικών ο αλγόριθμος "lbp:local binary patterns"

```
% select feature to be computed

%conf.featNames   =
{'sparse_sift','gistPadding','line_hists','lbp','lbphf','geo_texto
n','geo_color','geo_map8x8','hog2x2','texton','gist','tiny_image',
'ssim','denseSIFT'} ;

conf.featNames   = {'lbp'} ;
```

Τελευταίο βήμα είναι ο καθορισμός παραμέτρων εισόδου για την εκτέλεση του αλγορίθμου εξαγωγής χαρακτηριστικών που ορίσθηκε στο προηγούμενο στάδιο. Εδώ, ακολουθούν αρκετές γραμμές κώδικα (σε σχόλια). Η επιλογή γίνεται με τον ίδιο τρόπο (απαλοιφή των σχολίων από τις γραμμές οι οποίες αφορούν τον αλγόριθμο εξαγωγής χαρακτηριστικών που επιλέξαμε).

```
conf.lbp.extractFn              = @lbp_feature ;
conf.lbp.outside_quantization   = false ;
```

Μετά την ολοκλήρωση και του τελευταίου βήματος, ο κώδικας είναι έτοιμος για εκτέλεση.
Παρατηρώντας τον κώδικα βλέπουμε τα δύο τελευταία τμήματα, με τίτλους:
α) "get *list of images*" – Ανάκτηση λίστας εικόνων και
β) "*compute image features for all images*" – Υπολογισμός των χαρακτηριστικών για όλες της εικόνες.

Στο πρώτο τμήμα (*Ανάκτηση λίστας εικόνων*), επιτυγχάνεται η ανάκτηση των κατηγοριών αλλά και των εικόνων που θα χρησιμοποιηθούν αργότερα για εκπαίδευση (training) και έλεγχο (testing) στο SVM ,ενώ στο αμέσως επόμενο (*Υπολογισμός των χαρακτηριστικών για όλες τις εικόνες*), γίνεται ο υπολογισμός των χαρακτηριστικών για όλες τις εικόνες, όλων των κατηγοριών.

Τέλος, αν γίνει κάποια αλλαγή στον τρέχοντα αλγόριθμο εξαγωγής χαρακτηριστικών, απαιτείται η επανεκτέλεση του κώδικα για την αποθήκευση τους αφού πρώτα διαγραφούν τα ήδη προσδιορισμένα χαρακτηριστικά από τον αντίστοιχο φάκελο: *source/data/scene_Xclass/features*.

1.2.4.1.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ COMPUTE FEATURE

Στα αποτελέσματα της συνάρτησης αυτής συμπεριλαμβάνονται:

α) Τα αποτελέσματα κατά την εκτέλεση του κώδικα, τα οποία εμφανίζονται στο παράθυρο εντολών (command window), και είναι βοηθητικά.

β) Τα αποτελέσματα στο χώρο των μεταβλητών (workspace) που περιλαμβάνει τις μεταβλητές / πίνακες / δομές που σχηματίζονται κατά την διάρκεια εκτέλεσης του κώδικα και

γ) τα αποθηκευμένα χαρακτηριστικά που εξήχθησαν από τις εικόνες και διατηρούνται στον φάκελο features της εκάστοτε διαδρομής της βάσης δεδομένων όπως αναφέρθηκε και πιο πάνω.

Κατά την εκτέλεση του κώδικα, είναι φανερό από το παράθυρο εντολών πως ο υπολογισμός των χαρακτηριστικών γίνεται με τυχαία επιλογή της σειράς και της κατηγορίας της εικόνας εισόδου. Για παράδειγμα με τη χρήση του αλγορίθμου εξαγωγής χαρακτηριστικών «*tiny images*», έχουμε τα παρακάτω αποτελέσματα:

```
total number of images = 44850

42965

image /MITtallbuilding\image_0293    feature tiny_image    time 0.164129

37151

image /MITinsidecity\image_0078    feature tiny_image    time 0.057710

20043

image /MITopencountry\image_0258    feature tiny_image    time 0.059495

34149
```

Αυτό, υλοποιείται από τις παρακάτω γραμμές κώδικα :

```
myRstream = RandStream('mt19937ar','Seed',sum(100*clock));
RandStream.setDefaultStream(myRstream);
```

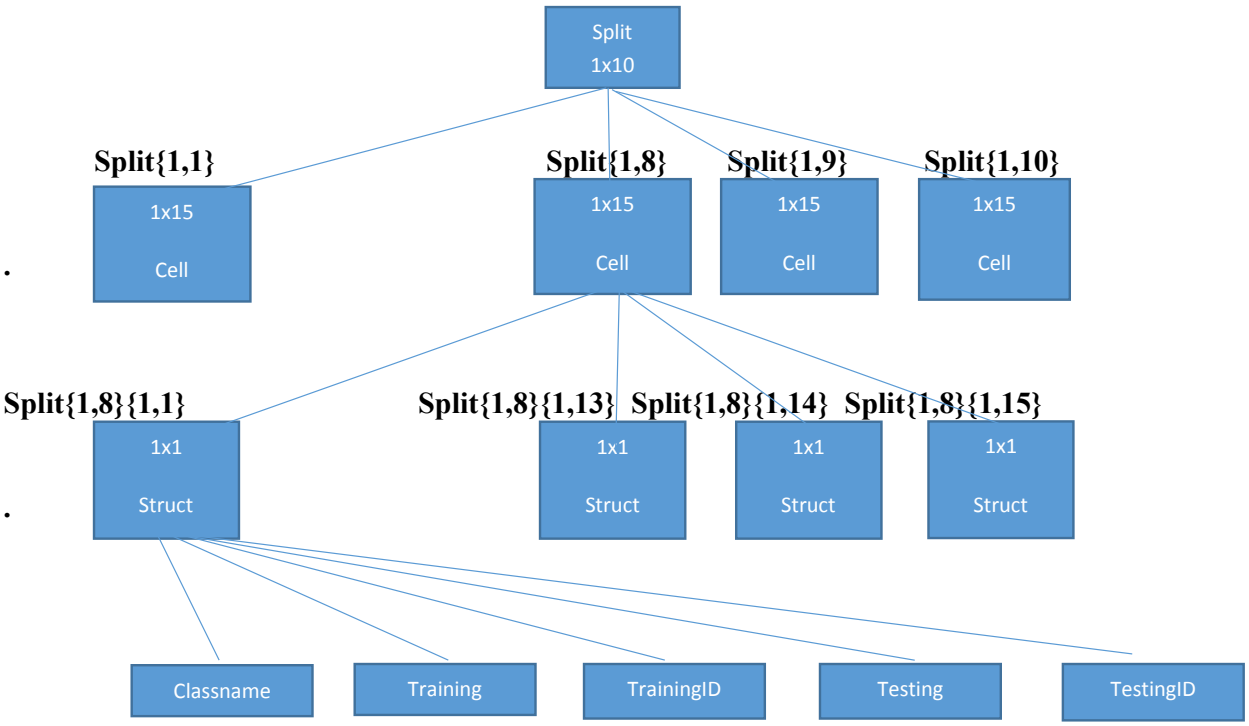
```
for i=randperm(length(parameterCell))
    disp(num2str(i));
    extractFeature(parameterCell{i},conf);
end
```

Γίνεται τυχαία επιλογή εικόνας (βασισμένη στην τυχαία βοηθητική μεταβλητή i, η οποία παράγεται από τυχαία επιλογή αριθμού μεταξύ του 1 και του συνόλου των εικόνων και από τις 10 στήλες του αρχείου split) για εξαγωγή χαρακτηριστικών (από τη συνάρτηση «*extractFeature*»). Ακόμη, γίνεται αναφορά στο όνομα του χαρακτηριστικού, στο χρόνο που απαιτείται για την εξαγωγή του χαρακτηριστικού αυτού από την εκάστοτε εικόνα, καθώς και στον κωδικό («id») της εικόνας.
Παράλληλα στο χώρο μεταβλητών «*workspace*» παρατηρούμε ένα σύνολο μεταβλητών/δομών/πινάκων κ.ο.κ στα οποία αποθηκεύονται πληροφορίες για την εκτέλεση και τα αποτελέσματα. Συγκεκριμένα:

1. *cnt*: προκύπτει από την λέξη "count" και είναι ένας μετρητής που περιέχει το σύνολο των εικόνων για όλους τους διαμοιρασμούς (split).
2. *conf*: προκύπτει από τη λέξη configure και κρατάει όλα τα στοιχεία που απαιτούνται για ρύθμιση παραμέτρων, όπως διαδρομές (paths), τα ονόματα των χαρακτηριστικών που εξάγονται , τη δεδομένη εκτέλεση από τις εικόνες κ.ο.κ.
3. *j,k,i* : είναι βοηθητικές μεταβλητές. Το i βοηθάει στην εκτέλεση των 10 split κατά την ανάκτηση της λίστας εικόνων και μετέπειτα ,κατά τον υπολογισμό των χαρακτηριστικών , κρατάει τον τυχαίο αριθμό επιλογής εικόνων .Το j κρατάει τον αριθμό της τρέχουσας κατηγορίας (σκέλος ανάκτησης λίστας εικόνων) ενώ το k τον αριθμό της τρέχουσας εικόνας (σκέλος ανάκτησης λίστας εικόνων).
4. *myRStream*(myRandomStream) : Περιέχει τις πληροφορίες για την δημιουργία της τυχαίας σειράς αριθμών που αναφέρθηκε παραπάνω.
5. *parameterCell* : Εδώ βρίσκονται όλες οι εικόνες όλων των κατηγοριών και για τους 10 διαμοιρασμούς (split). Το μέγεθος του πίνακα αυτού εξαρτάται από τον αριθμό των κατηγοριών καθώς και των εικόνων.
6. *Split*: είναι ένας πίνακας 1x10 που υπάρχει ήδη αποθηκευμένος, πριν την εκτέλεση της συνάρτησης *compute feature*. Η κάθε μία από αυτές τις στήλες περιέχει έναν πίνακα 1xX (οπου X ο συνολικός αριθμός των κατηγοριών-κλάσεων). Κάθε ένας από αυτούς του 1xX πίνακες περιέχει 15 δομές έκαστος (αφού αναφερόμαστε στις 15 κλάσεις της βάσης των εικόνων,). Κάθε μία περιέχει από 5 στοιχεία που αναφέρονται στη συνέχεια:
    1) *Classname* (όνομα κλάσης),το οποίο περιέχει το όνομα της κλάσης
    2) *Training* (εκπαίδευση), όπου είναι ένας πίνακας 1x100 και κρατάει τα ονόματα των 100 εικόνων που χρησιμοποιήθηκαν για την εκπαίδευση του SVM.

3) *TrainingID* (ID (ταυτότητα) εικόνας εκπαίδευσης): επίσης ένας πίνακας 1x100.Κρατάει τα αντίστοιχα ID της εικόνων με βάση τα οποία έγινε η εκπαίδευση.
4) *Testing* (έλεγχος), είναι ένας πίνακας 1xY όπου Y είναι ο συνολικός αριθμός των εικόνων της κατηγορίας *Classname* που βρισκόμαστε. Εδώ, εμπεριέχονται οι εικόνες, που μπήκαν στο SVM για ταξινόμηση και αναγνώριση.
5) *TestingID* (ID (ταυτότητα) εικόνων ελέγχου): επίσης ένας πίνακας 1xY όπου Y είναι ο συνολικός αριθμός των εικόνων της κατηγορίας *Classname* που βρισκόμαστε. Εδώ, καταχωρούνται τα ID των εικόνων, που μπήκαν στο SVM για ταξινόμηση και αναγνώριση.

Στο σχήμα 1η παρατίθεται ένα βοηθητικό διάγραμμα, ενός μέρους της δομής split



Τέλος, τα αποθηκευμένα χαρακτηριστικά, καταχωρούνται στον φάκελο: *source/data/scene_Xclass/features* με όλες τις κατηγορίες εικόνων, τις αντίστοιχες εικόνες, τις μετασχηματισμένες με βάση τον αλγόριθμο εξαγωγής χαρακτηριστικών που επιλέχθηκε.

1.2.4.2 ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΕΚΤΕΛΕΣΗ ΠΥΡΗΝΩΝ ΤΟΥ SVM

1.2.4.2.1 ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΣΥΝΑΡΤΗΣΗΣ RUN_KERNEL_SVM

Όπως και στην προηγούμενη συνάρτηση εξαγωγής των χαρακτηριστικών, έτσι και στη συνάρτηση "*run_kernel_svm*", χρήσης των πυρήνων και των χαρακτηριστικών για την εκτέλεση του SVM , περιέχεται η δομή και η λειτουργία της διαδικασίας αυτής.

Αμέσως μετά την *compute_feature*,γίνεται η εκτέλεση της *run_kernel_svm* ώστε να εξαχθούν τα τελικά αποτελέσματα στους φακέλους :

source/data/scene_Xclass/results (αποτελέσματα) και
source/data/scene_Xclass/kernels (πυρήνες)

Στο αρχείο *run_kernel_svm*, στις πρώτες γραμμές κώδικα γίνεται η αρχικοποίηση μιας μεταβλητής n η οποία καταγράφει τον αριθμό των πυρήνων και των χαρακτηριστικών που χρησιμοποιούμε. Πιο κάτω, όπως και στην *compute_feature* γίνεται επιλογή της κατηγορίας των κλάσεων των εικόνων που θέλουμε να χρησιμοποιήσουμε (8,15 ή SUN397). Για την εκάστοτε κατηγορία έχουμε την ρύθμιση των διαδρομών (paths), ώστε να πάρουμε τα χαρακτηριστικά που εξήχθησαν προηγουμένως από την εκτέλεση της *compute_feature*, τον φάκελο των αποτελεσμάτων και τον φάκελο που θα αποθηκευτούν οι πυρήνες. Τέλος, έχουμε τον καθορισμό της διαδρομής που βρίσκεται το αρχείο διαμοιρασμού της εκπαίδευσης σε 10 στάδια (split).

Οι διαδρομές αυτές φαίνονται παρακάτω, σε αντίστοιχη σειρά με την αναφερόμενη:

```
conf.featPath   = '..\..\data\scene_15class\feature\';
conf.kernelPath = '..\..\data\scene_15class\kernel\';
conf.resultPath = '..\..\data\scene_15class\result\';
conf.splitFile  = '..\..\data\scene_15class\split10.mat';
```

Στη συνέχεια γίνεται η αρχικοποίηση του αριθμού εικόνων εκπαίδευσης του SVM (100 εικόνες) καθώς και του αριθμού των εικόνων που χρησιμοποιούνται κατά την διαδικασία εκτέλεσης του SVM για αναγνώριση και ταξινόμηση (αρχικοποιείται στο άπειρο καθώς ο αριθμός εικόνων που περιέχονται από κατηγορία σε κατηγορία αλλάζει).

```
conf.max_num_train = 100;
conf.max_num_test = inf;
```

Επόμενο βήμα, η ρύθμιση των πυρήνων. Στο πεδίο Kernel Configuration υπάρχουν όλα τα χαρακτηριστικά, μέσα σε σχόλια. Με την αφαίρεσή τους καθορίζεται ο αλγόριθμος εξαγωγής χαρακτηριστικών καθώς και ο πυρήνας της επιλογής μας.Το τμήμα του κώδικα που αντιστοιχεί σε κάθε χαρακτηριστικό μαζί με τον πυρήνα του αρχίζει ουσιαστικά από το σημείο «*n=n+1*», μέχρι εκεί που ξεκινάει το επόμενο «*n=n+1*». Δίνεται, ακόμη, μία ποικιλία συνδυασμών ανάμεσα σε χαρακτηριστικά και πυρήνες, καθώς μπορούμε να συνδυάσουμε :

1. Ένα χαρακτηριστικό με διάφορους πυρήνες.
2. Πολλά χαρακτηριστικά με έναν πυρήνα.
3. Πολλά χαρακτηριστικά με πολλούς πυρήνες.
4. Ένα μεμονωμένο χαρακτηριστικό με έναν πυρήνα.

Αυτό επιτυγχάνεται βγάζοντας από σχόλια όσα κομμάτια κώδικα επιθυμούμε. Φυσικά, όσο περισσότερους συνδυασμούς έχουμε, τόσο πιο αργή γίνεται η εκτέλεση του κώδικα.

```
n=n+1;
parameterCell{n}.featureName  = 'geo_color';
parameterCell{n}.kernelName   = 'kchi2';
parameterCell{n}.weighted     = false;
parameterCell{n}.bow          = true;
parameterCell{n}.normalize    = true;
```

Για παράδειγμα, στο παραπάνω κομμάτι κώδικα, στη μεταβλητή *featureName* καταχωρείται το όνομα του χαρακτηριστικού προς χρήση, στην *kernelName* καταχωρείται το όνομα του πυρήνα που επιλέχθηκε. Ακόμη οι τρεις παραμέτροι weighted, bow, normalize στις οποίες δίνονται οι τιμές true,false μετασχηματίζουν ή όχι το τελικό αποτέλεσμα, μέσα από μια διαφορετική μαθηματική προσέγγιση. Να επισημανθεί επίσης ότι και εδώ γίνεται η χρήση του αρχείου διαμοιρασμού split.

Επόμενο τμήμα κώδικα («*individual kernels*») είναι ο υπολογισμός του πυρήνα/πυρήνων καθώς και των χαρακτηριστικών που έχουμε επιλέξει. Με την κλήση της *compute_kernel_svm* υλοποιείται η εκπαίδευση του svm. Το αρχείο split για κάθε μία από τις 15 κατηγορίες εικόνων έχει 100 επιλεγμένες εικόνες για training. Από αυτές κρατώνται οι επιδόσεις 6 συνόλων εικόνων [1 5 10 20 50 100]. (Αυτό φαίνεται από τα αποτελέσματα κατά την εκτέλεση του κώδικα στο παράθυρο εντολών).Εν συνεχεία, γίνεται η αποθήκευση των επιδόσεων αυτών στο αρχείο perf.mat (συντόμευση της λέξης performance – επίδοση) το οποίο αποθηκεύεται στον φάκελο source/data/results,ενώ παρακάτω αποθηκεύεται ο μέσος όρος των τιμών αυτών στην μεταβλητή avg_perf (average performance – μεσος όρος επίδοσης).

```
%% individual kernels
nLeng = length(parameterCell);
disp(num2str(nLeng));
for i=1:length(split)
    for j=1:nLeng
        disp(['split ' num2str(i) ' kernel ' num2str(j)]);
        para = parameterCell{j};
        para.splitID = i;
        perf(j,:,i) = compute_kernel_svm(para,conf);
        save([conf.resultPath 'perf.mat'],'perf');
    end
end
avg_perf = mean(perf,3);
avg_perf = avg_perf(:,end:-1:1);
```

Καθώς προχωράμε στο επόμενο κομμάτι κώδικα, βλέπουμε την διαδικασία δημιουργίας της γραφικής παράστασης του μέσου όρου των επιδόσεων (μεταβλητή avg_perf) στον άξονα Y ως προς τα 6 αποθηκευμένα σύνολα επιδόσεις στον άξονα X. Παρακάτω, στο μπλοκ «combine kernel configuration» (ρύθμιση συνδυασμού πυρήνων), οι μεταβλητές που δείχνουν τα δεδομένα,με βάση τα οποία γίνεται η εκτέλεση του κώδικα ως τώρα, αλλάζουν τιμές. Έχουμε έναν συνδυαστικό πυρήνα που ονομάζεται combine (συνδυασμός) , ενώ γίνεται και αλλαγή των χαρακτηριστικών σε all (όλα).Ακόμη, ο υπολογισμός αυτού του νέου συνδυαστικού πυρήνα γίνεται για κανονικοποιημένο και μη κανονικοποιημένο πυρήνα (εναλλαγή τιμών αληθής (true)

και ψευδής (false) της παραμέτρου normalize (κανονικοποίηση)).Με άλλα λόγια, αλλά και όπως παρατηρούμε παρακάτω (στο κομμάτι «combine kernel and

```
%% combine kernel configuration
n=0;
n=n+1;
paraCombine{n}.featureName  = 'all';
paraCombine{n}.kernelName   = 'combine';
paraCombine{n}.weighted     = false;
paraCombine{n}.bow          = false;
paraCombine{n}.normalize    = true;
n=n+1;
paraCombine{n}.featureName  = 'all';
paraCombine{n}.kernelName   = 'combine';
paraCombine{n}.weighted     = false;
paraCombine{n}.bow          = false;
paraCombine{n}.normalize    = false;
```

svm»(συνδυασμός πυρήνα και svm) γίνεται ο συνδυασμός όλων των αποτελεσμάτων όλων των πυρήνων (και χαρακτηριστικών), που έχουν επιλεγεί και εκτελέστηκαν ως τώρα.

Παρατηρώντας αυτό το κομμάτι του κώδικα, πλέον όλα γίνονται συνδυαστικά. Έχουμε ,όπως προαναφέρθηκε δύο συνδυαστικούς πυρήνες (έναν κανονικοποιημένο και έναν μη κανονικοποιημένο), οι οποίοι προκύπτουν από την κλήση της συνάρτησης combine_kernel (συνδυασμός πυρήνων).Επιπρόσθετα, γίνεται ξανά κλήση της compute_kernel_svm αυτή τη φορά για τους νέους πυρήνες. Τέλος, έχουμε τον υπολογισμό των νέων μέσων όρων επίδοσης αλλά και την εμφάνιση των τελικών αποτελεσμάτων ακρίβειας αναγνώρισης.

1.2.4.2.2 ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΥΝΑΡΤΗΣΗΣ RUN_KERNEL_SVM

Στα αποτελέσματα της συνάρτησης αυτής συμπεριλαμβάνονται και πάλι οι τρεις κατηγορίες που χωρίσαμε τα αποτελέσματα της compute feature: α) Τα αποτελέσματα κατά την εκτέλεση του κώδικα, τα οποία εμφανίζονται στο παράθυρο εντολών (command window). β) Τα αποτελέσματα στο χώρο των μεταβλητών (workspace) που περιλαμβάνει τις μεταβλητές/πίνακες/δομές που σχηματίζονται κατά την διάρκεια εκτέλεσης του κώδικα και γ) τα αποτελέσματα των πυρήνων που βρίσκονται στον φάκελο kernels αλλά και τα τελικά αποτελέσματα που βρίσκονται στο φάκελο results.

Ξεκινώντας από το παράθυρο εντολών βλέπουμε τα εξής:

```
split 1 kernel 1

_tiny_image__split_01__echi2__weighted_F__bow_F__normalize_F__.mat

train features are loaded..\..\data\scene_15class\feature\tiny_image\store\image_0159.mat

train features are packed

test features are loaded

test features are packed

kernels are computed

#train = 0100   Performance = 33.488323 %

#train = 0050   Performance = 29.689622 %

#train = 0020   Performance = 25.961735 %

#train = 0010   Performance = 21.915128 %

#train = 0005   Performance = 18.317199 %

#train = 0001   Performance = 13.459333 %
```

Όπως έχουμε αναφέρει και παραπάνω έχουμε χαρακτηριστικά με βάση τα οποία εκπαιδεύουμε το SVM και χαρακτηριστικά με βάση τα οποία το δοκιμάζουμε (μετά την εκπαίδευση).Έτσι και εδώ, για το χαρακτηριστικό «tiny image» για το πρώτο split του πυρήνα echi2 με τα weighted,bow και normalize να είναι ψευδή (ανενεργά) γίνεται η εκπαίδευση και η δοκιμή στο SVM.Εν συνεχεία, ενημερωνόμαστε για την φόρτωση και το πακετάρισμα των χαρακτηριστικών των εικόνων για εκπαίδευση αλλά και για δοκιμή (το path που παρεμβάλλεται είναι της τελευταίας εικόνας που βρίσκεται στο εκάστοτε split), λαμβάνουμε μήνυμα ότι έγινε ο υπολογισμός των πυρήνων και βλέπουμε τις αποδόσεις της εκπαίδευσης για κάποιες από τις 100 εικόνες που έχουν επιλεχθεί(βλ. ανάλυση δομής split αρχείου σελ. 8) και σταδιακά ο αριθμός των εικόνων εκπαίδευσης αυξάνεται μέχρι να χρησιμοποιηθούν και οι 100 εικόνες. Αυτή η διαδικασία γίνεται και για τους 10 διαχωρισμούς του αρχείου split.

```
_all__split_01__combine__weighted_F__bow_F__normalize_T__.mat

#train = 0100   Performance = 32.385988 %

#train = 0050   Performance = 29.794673 %

#train = 0020   Performance = 25.792470 %

#train = 0010   Performance = 21.885466 %

#train = 0005   Performance = 18.500834 %

#train = 0001   Performance = 13.601424 %

split 1 kernel 2

Train 1

Test 1

_all__split_01__combine__weighted_F__bow_F__normalize_F__.mat

#train = 0100   Performance = 33.488323 %

#train = 0050   Performance = 29.689622 %

#train = 0020   Performance = 25.961735 %

#train = 0010   Performance = 21.915128 %
```

Ακολουθεί η ίδια ακριβώς διαδικασία για τους συνδυαστικούς πυρήνες με την διαφορά ότι εδώ έχουμε έναν κανονικοποιημένο και έναν μη κανονικοποιημένο για κάθε split.
Κλείνοντας με τα αποτελέσματα την γραμμή εντολών του Matlab έχουμε την παρουσίαση των αποτελεσμάτων (δηλαδή της ακρίβειας αναγνώρισης).

```
combine kernel accuracy (normalized)   = 32.0

combine kernel accuracy (unnormalized) = 33.7
```

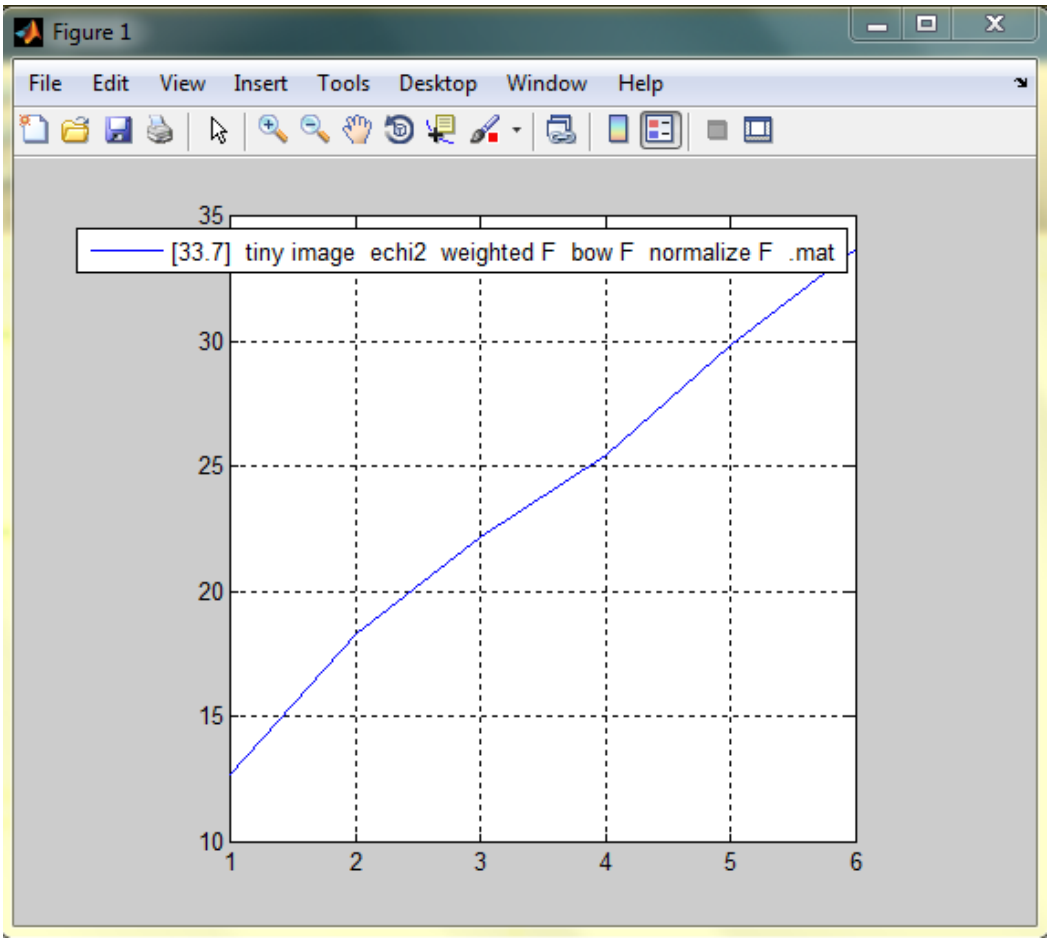
Στο workspace έχουμε τις μεταβλητές/δομές που χρησιμοποιούνται για την εκτέλεση της συνάρτησης αυτής καθώς και μεταβλητές/δομές που κρατάνε αποτελέσματα και λοιπά στοιχεία. Έτσι περιληπτικά, έχουμε:

1. avg_perf : Κρατάει την μέση απόδοση για κάθε πυρήνα ξεχωριστά.
2. avg_perf_comb: Εδώ έχουμε την μέση απόδοση των συνδυαστικών πυρήνων.
3. conf: Έχει αναλυθεί και στην παράγραφο της ανάλυσης αποτελεσμάτων του compute feature.Έχει ακριβώς την ίδια βοηθητική λειτουργία.
4. featMap: Πιθανών να είναι κάποια χαρτογράφηση των χαρακτηριστικών. Δεν περιέχει κάποια πληροφορία

5. foo: Μέγιστη μέση απόδοση συνδυαστικών πυρήνων.(προκύπτει απτά δύο τελικά αποτελέσματα).
6. i,j : βοηθητικές μεταβλητές.
7. mendian_avg_perf: Και πάλι βλέπουμε εδώ την μέγιστη μέση απόδοση (και πάλι προκύπτει από τα δύο τελικά αποτελέσματα).
8. n: Αριθμός πυρήνων (εδώ έχουμε δύο επειδή κρατάει τους δύο συνδυαστικούς πυρήνες και όχι αυτούς που διαλέξαμε εμείς εξαρχής).
9. nLeng: Αριθμός πυρήνων/χαρακτηριστικών που διαλέξαμε εμείς (στην προκειμένη περίπτωση είναι 1 καθώς έχουμε μόνο το tiny_image με τον πυρήνα echi2).
10. nLengCombine: Αριθμός συνδυαστικών πυρήνων.
11. ndx: Αριθμός των πυρήνων που εμείς επιλέξαμε.
12. outName: Όλα τα στοιχεία της εκτέλεσης(χαρακτηριστικό, πυρήνας κοκ).
13. para: Τρέχουσες παράμετροι σχετικά με το ποια χαρακτηριστικά χρησιμοποιούμε, σε ποιο split βρισκόμαστε κατά την εκτέλεση του κώδικα κ.ο.κ.
14. paraCombine:Παράμετροι/στοιχεία συνδυαστικών πυρήνων (αν είναι κανονικοποιημένοι κ.ο.κ)
15. parameterCell: Πληροφορίες σχετικά με τον πυρήνα που εμείς διαλέξαμε
16. perf: Κρατάει την τρέχουσα απόδοση των πυρήνων που εκτελούνται και παρουσιάζονται στη γραμμή εντολών.
17. perf_comb: Κρατάει τις αποδόσεις των δύο συνδυαστικών πυρήνων (κανονικοποιημένου, μη κανονικοποιημένου) που εκτελούνται και παρουσιάζονται στη γραμμή εντολών του Matlab.
18. plotLabel: Η ετικέτα που χρησιμοποιείται στην γραφική παράσταση που εμφανίζεται με το τέλος της εκτέλεσης.
19. split:Το γνωστό split αρχείο.

Ας στραφούμε τώρα ,στα αποθηκευμένα αποτελέσματα στους φακέλους kernels και results.Στον φάκελο των πυρήνων λοιπόν έχουμε όλα τα αποτελέσματα υπολογισμού των πυρήνων(όχι μόνο 6 όπως παρουσιάζονται στην γραμμή εντολών) αναλυτικά αποθηκευμένα σε πίνακες. Το όνομα του εκάστοτε αρχείου δείχνει για ποιον πυρήνα,split κ.ο.κ κρατάει αποτελέσματα. Το ίδιο ισχύει και για τα αποτελέσματα του SVM που βρίσκεται στον φάκελο results.

Καταληκτικά, εδώ έχουμε ένα ακόμη αποτέλεσμα .Μία γραφική παράσταση που δείχνει την απόδοση του SVM κατά την διάρκεια της εκτέλεσης. Περιέχει μία ετικέτα που προσδιορίζει τα στοιχεία εκτέλεσης (μέγιστη απόδοση, χαρακτηριστικό, πυρήνας, κανονικοποίηση κοκ),Ένα παράδειγμα φαίνεται στην παρακάτω εικόνα.

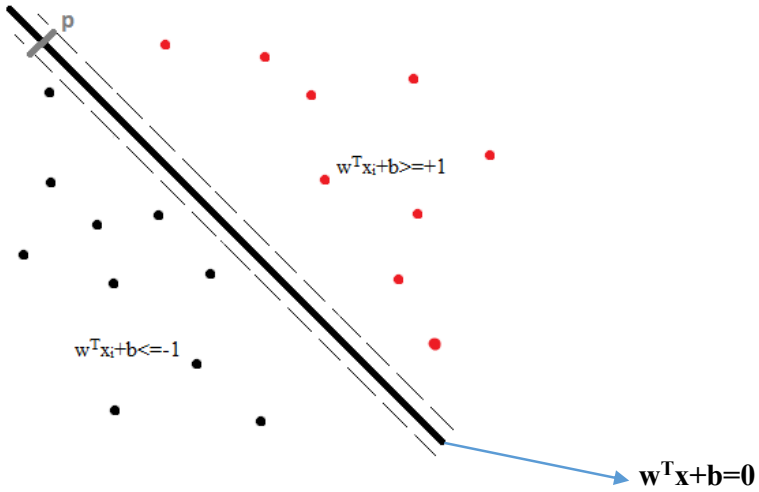


Συνοψίζοντας, στη συνάρτηση αυτή διαλέγουμε ένα ή περισσότερα χαρακτηριστικά καθώς και πυρήνες, και ακολουθεί εκπαίδευση του SVM καθώς και δοκιμή με βάση κάποιο συνδυασμό των χαρακτηριστικών-πυρήνων μέσω μιας μαθηματικής σχέσης για την εξαγωγή της απόδοσης του SVM.

1.2.5 SVM (Support Vector Machines)

Οι Μηχανές Διανυσμάτων Υποστήριξης (SVM) εμφανίστηκαν στο 1992 και βασίζονται στην στατιστική θεωρία μάθησης. Είναι ευρέος αποδεκτές και βρίσκουν εφαρμογή σε προβλήματα ταξινόμησης. Τα SVM χωρίζουν τα δεδομένα εκπαίδευσης σε δύο κλάσεις με την βοήθεια διανυσμάτων υποστήριξης. Υπάρχουν πολλοί τρόποι να χωριστεί ένα επίπεδο σε δύο κλάσεις, και με βάση την θεωρία των SVM ο διαχωρισμός αυτός γίνεται με την βοήθεια των διανυσμάτων υποστήριξης. Στόχος λοιπόν είναι η επιλογή των κατάλληλων w, b ,ώστε να γίνει η μεγαλοποίηση της απόστασης μεταξύ των παράλληλων υπερεπιπέδων, τα οποία διαχωρίζουν τα διανύσματα x_i σε δύο κλάσεις. Τα υπερεπίπεδα αυτά ικανοποιούν τις παραπάνω εξισώσεις

$w^T x_i+b \geq +1$ όταν $y_i=+1\}$
$w^T x_i+b \leq -1$ όταν $y_i=-1\ \} \geq y_i(w^T x_i +b)-1 \geq 0$



Σχήμα 1η.Υπερεπίπεδα διαχωρισμού.

Διαμορφώνεται λοιπόν το ακόλουθο πρόβλημα βελτιστοποίησης:

minimize ½ $||w||^2$

Με περιορισμούς

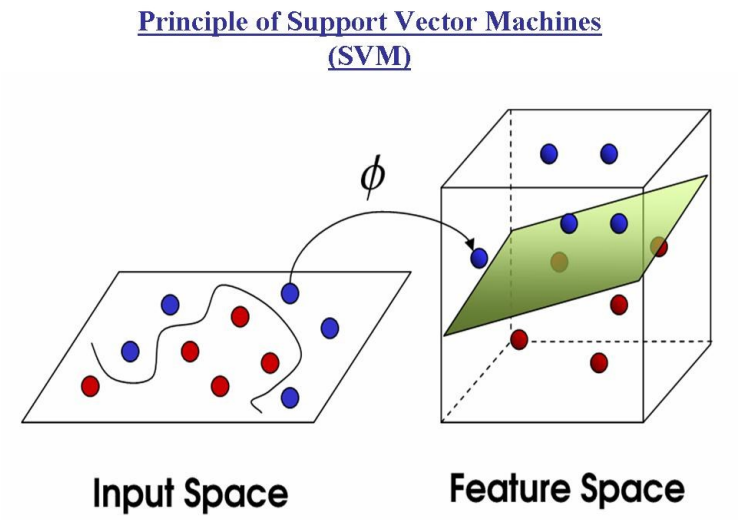
$Y_i(w^T x_i + b) \geq 1$,i=1,2,…,N

Με αυτή τη διαδικασία μεγιστοποιούμε το περιθώριο, ενώ παράλληλα ικανοποιούμε τους περιορισμούς, δηλαδή ότι όλα τα διανύσματα εκπαίδευσης βρίσκονται είτε από τη μία είτε από την άλλη πλευρά του υπερεπιπέδου. Τα διανύσματα που βρίσκονται πάνω στο υπερεπίπεδο είναι τα διανύσματα υποστήριξης ,καθώς υποστηρίζουν τα υπερεπίπεδα και έτσι καθορίζουν την λύση στο πρόβλημα.

Αυτός ο διαχωρισμός ισχύει όταν τα δεδομένα εισόδου είναι γραμμικώς διαχωριζόμενα. Όμως κάποια δεδομένα δεν μπορούν να διαχωριστούν γραμμικά. Στην περίπτωση αυτή υφίσταται μία μέθοδος που ονομάζεται «μετασχηματισμός πυρήνα». Κατά την διαδικασία αυτή, γίνεται μετάβαση από τον αρχικό χώρο X σε έναν άλλο, μεγαλύτερης διάστασης χώρο F ,στον οποίο τα διανύσματα θα χωρίζονται ,πλέον, γραμμικά. Όπως προαναφέρθηκε στο Κεφάλαιο 1.2.3.1 ο γενικός ορισμός του πυρήνα είναι:

$K(x,y)=\varphi(x)^T\varphi(y)$

,όπου φ(x) είναι μία συνάρτηση που μετασχηματίζει τον αρχικό χώρο σε έναν άλλο μεγαλύτερης διάστασης. Έτσι, με την μεταφορά των μη γραμμικών διανυσμάτων εισόδου σε έναν άλλο χώρο, που πιθανών να είναι γραμμικά διαχωρίσιμα μπορούμε πλέον να εφαρμόσουμε τις μεθόδους που εφαρμόζονται στα γραμμικά διαχωρίσιμα διανύσματα. Η διαδικασία είναι η ίδια, με την γραμμική περίπτωση, όμως πλέον γίνεται χρήση του διανύσματος μετασχηματισμού $\varphi(x_i)$ στη θέση του διανύσματος εισόδου x_i.Στο Σχήμα 1θ παρουσιάζεται σχηματικά η αλλαγή του χώρου, με χρήση της συνάρτησης φ(x). [14]



Σχήμα 1θ. Μετάβαση από χώρο X σε μεγαλύτερης διάστασης F.

Στον κώδικα μας , ουσιαστικά, η συνάρτηση run_kernel_svm επιτελεί αυτήν την διαδικασία ,της οποίας η εκτέλεση εξαρτάται από τους πυρήνες και τα χαρακτηριστικά που έχουμε επιλέξει.(Η επεξήγηση της γίνεται στο Κεφάλαιο 1.2.4.2). [15]

1.3 LBP(Local Binary Patterns)

1.3.1 ΕΙΣΑΓΩΓΗ

Τα LBP είναι χαρακτηριστικά τα οποία προτάθηκαν το 1996 από τους Ojala κ.α. και χρησιμοποιούνται έκτοτε ως χαρακτηριστικά ταξινόμησης. Θεωρούνται ισχυρά χαρακτηριστικά υφής. Αποτυπώνουν και κωδικοποιούν τοπικά χαρακτηριστικά των εικόνων και η κατανομή τους χαρακτηρίζει τις εικόνες αυτές.
Σύμφωνα με την αρχική μορφή των LBP,γίνεται χρήση μιας τοπικής μάσκας 3x3.Στη μάσκα αυτή, η φωτεινότητα του κεντρικού εικονοστοιχείου, λαμβάνεται ως κατώφλι Τ, με απόρροια τα γειτονικά εικονοστοιχεία να αποκτήσουν τιμές 1 ή 0 ,αν η φωτεινότητα τους είναι μεγαλύτερη ή μικρότερη του Τα, αντίστοιχα. Ένα παράδειγμα, μπορούμε να δούμε στο Σχήμα 1ι .Το κεντρικό εικονοστοιχείο έχει φωτεινότητα Τ=32.Τα γειτονικά υπόκεινται κατωφλίωση, με βάση την οποία έχουμε την εξαγωγή ενός δυαδικού αριθμού. Ο δυαδικός αριθμός αυτός μετατρέπεται σε δεκαδικό, και ανήκει στο διάστημα [0,255] και εκφράζει την τιμή του χαρακτηριστικού LBP για την συγκεκριμένη μάσκα.

44	56	9
12	32	67
23	68	18

1	1	0
0		1
0	1	0

Σχήμα 1ι.

$(11010100)_2=(148)_{10}$
Τιμή LBP

Τα LBP είναι χαρακτηριστικά τα οποία δεν εξαρτώνται από την μονοτονική αλλαγή των γκρι αποχρώσεων. Δηλαδή, αν μεταβληθούν οι αρχικές φωτεινότητες κατά g_d τότε οι τιμές των LBP χαρακτηριστικών δεν αλλάζουν. Τα χαρακτηριστικά LBP περιγράφουν την χωρική δομή μιας τοπικής υφής .Για το λόγο αυτό τα LBP μπορούν να συνδυαστούν με ένα απλό μέτρο αντίθεσης C το οποίο προκύπτει από την διαφορά της μέσης φωτεινότητας των εικονοστοιχείων που παίρνουν την τιμή 1 με αυτά που παίρνουν την τιμή 0.Για το παραπάνω παράδειγμα (Σχήμα 1ι), έχουμε:

$$C= \frac{(44+56+67+68)}{4} - \frac{(9+18+23+12)}{4} = 43,25$$

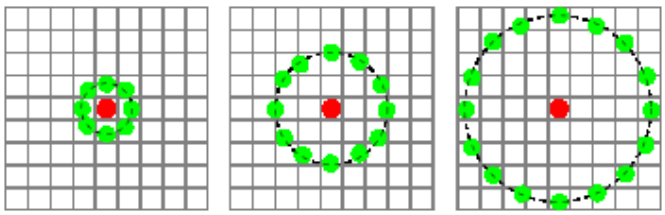
Το 2002 προτάθηκε η γενικευμένη μορφή των LBP. Σύμφωνα με αυτή , ο καθορισμός των τιμών LBP δεν γίνεται με βάση κάποια τετραγωνική μάσκα αλλά στην περιφέρεια ενός κύκλου R από το κεντρικό του εικονοστοιχείο. Πιο συγκεκριμένα, η υφή T εξαρτάται τοπικά από φωτεινότητες P+1 (P>0) εικονοστοιχείων (τα οποία βρίσκονται στην περιφέρεια ενός κύκλου με ακτίνα R). Αυτό εκφράζεται ως:

$$T=texture(g_c,g_0,g_1,\ldots,g_{p-1})$$

,όπου g_c η φωτεινότητα του κεντρικού εικονοστοιχείου και g_i , i=0,1…P-1 οι φωτεινότητες των λοιπών εικονοστοιχείων που κατανέμονται ομοιόμορφα στην περιφέρεια του κύκλου, που έχει ως κέντρο το σημείο (X_c,Y_c) και ακτίνα R.Οι συντεταγμένες των σημείων P δίνονται από τις σχέσεις:

$$x_i= x_c +Rcos(2\pi i/P)$$
$$y_i= y_c -Rsin(2\pi i/P)$$

Ένα παράδειγμα βλέπουμε στο σχήμα 1κ.



Σχήμα 1κ. P=8 R=1 P=12 R=2.5 P=16 R=4

Για την απόκτηση ανεξαρτησίας ως προς την μεταβολή της φωτεινότητας και χωρίς να υφίσταται απώλεια πληροφορίας από τις γειτονικές φωτεινότητες αφαιρούμε την κεντρική.

$$T=texture(g_c,g_0-g_c,g_1-g_c,\ldots,g_{p-1}-g_c)$$

Θεωρούμε τώρα ότι οι διαφορές αυτές είναι ανεξάρτητες της φωτεινότητας του κεντρικού εικονοστοιχείου, όμως στην πράξη δεν ισχύει πάντα αυτό. Λαμβάνοντας υπόψιν μία ελάχιστη απώλεια πληροφορίας, μπορούμε να γράψουμε ότι:

$$T\tilde{}textute(g_c)texture(g_c,g_0-g_c,g_1-g_c,\ldots,g_{p-1}-g_c)$$

Η συνάρτηση textute(g_c) περιγράφει τη συνολική φωτεινότητα της εικόνας η οποία είναι ανεξάρτητη από τη τοπική υφή οπότε και μπορεί να αγνοηθεί. Αρα η παραπάνω σχέση μπορεί να γραφτεί ως εξής:

$T \tilde{} \text{texture}(g_c, g_0-g_c, g_1-g_c, \ldots, g_{p-1}-g_c)$

Μέσα από μαθηματικές και θεωρητικές αποδείξεις, οι οποίες δεν αναλύονται στα πλαίσια την πτυχιακής αυτής εργασίας, καθώς ξεφεύγουν από τη σκοποθεσία της, η παραπάνω σχέση γίνεται:

$LBP_{P,R} = \Sigma\, s(g_i-g_c)2^i$,i[0,P-1]

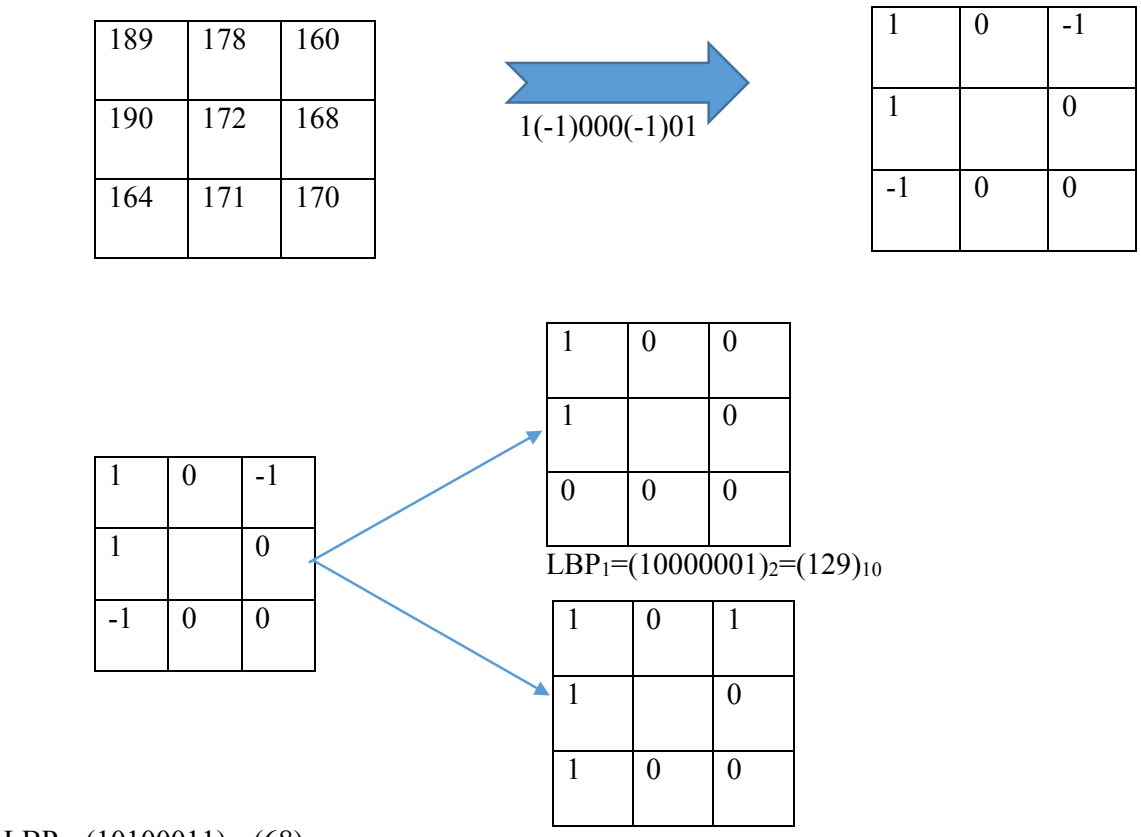
Επίσης, ο υπολογισμός των $LBP_{P,R}$ χαρακτηριστικών αφορά μόνο τα στοιχεία που βρίσκονται εντός του κύκλου με ακτίνα R.
Τα χαρακτηριστικά LBP παρέχουν ανθεκτικότητα σε διαταραχές του φωτισμού καθώς είναι αμετάβλητα σε μονοτονικούς μετασχηματισμούς φωτεινότητας. Επιπρόσθετα, έχει αποδειχθεί ότι τα LBP έχουν υψηλή διακριτική δύναμη για την ταξινόμηση της υφής των εικόνων. Παρόλα αυτά έχουμε μεγάλη ευαισθησία στο θόρυβο , ιδιαίτερα σε ομοιόμορφες χρωματικά περιοχές της εικόνας (όπως ο ουρανός ή το δέρμα του προσώπου),καθώς η κατωφλίωση είναι βασισμένη στο κεντρικό εικονοστοιχείο. Για την επίλυση του προβλήματος αυτού, έχουμε την επέκταση των LBP, ώστε να προκύπτουν τοπικά τρεις τιμές έναντι της μίας που είχαμε μέχρι τώρα. Η ονομασία αυτών: LTP (Local Ternary Patterns).
Στην περίπτωση των LTP χρησιμοποιούμε ως κατώφλι για κάθε ένα εικονοστοιχείο τρεις τιμές που προκύπτουν από την παρακάτω εξίσωση:

1, $I(x,y) \geq I(x+i,y+j)+T$
$T_{code}(x+I,y+j)$= 0, $|I(x,y)-I(x+i,y+j) < T$
-1, $I(x,y) \leq I(x+I,y+j)-T$ i= -1,0,1 y= -1,0,1

,όπου

T= κατώφλι (συνήθως <20)
I(x,y)= η τιμή της φωτεινότητας στο εκάστοτε εικονοστοιχείο (x,y)
T_{code} = ο κώδικας που προκύπτει.

Ένα παράδειγμα βλέπουμε παρακάτω, όπου T=8.Οπως είναι ορατό μετά την κατωφλίωση έχουμε τρεις τιμές. -1,0,1.Έτσι δεν γίνεται η άμεση μετατροπή του δυαδικού κώδικα που προκύπτει στον αντίστοιχο δεκαδικό. Διασπάται, λοιπόν ο κώδικας αυτός σε δύο επιμέρους LBP κώδικες όπου στον ένα τα ψηφία (-1) μετατρέπονται σε 0 και στον άλλο μετατρέπονται σε 1. [16]



Σχήμα 1λ.Παραγωγή LTP και διάσπαση LTP χαρακτηριστικών σε LBP.

1.3.2 ΥΛΟΠΟΙΗΣΗ LBP ΣΕ MATLAB

1.3.2.1 ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Παρακάτω βλέπουμε και αναλύουμε τον κώδικα που λάβαμε από την σελίδα του MIT και αποτετελεί με τον κώδικα LBP που χρησιμοποιήθηκε στην πτυχιακή αυτή εργασία.

Αρχικά λοιπόν, στρέφουμε την προσοχή μας στην συνάρτηση lbp_feature η οποία βρίσκεται στο \source\code\scene_sun\features και ο κώδικας της φαίνεται παρακάτω:

```
function [feat boxes] = lbp_feature(conf, im)

im1=imresize(im,[64 64]);
if(size(im1,3) < 3)
    %black and white image
    im = cat(3, im1, im1, im1); %make it a trivial color image
end
boxes = [1;1;size(im1,1);size(im1,2)];

%imshow(im1)

[lbp_L0 lbp_L1 lbp_L2] = lbp_original_4x4(im1);

feat.hists.L0 = lbp_L0;
feat.hists.L1 = lbp_L1;
feat.hists.L2 = lbp_L2;
% pause;
```

Βλέπουμε πως παίρνει δύο ορίσματα. Το βοηθητικό conf και την εκάστοτε εικόνα κάθε φορά. Γίνεται επαναπροσδιορισμός της εικόνας σε 64x64 και αν το πεδίο ορισμού των αποχρώσεων της εικόνας είναι μονόχρωμο ,δημιουργεί μία νέα εικόνα, βασισμένη στην αρχική αλλά με 3 κανάλια. Ακολουθεί κλήση της lbp_original_4x4 η οποία δέχεται ως όρισμα την νέα εικόνα και εξάγει 3 τιμές L0,L1 και L2 οι οποίες αποθηκεύονται σε έναν πίνακα για την εκάστοτε εικόνα στο F:\source\data\scene_Xclass\feature\Y , όπου Y ο φάκελος που αποθηκεύονται τα χαρακτηριστικά. Όσον αφορά την κλήση της συνάρτησης αυτής γίνεται μέσα από την extract_feature η οποία με την σειρά της έχει κληθεί από την compute_feature.

Κοιτώντας την lbp_original_4x4,βλέπουμε ότι περιέχει αναθέσεις της μορφής:

```
feature_L0(:,1) = lbp_original(I);
fL = length(feature_L0);

h = size(I,1);  w = size(I,2);

feature_L1(1     :  fL,1) =
lbp_original(I(1:round(h/2),1:round(w/2)));

…

feature_L2( 1     :   fL,1) = lbp_original(I(1:round(h/4),
1:round(w/4)));
feature_L2( fL+1  : 2*fL,1) = lbp_original(I(1:round(h/4),
round(w/4)+1:round(w/2)));
feature_L2( 2*fL+1: 3*fL,1) = lbp_original(I(1:round(h/4),
round(w/2)+1:round(w/4*3)));
feature_L2( 3*fL+1: 4*fL,1) = lbp_original(I(1:round(h/4),
round(w/4*3)+1:end));


…
```

,όπου h είναι το σύνολο των γραμμών και w το σύνολο των στηλών της εικόνας. Για την εξαγωγή των L0,L1,L2 γίνεται κλήση της lbp_original η οποία δέχεται ως όρισμα ένα κομμάτι της εικόνας (αυτό για το οποίο υπολογίζουμε το εκάστοτε lbp κάθε φορά).Στο L0 έχουμε το lbp της συνολικής εικόνας, στο L1 σπάμε την εικόνα σε 4 «κομμάτια» υπολογίζουμε το lbp σε αυτά ενώ στο L2 χωρίζουμε την εικόνα σε 16 «κομμάτια» και αντίστοιχα υπολογίζουμε το lbp.

Στρέφοντας την προσοχή στην lbp_original βλέπουμε ότι καλεί δύο συναρτήσεις, την getmapping και την lbp.Η getmapping δέχεται ως ορίσματα τον αριθμό των σημείων P που επιλέχθηκαν καθώς και τον τύπο χαρτογράφησης και επιστρέφει μία δομή που περιέχει τον πίνακα χαρτογράφησης για το lbp σε μια γειτονιά από σημεία P.Τέλος με την κλήση της συνάρτησης lbp (η οποία μπορεί να πάρει μεταβλητό αριθμό ορισμάτων) έχουμε τον υπολογισμό των lbp για το κάθε «κομμάτι» εικόνας.

Εκτέλεση του κώδικα αυτού για εξαγωγή lbp χαρακτηριστικών γίνεται μέσα από την συνάρτηση compute_feature με την κατάλληλη ρύθμιση των παραμέτρων και τα αποτελέσματα αποθηκεύονται ως πίνακες (αρχεία .mat) ανά κατηγορία στο φάκελο source\data\scene_Xclass\feature\lbp .

1.3.3 ΕΚΤΕΛΕΣΗ LBP ΣΤΙΣ 15 ΚΛΑΣΕΙΣ (ΚΑΤΗΓΟΡΙΕΣ) ΕΙΚΟΝΩΝ

Το αμέσως επόμενο βήμα ήταν η εκτέλεση του SVM για την εξαγωγή χαρακτηριστικών LBP στις 15 κλάσεις εικόνων. Η διαδικασία που ακολουθήθηκε ήταν να εξαγάγουμε αρχικά τα χαρακτηριστικά με την συνάρτηση compute_feature με την διαδικασία που περιγράφεται στο προηγούμενο Κεφάλαιο 1.2.4.1 και ύστερα, στην συνάρτηση run_kernel_svm να διαλέξουμε τις 15 κλάσεις εικόνων καθώς και το LBP με τους πυρήνες που είχαν οριστεί εξ αρχής.

Τα αποτελέσματα ήταν αρκετά υψηλά, όπως φαίνονται παρακάτω, και αυτός είναι ένας από τους λόγους που το LBP αποτελούσε μέρος σχεδόν κάθε πειραματισμού μας, σε συνδυασμό με άλλες μεθόδους που περιγράφονται σε επόμενο Κεφάλαιο, όπως τα zero crossings, τα wavelets , τα sign slope changes κ.ο.κ.

Συμπερασματικά, μπορούμε να πούμε πως ο κώδικας του LBP είναι ένας σχετικά «βαρύς» κώδικας για να εκτελεστεί σε έναν απλό ιδιωτικό υπολογιστή, καθώς μπορεί να διαρκέσει αρκετές ώρες. Για την εκτέλεση του χρειάστηκε η δέσμευση ενός ισχυρότερου επεξεργαστικά υπολογιστή, ο οποίος μας δόθηκε από το Τμήμα Πτυχιακών του ΑΤΕΙ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ.

Παρά ταύτα , αποτελεί έναν από τους ισχυρότερους αλγορίθμους υφής για αναγνώριση και ταξινόμηση εικόνων, και αυτό είναι εμφανές και από τα αποτελέσματα του.

1.3.4 ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Όπως αναφέρθηκε και στο Κεφάλαιο 1.2.4.2.2 η συνάρτηση run_kernel_svm βγάζει αποτελέσματα: α) στην γραμμή εντολών του Matlab όπου είναι τα αποτελέσματα που προκύπτουν κατά την διάρκεια εκτέλεσης του κώδικα, β) στο workspace στο οποίο έχουμε τις μεταβλητές γ) τους φακέλους kernel και result και τέλος δ) μία γραφική παράσταση που δείχνει την απόδοση του SVM κατά την διάρκεια της εκτέλεσης.

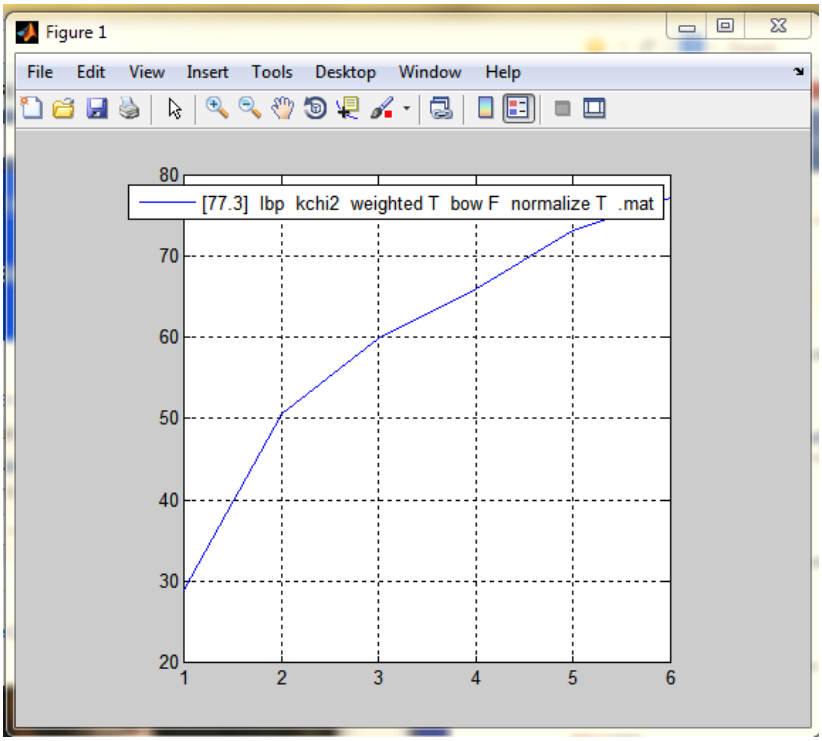
Εστιάζοντας στο LBP και χρησιμοποιώντας τα weighted και normalized ως αληθή βλέπουμε τα εξής αποτελέσματα:

Α) Κατά την διάρκεια του πρώτου μέρους της εκτέλεσης ,όπου έχουμε την εκπαίδευση του SVM με τους n πυρήνες που έχουμε επιλέξει ξεχωριστά, παρατηρούμε ότι τα αποτελέσματα κυμαίνονται από περίπου 27% για εκπαίδευση SVM με μία εικόνα ενώ όσο σταδιακά ανεβάζουμε τον αριθμό των εικόνων εκπαίδευσης το ποσοστό αυτό ανεβαίνει περίπου στο 77%.

Β)Στρέφοντας την προσοχή μας στο δεύτερο μέρος της εκπαίδευσης, όπου γίνεται συνδυασμός των πυρήνων, η εκπαίδευση με χρήση μιας εικόνας η απόδοση φτάνει περίπου στο 13% με 14% ενώ όσο ανεβάζουμε τον αριθμό εικόνων εκπαίδευσης η απόδοση ανεβαίνει κοντά στο 32%.Όπως είναι ορατό, ο συνδυασμός των πυρήνων δεν είναι αποδοτικός για το LBP.

Γ) Η γραφική παράσταση δημιουργείται βασισμένη στη μέση απόδοση της ατομικής εκτέλεσης των πυρήνων, καθότι είναι και η πιο αποδοτική, όπως είναι εμφανές και από τα αποτελέσματα.

Παρακάτω βλέπουμε την γραφική παράσταση:



1.4 TINY IMAGES

1.4.1 ΕΙΣΑΓΩΓΗ

Η βασική ιδέα πίσω από τον αλγόριθμο αυτό είναι ο επαναπροσδιορισμός του μεγέθους των εικόνων σε τετραγωνικές εικόνες 32x32,με τις οποίες εκπαιδεύουμε και δοκιμάζουμε το SVM. Ένα από τα ερωτήματα που γεννιέται είναι για ποιο λόγο να χάνουμε αρκετή και πιθανών χρήσιμη πληροφορία από την εικόνα και πως αυτό μπορεί να αποτελέσει αποδοτικό παράγοντα για το SVM.Ουσιαστικά, με το μέθοδο αυτή αφαιρούμε μη επιθυμητή πληροφορία από την εικόνα και κρατάμε το κεντρικό της θέμα, το οποίο είναι αυτό που επικρατεί στο μεγαλύτερο μέρος της εικόνας.

1.4.2 ΥΛΟΠΟΙΗΣΗ TINY IMAGES ΣΕ MATLAB

Αρχικά, βρίσκουμε μία συνάρτηση παρόμοια με αυτή των lbp στον φάκελο F:\source\code\scene_sun\features που ονομάζεται tiny_images και βλέπουμε τον κώδικα της παρακάτω:

```
…

 tiny_image = single(imresize(current_img, [64 64]));

 tiny_image = tiny_image(:);
 feat.tiny_image = tiny_image;
 boxes = [1;1;64;64];
```

Όπως και η συνάρτηση feature_lbp ,όπως όλες οι συναρτήσεις που καλεί η extract_feature τις οποίες θα δούμε παρακάτω έτσι και η tiny_images δέχεται δύο ορίσματα το conf και την τρέχουμε εικόνα και επιστρέφει το χαρακτηριστικό και την μεταβλητή boxes.Η διαδικασία προετοιμασίας της εικόνας για πέρασμα από το tiny_images είναι η ίδια με πριν.Τέλος μειώνουμε το μέγεθος της σε 64x64 (οι διαστάσεις που έδιναν από το MIT ήταν 32x32,απλα εμείς πειραματικά θελήσαμε να έχουμε λίγη παραπάνω πληροφορία).Η συνάρτηση αυτή είναι πάρα πολύ απλή και γρήγορη στην εκτέλεση της καθώς δεν έχει να επιτελέσει κάποιο άλλο έργο πέρα από την μείωση του μεγέθους της εικόνας. Οι εικόνες αυτές αποθηκεύονται ως αρχεία .mat ανά κατηγορία στο source\data\scene_Xclass\feature\tiny_image .

1.4.3 ΕΚΤΕΛΕΣΗ TINY IMAGES ΣΤΙΣ 15 ΚΛΑΣΕΙΣ (ΚΑΤΗΓΟΡΙΕΣ) ΕΙΚΟΝΩΝ

Αρχικά, στην πρώτη επαφή με το SVM σε μορφή Matlab όπως το πήραμε από το MIT ,αναζητήσαμε τον πιο απλό αλλά και γρήγορο αλγόριθμο εξαγωγής χαρακτηριστικών, ώστε να κατανοήσουμε την δομή του κώδικα και ως προς τις αλλαγές που απαιτούνται για την ολοκληρωμένη εκτέλεση του αλλά και ως προς την ανάλυση των αποτελεσμάτων που προκύπτουν.

Η προσοχή μας, λοιπόν, στράφηκε στον αλγόριθμο tiny_images, όπου αποτελεί έναν αλγόριθμό γρήγορο, σχετικά αποδοτικό αλλά και με εύκολη λογική και δομή. Φυσικά, ο tiny_images αποτελεί ακρογωνιαίο λίθο σε πολλές εφαρμογές αναγνώρισης εικόνας αλλά συγκρίνοντας αργότερα την απόδοση του με αλγορίθμους όπως ο LBP , δεν τον συμπεριλάβαμε στους μεταγενέστερους πειραματισμούς μας. Όμως, καθώς αποτελεί ένα πάρα πολύ καλό και βοηθητικό αλγόριθμο για να ξεκινήσει κάποιος την ενασχόληση του με την αναγνώριση και ταξινόμηση εικόνων με τη χρήση SVM θεωρείται αναγκαία η επεξήγηση της εκτέλεσης και των αποτελεσμάτων του.

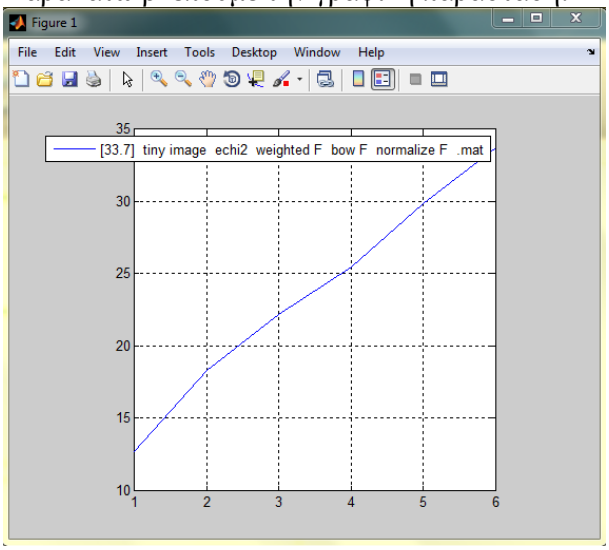
1.4.4 ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Η βάση δεδομένων που επιλέχτηκε για την εκτέλεση του tiny_images ήταν εκείνη των 15 κλάσεων, καθώς η διαφορά με τη βάση δεδομένων των 8 κλάσεων ήταν πολύ μικρή και η βάση δεδομένων της SUN των 397 κλάσεων ήταν πολύ μεγάλη και απαιτούσε μεγάλο αριθμό επεξεργαστικών πόρων.

Ο πυρήνας που επιλέχθηκε ήταν το echi2, διότι μετά από πειραματισμούς με άλλους πυρήνες , έδωσε τα καλύτερα αποτελέσματα.

Στο Α μέρος, όπου η εκτέλεση γίνεται ατομικά για κάθε επιλεχθέντα πυρήνα, η εκπαίδευση με χρήση μίας εικόνας αποδίδει περίπου 12% ενώ όσο αυξάνεται ο αριθμός των εικόνων εκπαίδευσης η απόδοση φτάνει μέχρι και το 34%.

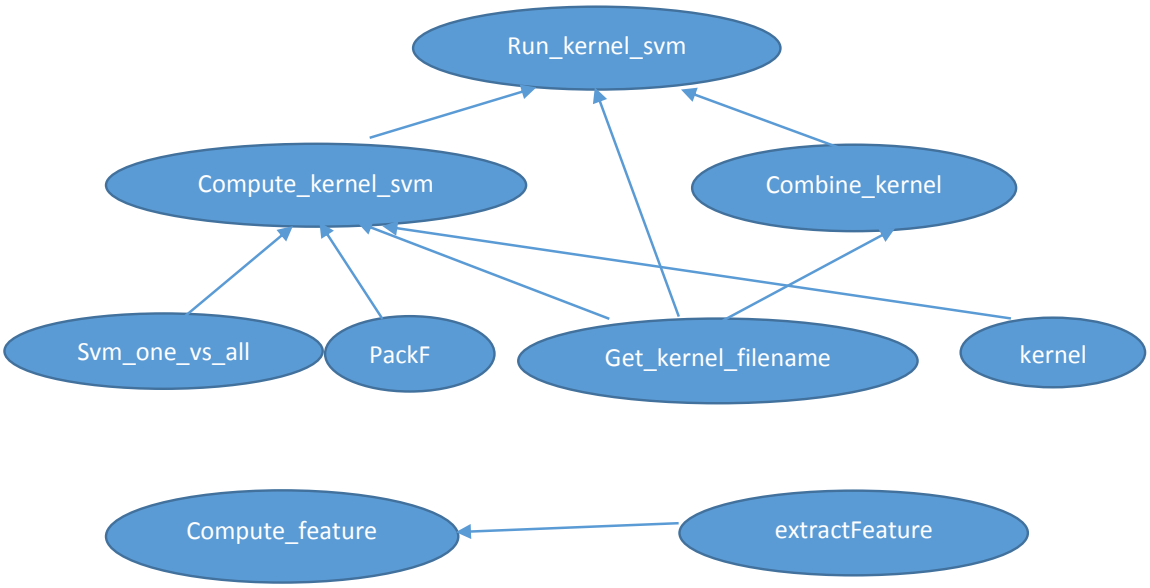
Στο Β μέρος, όπου συνδυάζονται οι πυρήνες τα αποτελέσματα δεν παρεκκλίνουν καθόλου. Η μέση απόδοση και για το Α και για το Β είναι 33,7%.Οπότε αυτό το ποσοστό χρησιμοποιείται και για την εξαγωγή της γραφικής παράστασης. Παρακάτω βλέπουμε την γραφική παράσταση:



1.5 ΑΝΑΛΥΣΗ ΥΠΟΛΟΙΠΩΝ ΣΥΝΑΡΤΗΣΕΩΝ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΟΥ ΚΩΔΙΚΑ

1.5.1 ΠΩΣ ΣΥΝΔΕΟΝΤΑΙ ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ ΜΕΤΑΞΥ ΤΟΥΣ

Εξετάζοντας τον κώδικα και παρατηρώντας την πολυπλοκότητα του κρίνεται σκόπιμη η επεξήγηση της διασύνδεσης, των βασικών συναρτήσεων που χρησιμοποιούνται, (ποια εκτελείται, ποια κάνει κλήση σε ποια κ.ο.κ),η οποία θα παρουσιαστεί σχηματικά για την διευκόλυνση του αναγνώστη.



Σχήμα 1μ.Ιεραρχική κατάταξη κλήσεων συναρτήσεων

Τα βελάκια ακολουθούν την φορά της κλήσης. Λ.χ η compute_feature καλεί την extractFeature.Φυσικά, η δομή του κώδικα δεν είναι τόσο απλή. Υπάρχουν εκατοντάδες συναρτήσεις στον συνολικό κώδικα που χρησιμοποιήσαμε, όμως δεν είναι αναγκαία η επεξήγηση τους ώστε να μπορεί κάποιος να επεξεργαστεί και να εισάγει νέα χαρακτηριστικά στον ήδη υπάρχων κώδικα, όπου είναι και ο σκοπός της πτυχιακής αυτής.
Παρακάτω, πριν προχωρήσουμε στο επόμενο Κεφάλαιο γίνεται μία περιληπτική επεξήγηση κάθε μίας από τις παραπάνω συναρτήσεις του σχήματος ώστε να δοθεί μία βασική ιδέα για την χρήση και την λειτουργικότητα της εκάστοτε συνάρτησης.

1.5.2 ΤΙ ΕΡΓΑΣΙΑ ΕΠΙΤΕΛΕΙ Η ΕΚΑΣΤΟΤΕ ΣΥΝΑΡΤΗΣΗ

Σε αυτό το σημείο γίνεται μία περιληπτική επεξήγηση της λειτουργικότητας της κάθε συνάρτησης.

- run_kernel_svm: Όπως αναλύθηκε και στο Κεφάλαιο 1.2.4.2 η συνάρτηση αυτή χρησιμεύει για την εκπαίδευση και δοκιμή του SVM με την χρήση πυρήνων, και ουσιαστικά μας δίνει τα τελικά αποτελέσματα.
- compute_feature: Η συνάρτηση αυτή επίσης έχει αναλυθεί σε προηγούμενο Κεφάλαιο (βλ. 1.2.4.1), και χρησιμοποιείται για να παραχθούν τα χαρακτηριστικά των εικόνων που κρατάμε στη βάση δεδομένων
- compute_kernel_svm: Εδώ γίνεται η φόρτωση των πακέτων για εκπαίδευση και δοκιμή του svm καθώς και η ίδια η εκπαίδευση του svm
- svm_one_vs_all: Ακόμα μία συνάρτηση η οποία καλείται από την compute_kernel_svm και συμβάλει στην εκπαίδευση αλλά και την δοκιμή μετά την εκπαίδευση του svm.
- get_kernel_filename: Επιστρέφει το όνομα του πυρήνα/πυρήνων που έχουμε επιλέξει
- kernel: Η συνάρτηση αυτή, περιέχει την μαθηματική δομή του κάθε πυρήνα.
- packF: Η συνάρτηση αυτή, ανάλογα τον πυρήνα, την κανονικοποίηση αλλά και το weighted και το bow (αν έχουν τιμές true ή false) «πακετάρει» τα χαρακτηριστικά για αποθήκευση με μία συγκεκριμένη κατάταξη στη μεταβλητή X.
- combine_kernel: Όπως περιγράψαμε στο Κεφάλαιο 1.2.4.2 γίνεται ένας συνδυασμός των επιλεγμένων πυρήνων για τα εκάστοτε χαρακτηριστικά. Η συνάρτηση αυτή επιτελεί όλες τις απαραίτητες λειτουργίες πάνω στο έργο αυτό.
- extractFeature: Η συνάρτηση αυτή καλείται, όπως φαίνεται και στο Σχήμα 1μ από την compute_feature και κάνει όλες τις εργασίες που χρειάζονται για να εξαχθούν τα χαρακτηριστικά.

ΚΕΦΑΛΑΙΟ 2: ΧΡΗΣΗ ΚΑΙ ΕΠΕΞΗΓΗΣΗ ΔΙΑΦΟΡΕΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ ΣΥΝΔΙΑΣΜΕΝΩΝ ΜΕ ΤΟΥΣ ΗΔΗ ΥΠΑΡΧΟΝΤΕΣ

2.1 ΕΙΣΑΓΩΓΗ

Αφότου έγινε η αναλυτική επεξήγηση του βασικού κώδικα καθώς και των αποτελεσμάτων του, στο Κεφάλαιο 1,πλέον,στο δεύτερο αυτό Κεφάλαιο παρουσιάζεται ο τρόπος με τον οποίο κάποιος μπορεί να επέμβει στον αρχικό κώδικα, να προσθέσει βάσεις δεδομένων καθώς και δικά του χαρακτηριστικά, προσαρμοσμένα στον τρόπο και την λογική εκτέλεσης του αρχικού κώδικα. Επιπρόσθετα, γίνεται η επεξήγηση των αλγορίθμων των νέων χαρακτηριστικών που εξάγονται από τις εικόνες.

2.2 ΜΕΤΑΤΡΟΠΗ ΥΠΑΡΧΟΝΤΟΣ ΚΩΔΙΚΑ ΚΑΙ ΠΕΙΡΑΜΑΤΙΣΜΟΙ ΓΙΑ ΤΗΝ ΑΥΞΗΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΤΟΥ MIT

2.2.1 ΛΟΓΟΙ ΕΝΑΣΧΟΛΗΣΗΣ ΜΕ ΤΗΝ ΜΕΤΑΤΡΟΠΗ ΤΟΥ ΥΠΑΡΧΟΝΤΟΣ ΚΩΔΙΚΑ

Είναι εύκολο και εύλογο να αναρωτηθεί κάποιος τους λόγους της ενασχόλησης με την μετατροπή αλλά και με την προσπάθεια βελτίωσης ενός κώδικα/αλγόριθμου τόσο πολύπλοκου και δύσκολου όπως αυτός του MIT.Όμως, η πολυπλοκότητα αλλά και η χρησιμότητα του είναι ένα από τα πιο ενδιαφέροντα σημεία του.

Πρωτίστως , η γνώση που επέρχεται από την μελέτη της λογικής αλλά και της δομής του κώδικα αυτού, είναι κολοσσιαία και μπορεί να χρησιμοποιηθεί σε πολλές άλλες έρευνες, καθώς μιλάμε για ευρέος χρησιμοποιούμενες επιστημονικές μαθηματικές μεθόδους, όπως η χρήση SVM ή τα LBP.

Ακόμη, η αναγνώριση και ταξινόμηση εικόνων μπορεί να επεκταθεί και σε αναγνώριση βίντεο ή ήχου, με απόρροια την ανάπτυξη πολλών ιατρικών συστημάτων αλλά και συστημάτων ασφάλειας.

Τέλος, αποτελεί ένα πάρα πολύ ενδιαφέρον θέμα, όπου , σε κάθε προσπάθεια μελέτης του, ανακαλύπτονται καινούριες πτυχές του, κατά συνέπεια προκύπτουν και ,καινούριες ιδέες αξιοποίησης και μετατροπής , καθώς μιλάμε για έναν πάρα πολύ μεγάλο κώδικα, που χρειάστηκε πολλά έτη εργασίας από πολλούς και καταξιωμένους επιστήμονες.

2.2.2 ΑΝΑΛΥΤΙΚΗ ΕΠΕΞΗΓΗΣΗ ΒΗΜΑΤΩΝ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΑΚΟΛΟΥΘΗΘΟΥΝ ΓΙΑ ΤΗΝ ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΩΝ ΣΤΟΝ ΚΩΔΙΚΑ

Πριν την παρουσίαση των νέων χαρακτηριστικών, κρίνεται απαραίτητη η επεξήγηση της διαδικασίας εισαγωγής και προσαρμογής τους, στον αρχικό κώδικα .Ουσιαστικά, ύστερα από μελέτη της δομής των αλγορίθμων, προσαρμόσαμε την εκτέλεση εξαγωγής χαρακτηριστικών ώστε να μην διαφέρει σε τίποτα με την εκτέλεση των χαρακτηριστικών όπως περιγράφηκε στο προηγούμενο Κεφάλαιο.

Σχετικά με την εισαγωγή μίας νέας βάσης δεδομένων, η διαδικασία είναι αρκετά απλουστευμένη. Ως πρώτο βήμα, είναι η εύρεση ή η δημιουργία μίας νέας βάσης δεδομένων. Αφού γίνει αυτό, πρέπει να αποθηκευτεί η βάση στον προορισμό \source\data . Ο φάκελος της βάσης πρέπει να πληροί την δομή που περιγράφεται στο Κεφάλαιο 1.1.1 περί αρχειοθέτησης του κώδικα. Ύστερα, πρέπει να οριστούν οι κατάλληλες για χρήση πληροφορίες εντός του κώδικα ,δηλαδή οι διαδρομές προορισμού για τις εικόνες και την αποθήκευση των χαρακτηριστικών, των πυρήνων αλλά και των τελικών αποτελεσμάτων. Τέλος, καταληκτικό βήμα η επιλογή της βάσης αυτής για εκτέλεση.

Έστω τώρα ότι θέλουμε να αναπτύξουμε έναν αλγόριθμο εξαγωγής χαρακτηριστικών με όνομα algoX πάνω στον ήδη υπάρχον κώδικα. Αρχικά, σκεφτόμαστε και σχεδιάζουμε ποιες τεχνικές θέλουμε να χρησιμοποιήσουμε για τον αλγόριθμο αυτό, λ.χ. lbp πάνω σε wavelet. Εφόσον έχουμε μελετήσει τον αλγόριθμο και τις παραμέτρους εισαγωγής και εξαγωγής του ξεκινάμε την υλοποίησή του.

Α)Το πρώτο βήμα για την υλοποίηση του θεωρητικού παραδείγματος algoX είναι να μεταβούμε στο φάκελο \source\code\scene_sun\features και να αναπτύξουμε μία συνάρτηση της παρακάτω μορφής:

```
function [feat boxes] = algoX_feature(conf, im)

if(size(im1,3) < 3)
        im = cat(3, im1, im1, im1); %make it a trivial color image
end
boxes = [1;1;size(im1,1);size(im1,2)];

%imshow(im1)

[feature_output feature_output2] = algoX(im1);

feat.hists.algo_out = feature_output;
feat.hists.algo_out2 = feature_output2;

% pause;
```

Όπως βλέπουμε παραπάνω, κάθε συνάρτηση αυτής της μορφής οφείλει να έχει δύο μεταβλητές εξόδου την feat και το boxes. Η feat χρησιμοποιείται για την αποθήκευση των χαρακτηριστικών της εκάστοτε εικόνας ενώ το boxes κρατάει τις διαστάσεις τις εικόνας. Τα ορίσματα εισόδου πρέπει να είναι το βοηθητικό conf αλλά και η εικόνα im. Φυσικά να σημειωθεί ότι ανάλογα τον αλγόριθμο μπορεί να αλλάξουν τα ορίσματα εισόδου. Εμείς ,όμως ,χρησιμοποιήσαμε την παραπάνω δομή. Στα πλαίσια της επεξεργασίας είναι η αλλαγή της εικόνας σε τριών καναλιών ή σε ασπρόμαυρη, ή όποια άλλη επεξεργασία κρίνεται απαραίτητη πριν την εξαγωγή των χαρακτηριστικών καθώς και η κλήση του αλγορίθμου που κάνει την εξαγωγή των χαρακτηριστικών. Τέλος, αποθηκεύουμε τα χαρακτηριστικά, της κάθε εικόνας, στη δομή hists.

Ας εστιάσουμε τώρα στην κλήση του algoX όπου γίνεται με έναν μερικώς, ιδιαίτερο τρόπο. Στο \source\code\scene_sun\features υφίστανται φάκελοι που περιέχουν τους διάφορους κώδικες των χαρακτηριστικών. Δημιουργούμε, λοιπόν και εμείς έναν φάκελο με το όνομα algoX.Μέσα στον φάκελο αυτό δημιουργούμε την

συνάρτηση μας algoX και ορίζουμε την εργασία που επιτελεί. Η συνάρτηση αυτή μπορεί να χρησιμοποιεί και άλλες συναρτήσεις οι οποίες είναι αποθηκευμένες στον ίδιο φάκελο. Εδώ πρέπει να εστιάσουμε την προσοχή μας. Έστω, παραδείγματος χάριν, ότι έχουμε δημιουργήσει αρκετούς διαφορετικούς φακέλους με αρκετά διαφορετικά χαρακτηριστικά. Σε καμία περίπτωση δεν πρέπει δύο η περισσότερες συναρτήσεις (ακόμα και οι βοηθητικές) να έχουν το ίδιο όνομα, ακόμα και να βρίσκονται σε διαφορετικούς φακέλους. Ο κώδικας δεν θα διαβάσει την συνάρτηση που βρίσκεται στο φάκελο που θέλουμε αλλά την πρώτη συνάρτηση που θα συναντήσει με πορεία αναζήτησης από πάνω μπρος τα κάτω. Το πιο πιθανό είναι ότι κάθε φορά αναζητάει την συνάρτηση που καλείται σε όλους τους φακέλους, καθώς ήταν ένα από τα προβλήματα που αντιμετωπίστηκαν κατά την διάρκεια των πειραματισμών μας.
Αφότου ολοκληρώσουμε τον κώδικα του algoX , πρέπει να προσαρμόσουμε τις βασικές συναρτήσεις ώστε να τον αναγνωρίζουν και να τον εκτελούν. Πάμε , λοιπόν, στο φάκελο \source\code\scene_sun . Ανοίγουμε το αρχείο packF. Εδώ, παρατηρούμε πολλές μαθηματικές συσχετίσεις και δομές. Εμείς, επιλέξαμε για όλα τα χαρακτηριστικά την πιο απλή, η οποία είχε την παρακάτω δομή:

```
case 'algoX'
  X(:,i) = [F{i}.hists.algo_out ; F{i}.hists.algo_out2];
```

Χρειάζεται προσοχή καθώς τα ονόματα των μεταβλητών που αποθηκεύονται πρέπει να είναι ίδια με αυτά που εξάγονται στην συνάρτηση algoX_feature.Τα X(:,i) και F{i} είναι σταθερά και δεν αλλάζουν. Γενικεύοντας την παραπάνω πρόταση έχουμε:

```
case 'ονομα αλγορίθμου'
  X(:,i) = [F{i}.struct.var ; F{i}.struct.var2 …];
```

Αφού ολοκληρώσουμε και με την packF σειρά έχει η compute_feature.Οι αλλαγές που πρέπει να γίνουν εδώ είναι σχετικά μικρές, και έχουν προσαρμοστικό χαρακτήρα. Αρχικά, διαλέγουμε την βάση δεδομένων από την οποία θα εξαχθούν τα χαρακτηριστικά. Στην συνέχεια, προσθέτουμε το όνομα του χαρακτηριστικού μας στην μεταβλητή conf.featNames ως εξής:

```
conf.featNames    = {'algoX'} ;
```

Προχωρώντας παρακάτω, στο σκέλος όπου ρυθμίζουμε τα χαρακτηριστικά, προσθέτουμε ένα μέρος κώδικα όπως αυτό:

```
conf.algoX.extractFn = @algoX_feature ;
conf.algoX.outside_quantization = false;
```

Όπου ρυθμίζονται οι βασικές παράμετροι για την εκτέλεση της συνάρτησης algoX_feature.Φυσικά αναλόγως την πολυπλοκότητα του αλγορίθμου που αναπτύσσεται, μπορεί να κριθεί αναγκαία η ρύθμιση περισσοτέρων παραμέτρων. Όσοι

αλγόριθμοι αναπτύχθηκαν από εμάς στα πλαίσια της πτυχιακής αυτής , δεν χρήζουν ρύθμισης περισσοτέρων παραμέτρων, πέρα από αυτές τις δύο.
Έτσι, ολοκληρώθηκε η ρύθμιση της πρώτης βασικής συνάρτησης για την εξαγωγή των χαρακτηριστικών. Μετά την πραγμάτωση της εργασίας αυτής, στρεφόμαστε προς την run_kernel_svm, όπου θα γίνει η εκπαίδευση και δοκιμή του SVM με βάση τα χαρακτηριστικά που μόλις δημιουργήθηκαν.

Στη συνάρτηση αυτή, η μόνη διαδικασία που απαιτείται είναι η εισαγωγή ενός κώδικα όπου θα καθορίζουμε το χαρακτηριστικό με βάση το οποίο θα γίνει η εκπαίδευση αλλά και τον πυρήνα.

```
n=n+1;
parameterCell{n}.featureName  = 'algoX;
parameterCell{n}.kernelName   = 'echi2';
parameterCell{n}.weighted     = false;
parameterCell{n}.bow          = false;
parameterCell{n}.normalize    = false;
```

Σε κάθε μας δοκιμή τα weighted,bow και normalize τα θέταμε ως false, καθώς στην packF δεν ορίσαμε μαθηματική δομή για αυτές τις τρεις παραμέτρους. Θα μπορούσε όμως να γίνει. Απλά θα απαιτούσε την εισαγωγή περισσότερων ελέγχων στον κώδικα στην περίπτωση του algoX.

Καταληκτικό βήμα η εκτέλεση του κώδικα και η παρακολούθηση των αποτελεσμάτων.

2.2.3 WAVELETS

2.2.3.1 ΤΙ ΕΙΝΑΙ ΤΑ WAVELETS: ΟΡΙΣΜΟΣ

Η αναζήτηση όλο και καλύτερων πληροφοριών μη περιοδικών σημάτων με απότομες μεταβολές οδήγησε στην ανάπτυξη συναρτήσεων οι οποίες είναι σχεδιασμένες ώστε η κάθε μία από αυτές να αντιστοιχεί σε συγκεκριμένο επίπεδο ανάλυσης, άρα, κατά συνέπεια, και σε κάποιες συγκεκριμένες συχνότητες.
Αρχικά υπήρξε η ανάλυση του μετασχηματισμού Fourier,όπου το σήμα αναλύεται μέσω ημιτονοειδών συναρτήσεων, διαφορετικών συχνοτήτων και έχουμε την μεταφορά του σήματος από το επίπεδο του χρόνου σε αυτό της συχνότητας. Παρόλο, που η πληροφορία των σημάτων στο πεδίο των συχνοτήτων είναι μεγάλης σημασίας η χρονική πληροφορία χάνεται .Επομένως, αν το σήμα μας περιέχει μη στάσιμα χαρακτηριστικά, όπως μετατόπιση, απότομες αλλαγές κ.ο.κ. η ανάλυση Fourier δεν είναι σε θέση να τα εντοπίσει.

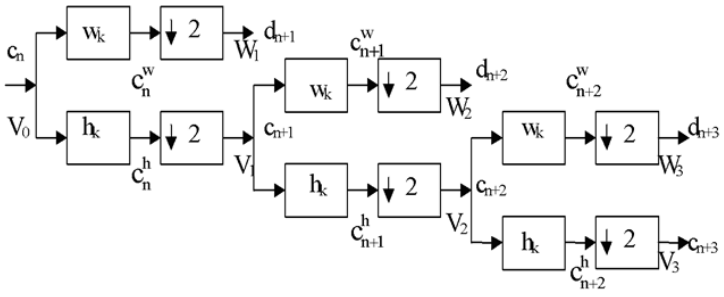
Έτσι ,έγινε δοκιμή να «σπάσει» το συνολικό σήμα σε υποσήματα και να γίνεται ανάλυση ενός χρονικού παραθύρου τη φορά (STFT-Short Time Fourier transform), έχοντας ως αποτέλεσμα την απεικόνιση του σήματος σε μια δισδιάστατη συνάρτηση χρόνου-συχνότητας. Όμως και πάλι η λεπτομέρεια/ακρίβεια της πληροφορίας του χρόνου εξαρτάται και καθορίζεται από το μέγεθος του χρονικού «παραθύρου», το οποίο είναι σταθερό για όλες τις συχνότητες.

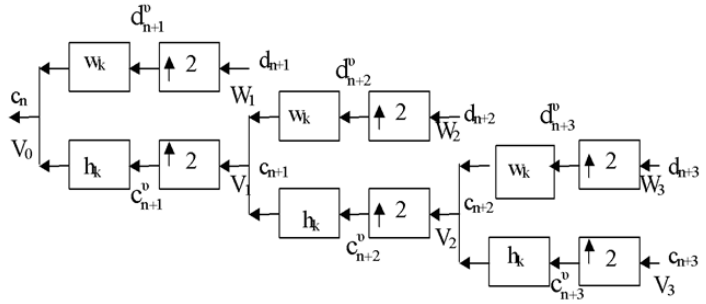
Τα προβλήματα αυτά έγινε προσπάθεια να λυθούν με μία νέα τεχνική που ονομάζεται wavelet.

Τα wavelets είναι σχετικά, ένας καινούριος μαθηματικός μετασχηματισμός, που βρίσκει εφαρμογή κυρίως στην Ψηφιακή Επεξεργασία Σημάτων, ό οποίος μετά την εμφάνιση του , το 1985, εξελίχθηκε ραγδαία και χρησιμοποιήθηκε κατά κόρον. Χρησιμοποιούν μία «παραθυρική» τεχνική με περιοχές μεταβλητού μεγέθους. Δηλαδή, για πληροφορίες χαμηλής συχνότητας έχουμε μεγάλα χρονικά παράθυρα ενώ για πληροφορίες υψηλής συχνότητας μικρά χρονικά παράθυρα. Έτσι, επιτυγχάνεται η ανάλυση μιας συγκεκριμένης περιοχής του αρχικού σήματος με απόρροια την αποκάλυψη όψεων των δεδομένων που πιθανά χάνονται με άλλες τεχνικές. Ακόμη η τεχνική αυτή χρησιμοποιείται για συμπίεση η αποθορυβοποίηση ενός σήματος με πολύ μικρό εκφυλισμό.
Με τον όρο wavelet (δηλαδή «κυματίδιο», νοείται η μορφή ενός μικρού, εντοπισμένου στο χρόνο κύματος .Αυτό σημαίνει ότι λαμβάνει και θετικές και αρνητικές τιμές.
Τα κύρια είδη των wavelets έχουν ως εξής:

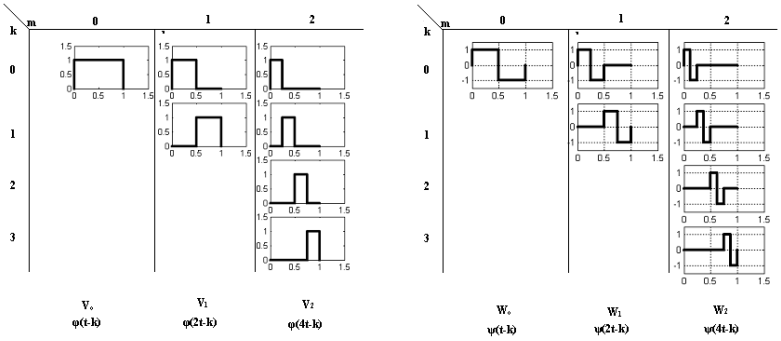
- Continuous Wavelet Transform (CWT) : Η συνάρτηση και ο μετασχηματισμός είναι συνεχείς συναρτήσεις.
- Χρονικά Διακριτός Μετασχηματισμός Wavelet Συνεχούς
- Συνάρτησης (Discrete Time Wavelet Transform, DWT) :Η συνάρτηση είναι συνεχής αλλά ο wavelet μετασχηματισμός είναι διακριτός.
- Διακριτός Μετασχηματισμός Wavelet (Discrete Wavelet Transform, DWT) : Η συνάρτηση και ο μετασχηματισμός είναι διακριτοί.

Στην εργασία αυτή γίνεται χρήση του DTW (Διακριτός Μετασχηματισμός Wavelet.Αποτελεί έναν αρκετά γρήγορο μετασχηματισμό και βασίζεται στη λογική ότι με την χρήση ανάλυσης πολλαπλών επιπέδων είναι εφικτή η αναπαράσταση ενός σήματος x(n),για την μετατροπή του σήματος σε wavelet συντελεστές. Ως σήμα πολλαπλών επιπέδων ορίζεται είναι η διαίρεση του σήματος x(t) σε επί μέρους σήματα, σε κάθε ένα από τα οποία περιέχεται ένα μέρος των πληροφοριών και των συχνοτήτων που εμπεριέχονται και στο αρχικό σήμα. Στα Σχήματα 2α αλλά και 2β βλέπουμε μία γραφική αναπαράσταση των δύο αυτών μετασχηματισμών. [17][18][19]





Σχήμα 2α.Ανάλυση και επανασύνθεση του σήματος



Σχήμα 2β. Σήμα Πολλαπλών Επιπέδων

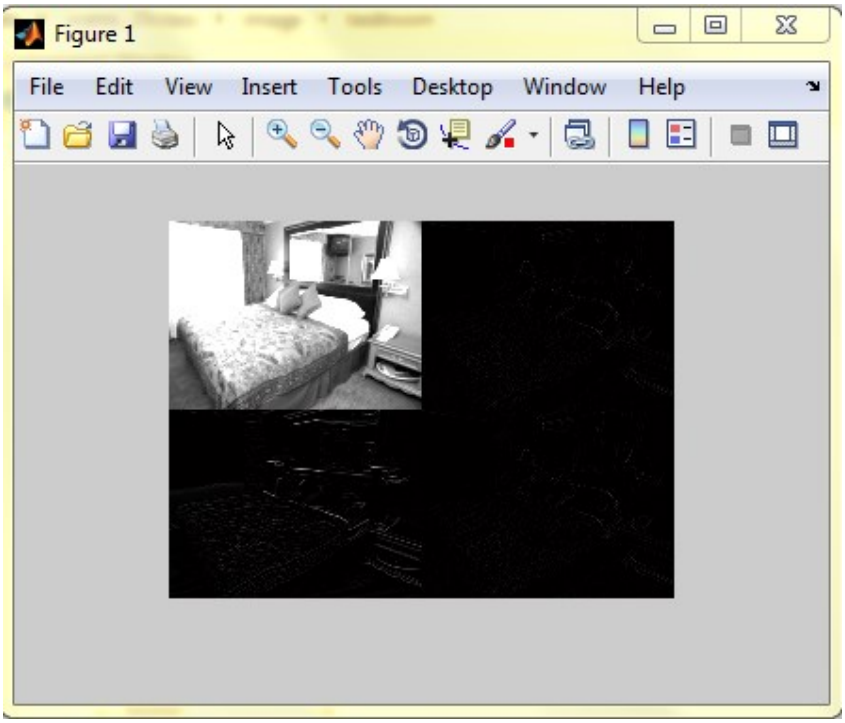
Τα wavelets χρησιμοποιούνται σε μια πληθώρα εφαρμογών ως εργαλεία επεξεργασίας σήματος. Οι κυριότερες είναι στην Ιατρική και τη Μηχανολογία, για εντοπισμό απότομων ασυνεχειών και σημείων κατάρρευσης που υποδηλώνουν προβλήματα στην λειτουργία και οι οποίες θα χάνονταν λόγω θορύβου αν χρησιμοποιούσαμε άλλες τεχνικές ,όπως τον μετασχηματισμό Fourier. Ακόμη, εφαρμόζονται στην αστρονομία για τον διαχωρισμό σημάτων από το θόρυβο (αποθορυβοποίηση), ακόμα και στα μακροοικονομικά για μακροπρόθεσμες προβλέψεις της πορείας των μετοχών. Μια από τις πιο γνωστές εφαρμογές, στις οποίες γίνεται χρήση wavelet είναι ο αλγόριθμος συμπίεσης εικόνας JPEG2000, ο οποίος δημιουργήθηκε προς αντικατάσταση του JPEG. Ο αλγόριθμος αυτός πετυχαίνει πολύ ανώτερα αποτελέσματα από τον κλασικό JPEG όσον αφορά την ταχύτητα αλλά και την συμπίεση των αρχείων. Ακόμη τα wavelets μπορούν να δίνουν επιλεκτικά μέχρι ένα επίπεδο λεπτομερειών σε ένα σήμα. Αυτό είναι χρήσιμο για περιπτώσεις που έχουμε μεγάλες εικόνες από τις οποίες δεν χρειαζόμαστε πολλές λεπτομέρειες, αλλά συγκεκριμένα βασικά στοιχεία.

2.2.3.2 ΔΗΜΙΟΥΡΓΙΑ ΑΛΓΟΡΙΘΜΟΥ ΜΕ ΤΗ ΧΡΗΣΗ WAVELET ΣΕ ΠΕΡΙΒΑΛΛΟΝ MATLAB

Αρχικό βήμα για την έναρξη της ανάπτυξης των νέων αλγορίθμων ,ήταν η μελέτη των διακριτών wavelets, τα οποία συνδυάστηκαν με σχεδόν όλους τους αλγόριθμους που αναπτύχθηκαν, λόγω της μεγάλης πληροφορίας που παρέχουν κατά τον μετασχηματισμό καθώς και της ταχύτητας τους. Ακόμη, ο διακριτός μετασχηματισμός, έγινε σε δύο επίπεδα, χρησιμοποιώντας την συνάρτηση dwt2 η οποία ορίζεται ως

```
[ca,ch,cv,cd] = dwt2(image,'db1','mode','sym');
```

,όπου image η εικόνα επάνω στην οποία τα εφαρμοστεί η ανάλυση, με τύπο επέκτασης mode.Οι μεταβλητές εξόδου αποτελούν την προσεγγιστική εικόνα (εκείνη με την περισσότερη λεπτομέρεια),καθώς και την οριζόντια, κάθετη αλλά και διαγώνια ανάλυση, αντίστοιχα. Παρακάτω, βλέπουμε ένα παράδειγμα εφαρμογής της συνάρτησης αυτής.



Σχήμα 2γ. Αποτέλεσμα wavelet.

Αυτό μπορεί να το εφαρμόσει κάποιος με τα εξής βήματα:

- Να επιλέξει την εικόνα πάνω στην οποία θέλει να εφαρμόσει το wavelet.

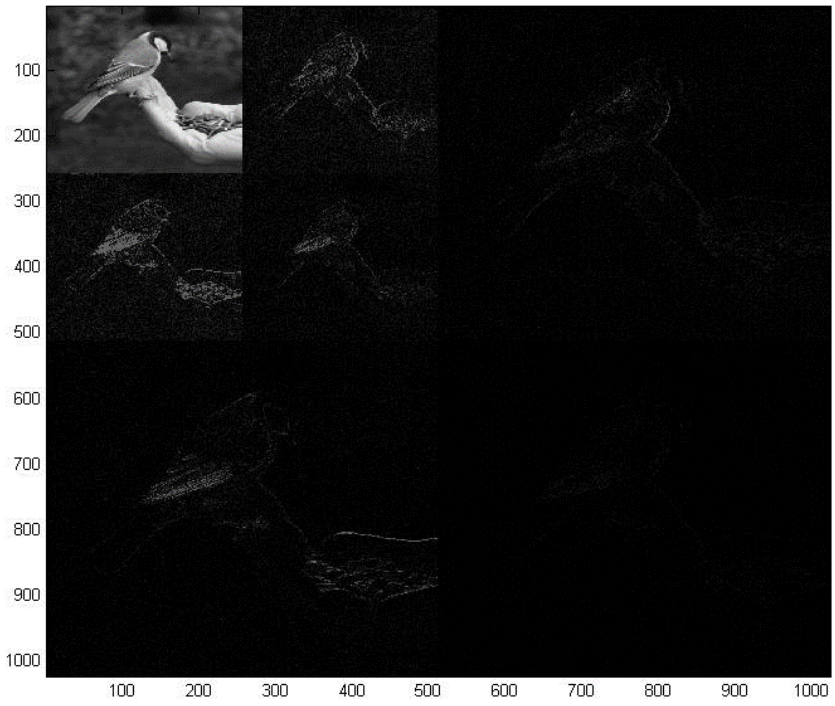
- Να φορτώσει την εικόνα με την εντολή aa=imread(‘image_name.jpg’);
- Να εφαρμόσει wavelet με την εντολή [ca ch cv cd]=dwt2(aa,’db1’,’mode’,’sym’); (Να σημειωθεί εδώ, ότι μπορεί κάποιος να αλλάξει τα ορίσματα της συνάρτησης αυτής καθώς δίνονται αρκετές επιλογές. Εμείς στην πτυχιακή εφαρμόσαμε τα παραπάνω ορίσματα. Το όνομα του wavelet αφορά το ποιο wavelet χρησιμοποιούμε. Εμείς εδώ παίρνουμε το’db1’ αλλά υπάρχουν και άλλα όπως το haar,db2 κ.ο.κ. Το mode θέτει το σήμα ή τον τρόπο επέκτασης εικόνας στα διακριτά wavelet. Οι τρόποι επέκτασης αντιπροσωπεύουν διαφορετικούς τρόπους χειρισμού του προβλήματος των συνόρων στρέβλωσης των σημάτων στην ΨΕΕ. Στην Βιβλιογραφία [20] δίνεται η πηγή για την εύρεση διαφορετικών modes.)
- Το αμέσως επόμενο βήμα είναι καθαρά βοηθητικό και αφορά την εμφάνιση του αποτελέσματος. Μπορούμε να ενώσουμε τις 4 αυτές εικόνες σε μία και να την εμφανίσουμε. Αυτό επιτυγχάνεται με την χρήση της παρακάτω εντολής:
  aa = [ca ch; cv cd];
- Τέλος, έχουμε την εμφάνιση του τελικού αποτελέσματος με την imshow(aa,[0 255]). Επειδή τα wavelets περιέχουν και θετικές και αρνητικές τιμές φέρνουμε τις εικόνες στο διάστημα [0,255], κάτι που φυσικά δεν ισχύει στον αλγόριθμό μας, καθότι θέλουμε και τις θετικές και τις αρνητικές τιμές.

Εφόσον δοκιμάσουμε τα wavelets σε μία μεμονωμένη εικόνα, προχωράμε στην εφαρμογή στις 15 κλάσεις. Αρχικά, χρειάστηκε να αναπτυχθεί ένας πρόχειρος αλγόριθμος, ο οποίος ήταν ανεξάρτητος από την compute_feature, διάβαζε όλες τις εικόνες και έκανε εξαγωγή των χαρακτηριστικών στον φάκελο source\data\scene_15class\feature. Αργότερα, έγινε αναπροσαρμογή του κώδικα, όπως υποδεικνύει το Κεφάλαιο 2.2.2.Το μοναδικό βήμα λοιπόν, στην εφαρμογή του αλγορίθμου αυτού ήταν η κλήση μιας συνάρτησης η οποία καλούσε την dwt2 2 φορές και εξήγαγε τα χαρακτηριστικά με χρήση wavelets.Παρακάτω βλέπουμε αυτό το κομμάτι του κώδικα:

```
[ca,ch,cv,cd] = dwt2(im,'db1','mode','sym');
[ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');
```

Στην δεύτερη κλήση της dwt2, χρησιμοποιείται ο συντελεστής προσέγγισης ca ο οποίος προκύπτει από την πρώτη κλήση της dwt2.Έτσι, επιτυγχάνεται ο διακριτός μετασχηματισμός wavelet δύο επιπέδων. Παρακάτω μπορούμε να δούμε το αποτέλεσμα, σε μία εικόνα στο Σχήμα 2δ, ώστε να κατανοήσουμε σχηματικά τον μετασχηματισμό αυτό.

Μία πολύ βοηθητική συνάρτηση για την κατανόηση των μετασχηματισμών wavelet είναι και η wavedemo, η χρήση της οποίας εμφανίζει ένα γραφικό περιβάλλον με το οποίο μας δίνεται η δυνατότητα να παρακολουθήσουμε σχηματικά, με παραδείγματα την διαδικασία των μετασχηματισμών.



Σχήμα 2δ.Μετασχηματισμός Wavelet 2 επιπέδων.

2.2.3.3 ΕΦΑΡΜΟΓΗ LBP ΜΑΖΙ ΜΕ ΤΑ WAVELETS

Μετά την δημιουργία και μελέτη των wavelets, έγινε το πρώτο βήμα για τον συνδυασμό δύο ή περισσότερων χαρακτηριστικών, με σκοπό την αύξηση της απόδοσης του SVM.Ο πρώτος συνδυασμός που χρησιμοποιήθηκε ήταν τα wavelets μαζί με το LBP.

Πιο συγκεκριμένα, αρχικά εφαρμόσαμε σε κάθε εικόνα μετασχηματισμό wavelet, και ύστερα στις αρχικές εικόνες, για κάθε εικόνα εφαρμόσαμε LBP.Προκύπτουν λοιπόν, ένας πίνακας 1x6 από το wavelet καθώς αποθηκεύουμε τα ch2 cv2 cd2 ch cv cd σε έναν πίνακα. Από το LBP έχουμε τα γνωστά L0,L1 και L2.Ακολούθησε η αποθήκευση των 4 αυτών αποτελεσμάτων σε μία κοινή δομή hists για κάθε εικόνα.

Παρακάτω, βλέπουμε τον κώδικα που αναπτύχθηκε σε Matlab για την εξαγωγή των χαρακτηριστικών με αυτόν τον τρόπο. Ουσιαστικά η διαδικασία ανάπτυξης του αλγορίθμου αυτού, όπως και όλων των παρακάτω είναι εκείνη που περιγράφεται στο Κεφάλαιο 2.2.2 , οπότε εδώ γίνεται μόνο η επεξήγηση του βασικού σώματος του κώδικα.

```
function wf = waveletCreator_Lbp(im)



[ca,ch,cv,cd] = dwt2(im,'db1','mode','sym');

 [ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

wf=[ch;cv;cd;ch2;cv2;cd2];



end
```

Η διαδικασία δημιουργίας των wavelet είναι γνωστή και φαίνεται παραπάνω. Εν συνεχεία, γίνεται η δημιουργία των LBP όπως περιγράφηκε στο Κεφάλαιο 1.3 και στη συνέχεια τα αποτελέσματα αυτών των δύο χαρακτηριστικών, ενώνονται σε μία κοινή δομή :

```
wf=struct('wf',f,'L0',feature_L0,'L1',feature_L1,'L2',feature_L2);
```

Η συνάρτηση struct δημιουργεί μία δομή δεδομένων. Δεν παίρνει συγκεκριμένο αριθμό μεταβλητών, και κατά την χρήση της απαιτείται η δήλωση του ονόματος της εκάστοτε μεταβλητής, πριν την αποθήκευσή της. Λ.χ. wf=struct(‘name’,’value).

Η εξαγωγή των χαρακτηριστικών και η εκπαίδευση και δοκιμή του SVM γίνεται με τον ίδιο τρόπο που περιγράφεται στο Κεφάλαιο 1.2.4.

Τα χαρακτηριστικά αυτά εξήχθησαν στον φάκελο \source\data\scene_15class\feature\wavelet_lbp

2.2.3.4 ΕΦΑΡΜΟΓΗ LBP ΣΕ WAVELETS

Η επόμενη ιδέα που αναπτύχθηκε ήταν η εφαρμογή των LBP όχι στις αρχικές εικόνες αλλά στις μετασχηματισμένες με wavelet.Με μια πρώτη σκέψη, η ιδέα αυτή φαντάζει πολύπλοκη, καθώς γίνεται λόγος για 6 wavelet ανά εικόνα με L0,L1 και L2 για κάθε προσέγγιση του wavelet.Δηλαδή, 18 διαφορετικά LBP χαρακτηριστικά ανά εικόνα συν τα wavelets ξεχωριστά. Αρχικά λοιπόν, δημιουργούμε για κάθε εικόνα 7 προσεγγίσεις με την χρήση του wavelet, και για κάθε μία από αυτές τις 7 προσεγγίσεις εξαγάγουμε τα L0,L1 και L2.Τα χαρακτηριστικά αυτά αποθηκεύονται στον φάκελο \source\data\scene_15class\feature\waveletlbp.

```
[ca,ch,cv,cd] = dwt2(Im,'db1','mode','sym');

 [ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

 [L0 L1 L2]=wlbp_lbp_original_4x4(ca2);

 [L3 L4 L5]=wlbp_lbp_original_4x4(ch2);

 [L6 L7 L8]=wlbp_lbp_original_4x4(cv2);

 [L9 L10 L11]=wlbp_lbp_original_4x4(cd2);

 [L12 L13 L14]=wlbp_lbp_original_4x4(ch);

 [L15 L16 L17]=wlbp_lbp_original_4x4(cv);

 [L18 L19 L20]=wlbp_lbp_original_4x4(cd);
```

Στη συνέχεια, εφαρμόζουμε και απλό μετασχηματισμό wavelet και αποθηκεύουμε την δομή μας κατά τον γνωστό τρόπο, με την χρήση της συνάρτησης struct.

```
[wf]=waveletlbpCreator(Im);

WEL=struct('wf',wf,'L0',L0,'L1',L1,'L2',L2,'L3',L3,'L4',L4,'L5',L5,'L6',L6,'L7',L7,'L8',L8,'L9',L9,'L10'
,L10,'L11',L11,'L12',L12,'L13',L13,'L14',L14,'L15',L15,'L16',L16,'L17',L17,'L18',L18,'L19',L19,'L20
',L20);
```

Όπως παρατηρούμε, είναι μία αρκετά μεγάλη δομή, όμως η ανάπτυξή της δεν είναι δύσκολη. Φυσικά, λόγω του μεγέθους της αλλά και της πολλαπλής χρήσης LBP, είναι αρκετά αργή κατά την εκτέλεση και απαιτεί αρκετή επεξεργαστική ισχύ.

2.2.4 ΕΝΕΡΓΕΙΕΣ

2.2.4.1 ΤΙ ΕΙΝΑΙ Η ΕΝΕΡΓΕΙΕΣ: ΟΡΙΣΜΟΣ

Στην επιστήμη της Ψηφιακής Επεξεργασίας Σήματος, ως ενέργεια ορίζεται το παρακάτω άθροισμα:

$E=\sum_{-\infty}^{\infty} |x(n)|^2$
,όπου x(n) το σήμα.

Η E μπορεί να τείνει στο άπειρο η να παίρνει συγκεκριμένες τιμές. Όταν ισχύει το δεύτερο, δηλαδή $0<E<\infty$ τότε το σήμα λέγεται ενεργειακό. Οι ενέργειες στα πλαίσια της Ψηφιακής Επεξεργασίας και Ανάλυσης Εικόνων, εφαρμόζεται όπως και στα υπόλοιπα σήματα. Ως άθροισμα του σήματος της εικόνας.

Κοιτώντας τώρα, τις ενέργειες από πλευράς κώδικα και Matlab, εφαρμόζονται πάρα πολύ εύκολα με την χρήση της συνάρτησης wenergy2.Ομως, εδώ διαφοροποιούμε λίγο τον κώδικα από αποψη wavelet καθώς κάνουμε χρήση της wavedec2 και όχι της dwt2. Η wavedec2 παίρνει ως ορίσματα την εικόνα, το επίπεδο μετασχηματισμού (εμείς κάνουμε μετασχηματισμό 2 επιπέδων), καθώς και το όνομα του wavelet που εφαρμόζουμε, και επιστρέφει τα διανύσματα ανάλυσης.

Ακριβώς μετά εφαρμόζουμε το wenergy2,το οποίο είναι μία συνάρτηση όπου για διακριτό μετασχηματισμό wavelet δύο επιπέδων επιστρέφει το Ea όπου αποτελεί μία προσεγγιστική απεικόνιση/μορφή της αρχικής εικόνας καθώς και τα Eh,Ev,Ed, τα οποία παρέχουν τα ποσοστά ενέργειας που αντιστοιχούν στις οριζόντιες, κάθετες και διαγώνιες λεπτομέρειες αντίστοιχα.

Εδώ,εφόσον είμαστε σε ανάλυση δύο επιπέδων παίρνουμε 7 τιμές:

```
                                [ca,ch,cv,cd] = wenergy2(cx,cl);
                                wf(1,1)=ca;
                                wf(2,1)=ch(1,1);
                                wf(3,1)=ch(1,2);
                                wf(4,1)=cv(1,1);
                                wf(5,1)=cv(1,2);
                                wf(6,1)=cd(1,1);
                                wf(7,1)=cd(1,2);
```

Κάθε μεταβλητή που προέκυψε από την wenergy2 περιέχει τιμές και πρώτου και δεύτερου επιπέδου μετασχηματισμού. Η παραπάνω μορφή, είναι η κλασική μορφή, που θα χρησιμοποιηθεί για όλους τους διαφορετικούς συνδυασμούς των ενεργειών με άλλους αλγορίθμους.

2.2.4.2 ΔΗΜΙΟΥΡΓΙΑ ΑΛΓΟΡΙΘΜΟΥ ΕΝΕΡΓΕΙΩΝ ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ WAVELET ΣΕ ΠΕΡΙΒΑΛΛΟΝ MATLAB

Τα wavelets αλλά και οι ενέργειες είναι δύο πολύ γρήγορες και εύκολες τεχνικές. Αυτό καθιστά και εύκολο τον συνδυασμό τους, καθώς και την εκτέλεση τους.
Τα βήματα που πρέπει να γίνουν είναι πολύ απλά. Δημιουργούμε τα wavelets στην συνέχεια εφαρμόζουμε ενέργειες και αποθηκεύουμε σε μια ενιαία δομή:

```
[cx,cl]=wavedec2(im,2,'db1');
                                [ca,ch,cv,cd] = wenergy2(cx,cl);
                                wf(1,1)=ca;
                                wf(2,1)=ch(1,1);
                                wf(3,1)=ch(1,2);
                                wf(4,1)=cv(1,1);
                                wf(5,1)=cv(1,2);
                                wf(6,1)=cd(1,1);
                                wf(7,1)=cd(1,2);
```

Τα αποτελέσματα του κώδικα αυτού βρίσκονται στον φάκελο \source\data\scene_15class\feature\wavEnergy ενώ ο ολοκληρωμένος κώδικας στο \source\code\scene_sun\features\wavEnergy.

2.2.4.3 ΕΦΑΡΜΟΓΗ ΕΝΕΡΓΕΙΩΝ ΣΕ WAVELET ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ LBP

Μια ενδιαφέρουσα μέθοδος είναι ο συνδυασμός της παραπάνω λογικής με τα LBP.Φυσικά, έχοντας ήδη τρέξει αλγορίθμους οι οποίοι περιλαμβάνουν LBP είναι λογικό να συμπεράνουμε ότι ο αλγόριθμος αυτός θα είναι πιο βαρύς και αργός αλλά σίγουρα τα χαρακτηριστικά θα περιλαμβάνουν πολύ περισσότερη λεπτομέρεια και υποθέτουμε πως θα έχουν πολύ καλύτερη απόδοση, από την απλή χρήση των ενεργειών μόνο. Μία πρώτη ιδέα, να συνδυάσουμε τις ενέργειες που είναι εφαρμοσμένες σε wavelet με το LBP κάθε αρχικής εικόνας. Επομένως σχηματίζεται μία δομή η οποία περιέχει 4 παραμέτρους. Τον πίνακα των ενεργειών και τα L0,L1 και L2.

```
[L0 L1 L2]=wELbp_lbp_original_4x4(Im);

[wf]=waveletEnergyCreator(Im);

WEL=struct('wf',wf,'L0',L0,'L1',L1,'L2',L2);
```

Η συνάρτηση waveletEnergyCreator κάνει εξαγωγή των ενεργειών κατά τον γνωστό τρόπο.Τα χαρακτηριστικά αυτά βρίσκονται στο φάκελο \source\data\scene_15class\feature\wavEnergyLbp ενώ ο κώδικας βρίσκεται στο \source\code\scene_sun\features\wavEnergyLbp.

2.2.4.4 ΕΦΑΡΜΟΓΗ ΕΝΕΡΓΕΙΩΝ ΣΕ WAVELET ΣΕ ΣΥΝΔΥΑΣΜΟ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ LBP ΣΕ WAVELET

Όπως και σε προηγούμενο Κεφάλαιο, έτσι και σε αυτό, επεκτείνουμε τον προηγούμενο αλγόριθμο, δοκιμάζοντας τον συνδυασμό των LBP πάνω στα wavelets ενώ παράλληλα έχουμε και τις ενέργειες πάνω στα wavelets.Φυσικά, ο αλγόριθμος τώρα γίνεται ακόμα πιο βαρύς και δύσκολος στην εκτέλεση.

```
[wf]=waveletEnergyCreator(Im);
[ca,ch,cv,cd] = dwt2(Im,'db1','mode','sym');
[ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

 [L0 L1 L2]=wE_lbp_lbp_original_4x4(ca2);
 [L3 L4 L5]=wE_lbp_lbp_original_4x4(ch2);
 [L6 L7 L8]=wE_lbp_lbp_original_4x4(cv2);
 [L9 L10 L11]=wE_lbp_lbp_original_4x4(cd2);
 [L12 L13 L14]=wE_lbp_lbp_original_4x4(ch);
 [L15 L16 L17]=wE_lbp_lbp_original_4x4(cv);
 [L18 L19 L20]=wE_lbp_lbp_original_4x4(cd);
```

Έτσι, λοιπόν, έχουμε την εφαρμογή των LBP για κάθε ένα από τα ca2,ch2,cv2,cd2,ch, cv,cd ενώ στη συνέχεια γίνεται δημιουργία των ενεργειών πάνω στα wavelet και η αποθήκευση σε μία ενιαία δομή. Τα αποτελέσματα της εκτέλεσης του κώδικα αυτού βρίσκονται στο φάκελο \source\data\scene_15class\feature\wavEnergy_lbp ενώ ο κώδικας βρίσκεται στο source\code\scene_sun\features\wavEnergy_lbp.

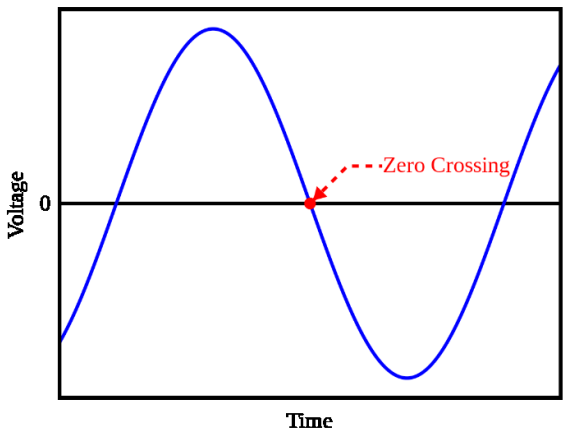
2.2.5 ZERO CROSSINGS

2.2.5.1 ΟΡΙΣΜΟΣ: ΤΙ ΕΙΝΑΙ ΤΑ ZERO CROSSINGS

Ως zero crossing ορίζεται ένα σημείο στο οποίο το σήμα μιας μαθηματικής συνάρτησης αλλάζει (από θετικό σε αρνητικό).Αυτό παρουσιάζεται πάνω σε ένα γράφημα, ως μια γραμμή που τέμνει των άξονα x'x. Χρησιμοποιείται σε επιστήμες όπως η Ηλεκτρονική. τα Μαθηματικά, την Ψηφιακή Επεξεργασία Ήχου αλλά και την Ψηφιακή Επεξεργασία Εικόνας.

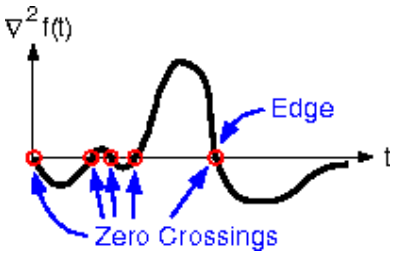
Στο πεδίο της ΨΕΕ δίνεται ιδιαίτερη έμφαση στους τελεστές που κάνουν αναζήτηση ακμών σε μια εικόνα. Η τεχνική αυτή καλείται «εύρεση ακμών» ή «φίλτρα κλίσης». Ένα τέτοιο φίλτρο αναζητά περιοχές στις οποίες υφίσταται έντονη και άμεση αλλαγή των εικονοστοιχείων. Τα σημεία αυτά χαρακτηρίζουν συνήθως μία ακμή ή ένα όριο.

Ένα φίλτρο Laplace είναι ένα φίλτρο αυτής της οικογένειας, παρόλο που εκτελεί την εργασία με διαφορετικό τρόπο. Ψάχνει σημεία όπου το ψηφιακό σήμα μιας εικόνας περνάει από μια προκαθορισμένη τιμή '0' και σηματοδοτεί αυτό το σημείο ως πιθανό σημείο ακμής. Επειδή το σήμα τέμνει το σημείο '0' καλείται zero crossing (το σημείο που τέμνει το '0'). [21]
Παρακάτω βλέπουμε και ένα σχηματικό παράδειγμα των zero crossings:



Σχήμα 2ε.zero crossing

Σχήμα 2στ. Εύρεση ακμών με zero crossings.



Σχήμα 2ζ. Εύρεση ακμών μέσω zero crossings.

2.2.5.2 ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ ZERO CROSSINGS ΣΕ ΠΕΡΙΒΑΛΛΟΝ MATLAB

Αρχικά, βλέπουμε την απλή υλοποίηση των zero crossings σε περιβάλλον Matlab, ώστε να γίνει κατανόηση του κώδικα και της λογικής του, και ύστερα παρουσιάζεται ο συνδυασμός των zero crossings με άλλες τεχνικές όπως τα wavelets ή το lbp.
Ο κώδικας που χρησιμοποιείται για την εξαγωγή των zero crossings φαίνεται παρακάτω:

```
z1=find(diff(rh>0)~=0)+1;
xh=length(z1);
```

Στο Z1 αποθηκεύονται τα σημεία τα οποία γίνονται αλλαγές πάνω και κάτω από το 0, για την εικόνα rh ,ενώ στο xh κρατάει το συνολικό αριθμό των σημείων αυτών.
Με βάση αυτό τον κώδικα υλοποιούνται και τα υλοποιούνται και τα zero crossings στους συνδυαστικούς κώδικες.

2.2.5.3 ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ ZERO CROSSINGS ΣΕ ΣΥΝΔΥΑΣΜΟ ΜΕ LBP ΠΑΝΩ ΣΕ WAVELETS

Στο προηγούμενο Κεφάλαιο είδαμε μία απλή και βασική υλοποίηση των zero crossings σε περιβάλλον Matlab. Επιχειρούμε να αυξήσουμε την απόδοση των SVM συνδυάζοντας δύο διαφορετικούς αλγορίθμους μαζί με τα zero crossings, αυτούς των wavelets και των lbp.

Αρχικά γίνεται εφαρμογή μετασχηματισμού wavelet πάνω στις εικόνες των 15 κλάσεων. Μετά, στα αποτελέσματα που εξάγονται εφαρμόζουμε zero crossings.Τα αποτελέσματα αυτά συνδυάζονται με την εφαρμογή lbp πάνω σε μετασχηματισμό wavelet και τα τελικά αποτελέσματα αποθηκεύονται σε μία κοινή δομή για κάθε εικόνα.

Ο κώδικας είναι αποθηκευμένος στο φάκελο source\code\scene_sun \features\zc_lbp ενώ τα αποτελέσματα αποθηκεύονται στο source\data\ scene_15class\feature\zc_lbp.

```
[ca,ch,cv,cd] = dwt2(im,'db1','mode','sym');
 [ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

rh = ch;
z1=find(diff(rh>0)~=0)+1
xh=length(z1);

rv = cv;
z2=find(diff(rv>0)~=0)+1
xv=length(z2);

rd = cd;
z3=find(diff(rd>0)~=0)+1; xd=length(z3);

rh2 = ch2;
z4=find(diff(rh2>0)~=0)+1; xh2=length(z4);

rv2 = cv2;
z5=find(diff(rv2>0)~=0)+1
xv2=length(z5);

rd2 = cd2;
z6=find(diff(rd2>0)~=0)+1; xd2=length(z6);

ra2 = ca2;
z7=find(diff(ra2>0)~=0)+1;
xa2=length(z7);

z1all=[xh ; xv; xd; xh2; xv2; xd2; xa2];
```

Παραπάνω βλέπουμε αναλυτικά την υλοποίηση των zero crossings μετά την εφαρμογή του μετασχηματισμού wavelet. Για κάθε προσέγγιση (οριζόντια ,διαγώνια και κάθετη) και των δύο επιπέδων του μετασχηματισμού, εφαρμόζουμε zero crossings και αποθηκεύουμε τα αποτελέσματα σε μια ενιαία δομή.

```
[ca,ch,cv,cd] = dwt2(I,'db1','mode','sym');
 [ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

 [L0 L1 L2]=lbp_original_4x4(ca2);
 [L3 L4 L5]=lbp_original_4x4(ch2);
 [L6 L7 L8]=lbp_original_4x4(cv2);
 [L9 L10 L11]=lbp_original_4x4(cd2);
 [L12 L13 L14]=lbp_original_4x4(ch);
 [L15 L16 L17]=lbp_original_4x4(cv);
 [L18 L19 L20]=lbp_original_4x4(cd);
[zall]=waveletZcCreator2(I);

zcall=struct('zall',zall,'L0',L0,'L1',L1,'L2',L2,'L3',L3,'L4',L4,'
L5',L5,'L6',L6,'L7',L7,'L8',L8,'L9',L9,'L10',L10,'L11',L11,'L12',L
12,'L13',L13,'L14',L14,'L15',L15,'L16',L16,'L17',L17,'L18',L18,'L1
9',L19,'L20',L20);
```

Στη συνέχεια, εφαρμόζουμε LBP σε εικόνες μετασχηματισμένες κατά wavelet με τον γνωστό τρόπο και όλα τα αποτελέσματα αποθηκεύονται σε μία κοινή δομή.

Ο εν λόγω αλγόριθμος, είναι ένας αρκετά αποδοτικός αλγόριθμος αλλά η χρήση των LBP , καθιστά τον κώδικά του αρκετά βαρύ και αργό, καθώς απαιτείται αρκετή ώρα για να εκτελεστεί και να παράγει αποτελέσματα.

2.2.5.3 ΕΝΕΡΓΕΙΕΣ ΕΦΑΡΜΟΣΜΕΝΕΣ ΣΕ WAVELETS ΣΕ ΣΥΝΔΥΑΣΜΟ ΜΕ ZERO CROSSINGS

Στο Κεφάλαιο 2.2.4 έγινε αναφορά σε τεχνικές που χρησιμοποιούν ενέργειες για την εξαγωγή των χαρακτηριστικών από τις εικόνες. Εδώ, γίνεται μία δοκιμή των ενεργειών σε συνδυασμό με τα zero crossings.

Πιο συγκεκριμένα, γίνεται εφαρμογή μετασχηματισμού wavelet στις εικόνες. Κατόπιν, εφαρμόζονται zero crossings.Στη συνέχεια ,ξανά σε μετασχηματισμένες κατά wavelet εικόνες, εφαρμόζονται ενέργειες. Όλα αυτά τα χαρακτηριστικά που εξήχθησαν, αποθηκεύονται σε μία δομή, στο φάκελο source\data\scene_15class\ feature\WavEnergy_WavZc. Έχουμε μία δομή για κάθε εικόνα κάθε κατηγορίας . Ο συνολικός κώδικας βρίσκεται στο φάκελο \source\code\scene_sun\features \WavEnergy_WavZc.

```
[ca,ch,cv,cd] = dwt2(im,'db1','mode','sym');
 [ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');
rh = ch;
z1=find(diff(rh>0)~=0)+1;
xh=length(z1);

rv = cv;
z2=find(diff(rv>0)~=0)+1;
xv=length(z2);
```

```
[cx,cl]=wavedec2(im,2,'db1');
                              [ca,ch,cv,cd] = wenergy2(cx,cl);
                              wf(1,1)=ca;
                              wf(2,1)=ch(1,1);
                              wf(3,1)=ch(1,2);
                              wf(4,1)=cv(1,1);
                              wf(5,1)=cv(1,2);
                              wf(6,1)=cd(1,1);
                              wf(7,1)=cd(1,2);
```

```
wf=waveletEnergyCreator_WavEnergy_WavZc(Im);
 zc=waveletZcCreator(Im);

WEL=struct('wf',wf,'zc',zc);
```

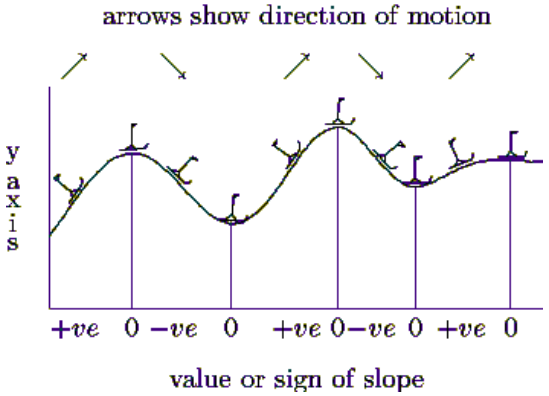
Ο τρόπος δημιουργίας των zero crossings,των ενεργειών αλλά και της αποθήκευσης τους, είναι ο ίδιος με τα προηγούμενα Κεφάλαια, στα οποία χρησιμοποιήθηκαν αυτές οι τεχνικές εξαγωγής χαρακτηριστικών, και διαφαίνεται στα παραπάνω μέρη κώδικα. Στο πρώτο μέρος βλέπουμε ένα κομμάτι από τη δημιουργία των zero crossings,στο δεύτερο γίνεται μία υπενθύμιση των ενεργειών ενώ στο τρίτο και τελευταίο κομμάτι κώδικα βλέπουμε πως γίνεται η αποθήκευση των χαρακτηριστικών που εξήχθησαν.
Ο αλγόριθμος αυτός, είναι ένας γρήγορος αλγόριθμος.

2.2.6 SIGN SLOPE CHANGE

2.2.6.1 ΟΡΙΣΜΟΣ: ΤΙ ΕΙΝΑΙ ΤΟ SIGN SLOPE CHANGE

Η τελευταία δοκιμή πάνω στην βάση δεδομένων των 15 κλάσεων, στα πλαίσια της πτυχιακής αυτής εργασίας, προσθέτει ακόμα μία μαθηματική μέθοδο τα sign slope changes.

Ουσιαστικά, πρόκειται για μία μέθοδο όπου, εντοπίζει τοπικά τα σημεία στα οποία το σήμα αυξάνεται, φθίνει ή παραμένει σταθερό. Τα σημεία αυτά αποτελούν τοπικά μέγιστα και ελάχιστα του σήματος. Παρακάτω ,στο Σχήμα 2η, βλέπουμε ένα ενδεικτικό βοηθητικό σχήμα, των σημείων αυτών, ενός σήματος. [22]



Σχήμα 2η. Τα τοπικά ακρότατα ενός σήματος.

2.2.6.2 ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΟΥ SIGN SLOPE CHANGE ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ WAVELETS

Η υλοποίηση των sign slope changes έγινε σε συνδυασμό με τα wavelets, καθώς, από προηγούμενους πειραματισμούς προέκυψε το συμπέρασμα ότι τα wavelets αποτελούν έναν αρκετά βοηθητικό παράγοντα στην αναγνώριση και ταξινόμηση των εικόνων, καθώς εμπεριέχουν πληροφορία του σήματος της εικόνας και ως προς το χρόνο αλλά και ως προς την συχνότητα (για αυτό και έχουν χρησιμοποιηθεί κατά κόρον σε όλους σχεδόν τους πειραματισμούς μας).

Παρακάτω παρατίθενται ο αλγόριθμος των sign slope changes σε Matlab, όπως χρησιμοποιήθηκε στην εργασία αυτή, συνδυασμένος με μετασχηματισμό wavelet.

```
[ca,ch,cv,cd] = dwt2(im,'db1','mode','sym');
[ca2 ch2 cv2 cd2] = dwt2(ca,'db1','mode','sym');

SSCh = sum(sum(and((ch > 0), not(circshift((ch > 0), 1)))));

SSCv = sum(sum(and((cv > 0), not(circshift((cv > 0), 1)))));

SSCd = sum(sum(and((cd > 0), not(circshift((cd > 0), 1)))));

SSCh2 = sum(sum(and((ch2 > 0), not(circshift((ch2 > 0), 1)))));

SSCv2 = sum(sum(and((cv2 > 0), not(circshift((cv2 > 0), 1)))));

SSCd2 = sum(sum(and((cd2 > 0), not(circshift((cd2 > 0), 1)))));
```

Έχουμε λοιπόν την δημιουργία του μετασχηματισμού wavelet στην εκάστοτε εικόνα, κατά τον γνωστό τρόπο. Στη συνέχεια, στις μεταβλητές προσδιορισμού που προέκυψαν από τα wavelets,εφαρμόζεται η μέθοδος sign slope changes.Συγκεκριμένα, γίνεται αθροιστικός έλεγχος των αλλαγών της εκάστοτε μεταβλητής. Τα αποτελέσματα αποθηκεύονται σε μία ενιαία δομή, στον φάκελο source\data\scene_15class\feature\ssc ενώ ο κώδικας βρίσκεται αποθηκευμένος στο φάκελο source\code\scene_sun\features\ssc.

ΚΕΦΑΛΑΙΟ 3: ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΠΕΙΡΑΜΑΤΙΚΟΥ ΚΩΔΙΚΑ

3.1 ΕΙΣΑΓΩΓΗ

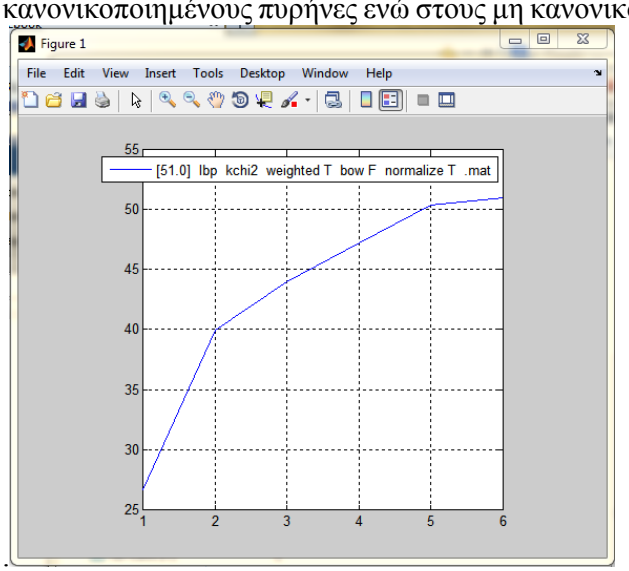
Κατά την διάρκεια των πειραματισμών, το πιο δύσκολο κομμάτι ήταν η παρατήρηση και ανάλυση των αποτελεσμάτων, ώστε να αυξηθούν κατά το μέγιστο βαθμό, για την επίτευξη της αρχικής σκοποθεσίας, αλλά και για την παρουσίαση τους στην εργασία αυτή. Στο Κεφάλαιο λοιπόν αυτό, παρουσιάζονται τα τελικά αποτελέσματα όλων των πειραματισμών, συνοπτικά, μαζί με τις γραφικές τους παραστάσεις ,αλλά και μερικές παρατηρήσεις όσον αφορά τους αλγορίθμους που χρησιμοποιήθηκαν για την εξαγωγή των αποτελεσμάτων αυτών.

3.2 ΕΦΑΡΜΟΓΗ LBP ΣΕ WAVELETS

Κατά την πρώτη επαφή με τον κώδικα, η αρχική ιδέα (η οποία αργότερα αναπτύχθηκε), ήταν η εφαρμογή μετασχηματισμού wavelet δύο επιπέδων στις εικόνες, και τη συνέχεια στα αποτελέσματα του μετασχηματισμού αυτού να εφαρμοσθεί LBP.Όμως, τα χαρακτηριστικά που εξήχθησαν, δεν αποθηκεύτηκαν σε μια ενιαία δομή όπως συνηθίσαμε να το βλέπουμε ως τώρα. Αρχικά , εφαρμόστηκε μετασχηματισμός wavelet σε ένα επίπεδο, τα αποτελέσματα αποθηκεύτηκαν ξεχωριστά για κάθε μεταβλητή προσδιορισμού ,και στη συνέχεια έγινε εισαγωγή των χαρακτηριστικών στο SVM. Παρακάτω βλέπουμε τα αποτελέσματα αυτά.

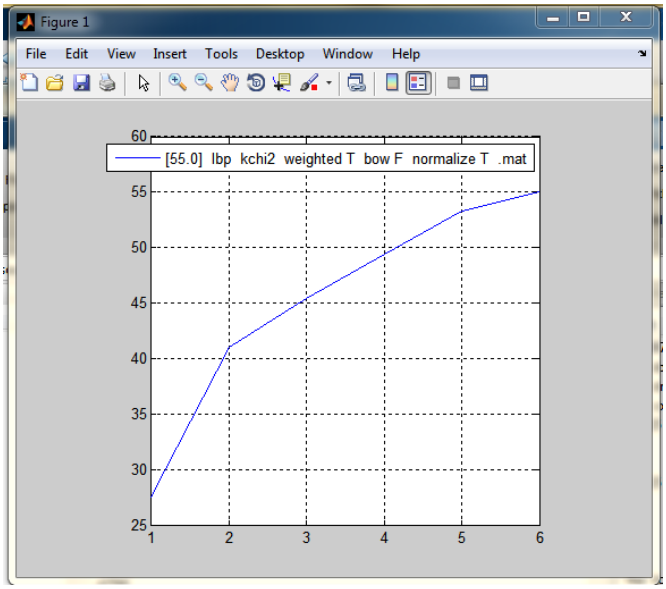
- Εφαρμογή LBP στα χαρακτηριστικά διαγώνιου προσδιορισμού

  Με την εφαρμογή αυτή, ήταν εμφανή τα πρώτα θετικά αποτελέσματα καθώς, υπήρξε αύξηση της απόδοσης σε σχέση με την απλή εφαρμογή LBP στις αρχικές εικόνες. Συγκεκριμένα, βλέπουμε απόδοση ακρίβειας 54.0% για τους κανονικοποιημένους πυρήνες ενώ στους μη κανονικοποιημένους έχουμε 51.0%



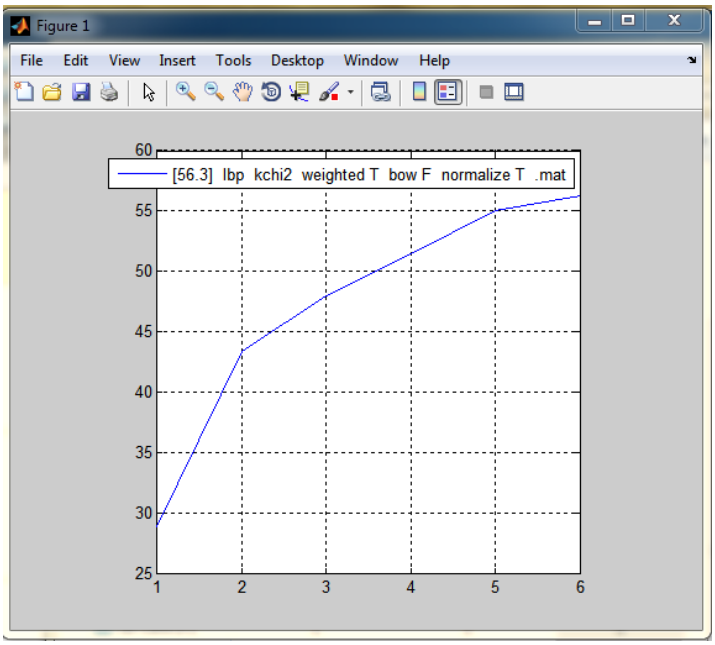
- Εφαρμογή LBP στα χαρακτηριστικά οριζόντιου προσδιορισμού

  Η αμέσως επόμενη εφαρμογή ήταν στα χαρακτηριστικά οριζόντιου προσδιορισμού. Τα αποτελέσματα ήταν ακόμα πιο θετικά, καθώς, υπήρξε μερική αύξηση στο 58.0% για τους κανονικοποιημένους πυρήνες και στο 55.0% για τους μη κανονικοποιημένους. Παρατίθεται και η γραφική παράσταση.



- Εφαρμογή LBP στα χαρακτηριστικά κάθετου προσδιορισμού

  Η τελευταία δοκιμή για αυτόν τον πειραματισμό περιλαμβάνει την εφαρμογή των χαρακτηριστικών των κάθετων προσδιορισμών στο SVM. Ακολούθησε αύξηση στο 59.5% για τους κανονικοποιημένους πυρήνες και στο 56.3% για τους μη κανονικοποιημένους. Βλέποντας συνολικά τα αποτελέσματα, προέκυψε ένα αρχικό συμπέρασμα ότι ίσως τα χαρακτηριστικά των κάθετων προσδιορισμών εμπεριέχουν τις περισσότερες πληροφορίες σχετικά με το χρόνο και τη συχνότητα των εικόνων. Δίνεται και η γραφική παράσταση αυτών.

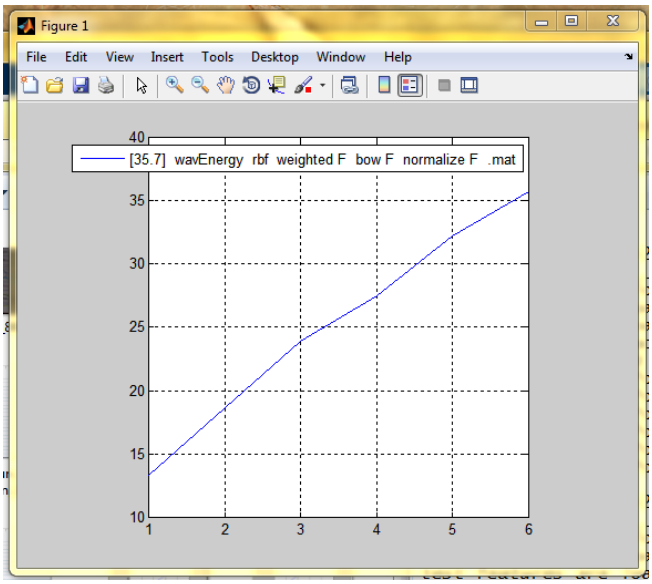


3.3 ΕΝΕΡΓΕΙΕΣ ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ WAVELETS

Η επόμενη ιδέα, αφού είδαμε την ικανότητα του μετασχηματισμού wavelet να αυξήσει τα αποτελέσματα, ήταν να εφαρμόσουμε ενέργειες σε μετασχηματισμό wavelet.Εδώ, δοκιμάσαμε δύο πολύ απλές μεθόδους. Τα χρησιμοποιήσουμε και τις 7 μεταβλητές προσδιορισμού και μετά να αφαιρέσουμε την approximation εικόνα του δευτέρου επιπέδου, και να εφαρμόσουμε τις ενέργειες στις υπόλοιπες 6.Τα τελικά αποτελέσματα σε σχέση με αυτά της εφαρμογής LBP σε wavelet δεν ήταν και τόσο ικανοποιητικά καθώς έπεσε αρκετά η απόδοση του συστήματος.

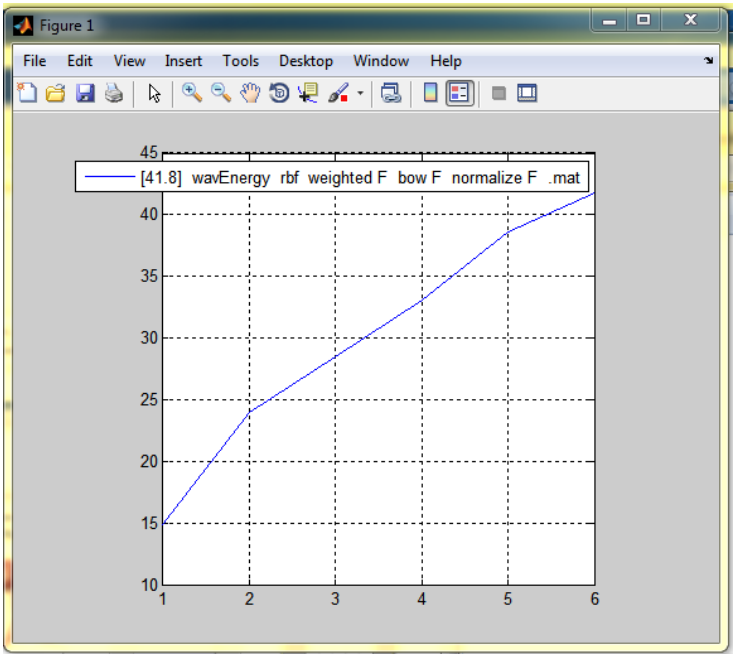
- Εφαρμογή ενεργειών στα 7 χαρακτηριστικά προσδιορισμού του μετασχηματισμού wavelet

  Όπως αναφέρθηκε, τα αποτελέσματα του αλγορίθμου αυτού δεν ήταν καθόλου ικανοποιητικά σε σχέση με τον αλγόριθμο στο Κεφάλαιο 3.2.Όπως κρίνεται σημαντική η αναφορά τους, για την ολοκληρωμένη παρουσίαση της πραγμάτωσης της εργασίας αυτής. Έτσι, λοιπόν, για τους κανονικοποιημένους πυρήνες η ακρίβεια έπεσε στο 36.9% ενώ για τους μη κανονικοποιημένους στο 35.7%.



- <u>Εφαρμογή ενεργειών στα 6 χαρακτηριστικά προσδιορισμού του μετασχηματισμού wavelet</u>

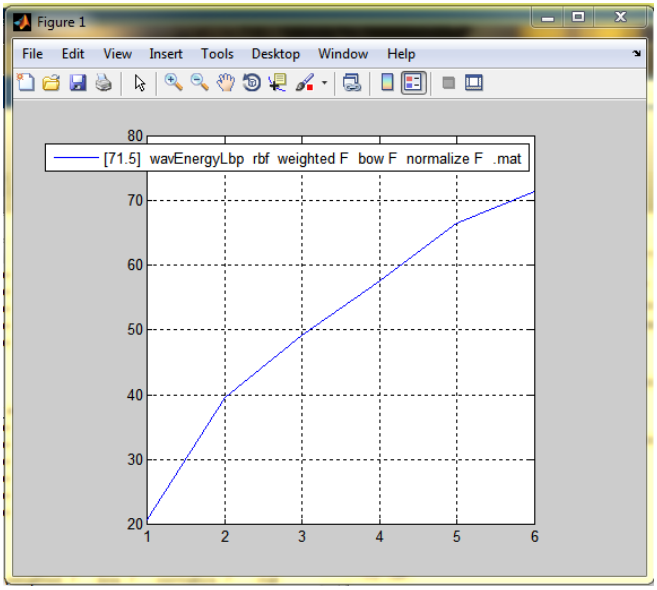
  Τα αποτελέσματα του αλγορίθμου αυτού δεν διαφέρουν πολύ, καθώς, για τους κανονικοποιημένους πυρήνες, έχουμε απόδοση 42.6% ενώ για τους μη κανονικοποιημένους 41.8%. Τα αποτελέσματα σε σχέση με αυτά του προηγούμενου Κεφαλαίου παραμένουν χαμηλά.



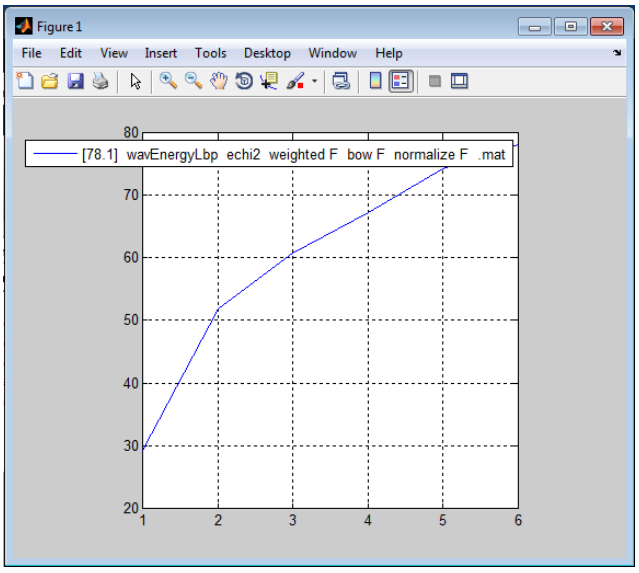
3.4 ΕΦΑΡΜΟΓΗ ΕΝΕΡΓΕΙΩΝ ΣΕ WAVELET ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ LBP

Η παρατήρηση ό,τι τα LBP σε συνδυασμό με τον μετασχηματισμό wavelet αποδίδει ως αλγόριθμος αναγνώρισης και ταξινόμησης εικόνων, και μην θέλοντας να αφήσουμε τις ενέργειες ανεκμετάλλευτες χωρίς να έχουμε προσπαθήσει να τις χρησιμοποιήσουμε συνδυαστικά με έναν αλγόριθμο πιο ισχυρό, όπως αυτό του LBP,προέκυψε η ιδέα για ακόμα έναν αλγόριθμο. Σε αυτόν εφαρμόζουμε ενέργειες σε μετασχηματισμένες εικόνες κατά LBP (χρησιμοποιώντας τις 6 μεταβλητές προσδιορισμού που προκύπτουν από τον μετασχηματισμό wavelet που εφαρμόζεται στην εκάστοτε εικόνα κάθε κατηγορίας),και στη συνέχεια αποθηκεύουμε τα αποτελέσματα σε μία ενιαία δομή, μαζί με τα αποτελέσματα που προκύπτουν από την εφαρμογή LBP στις αρχικές εικόνες (μεταβλητές L0,L1,L2).Ακόμη, εδώ δοκιμάζουμε να εκτελέσουμε τον κώδικα με διαφορετικούς πυρήνες, και παρατηρούμε ότι υπάρχει μία μικρή αυξομείωση στα τελικά αποτελέσματα.

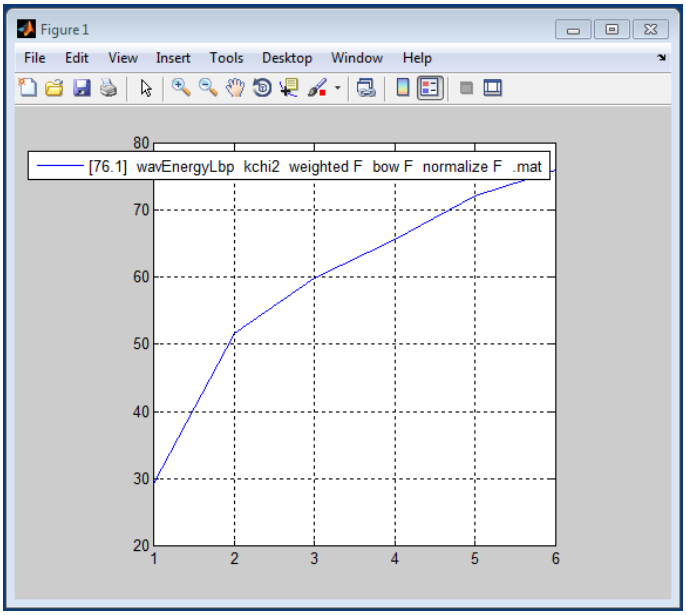
Συγκεκριμένα, η αρχική εκτέλεση έγινε με τον πυρήνα rbf όπου σε κανονικοποιημένη μορφή έδωσε ακρίβεια 72.4% ενώ σε μη κανονικοποιημένη 71.5%.



Στη συνέχεια, έγινε αλλαγή του πυρήνα σε echi2, οπου τα αποτελέσματα αυξήθηκαν και άλλο. Για κανονικοποιημένη μορφή του πυρήνα αυτού , υπήρξε απόδοση στο 73.9% ενώ για μη κανονικοποιημένη στο 81.4%.Η απόδοση αυτή ήταν από τις πιο υψηλές και ικανοποιητικές που είχαν προκύψει , ως τώρα. Παρακάτω παρουσιάζεται και το σχετικό διάγραμμα των αποδόσεων του αλγορίθμου αυτού.



Ο επόμενος , και τελευταίος πυρήνας που εφαρμόστηκε ήταν το kchi2.Τα αποτελέσματα παραμένουν υψηλά αλλά πέφτουν σε σχέση με τη χρήση του echi2.Αναλυτικότερα, στην κανονικοποιημένη μορφή του πυρήνα αυτού έχουμε απόδοση στο 75.5% ενώ στη μη κανονικοποιημένη στο 76.1%.

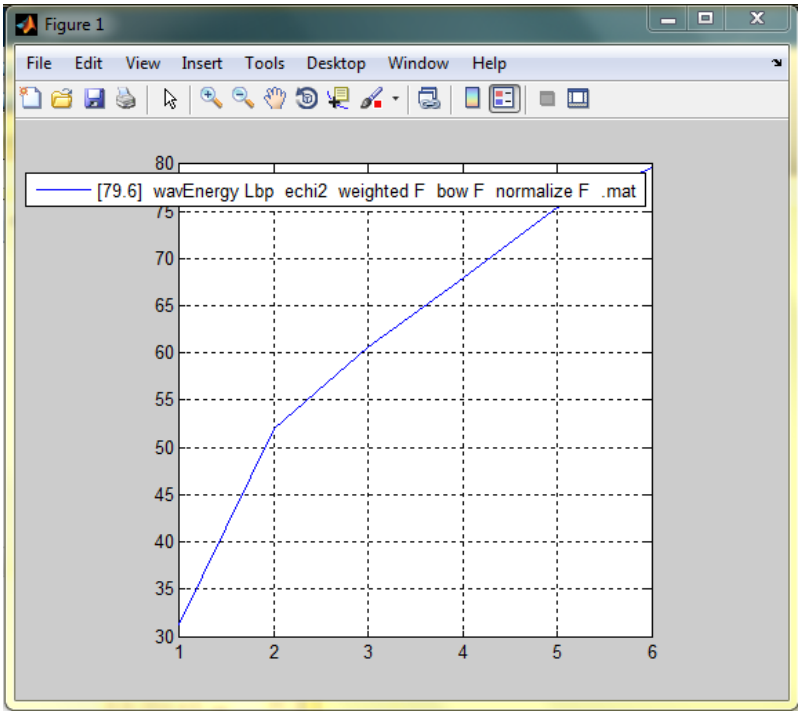


3.5 ΕΦΑΡΜΟΓΗ ΕΝΕΡΓΕΙΩΝ ΣΕ WAVELET ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ ΤΗΝ ΕΦΑΡΜΟΓΗ LBP ΣΕ WAVELET

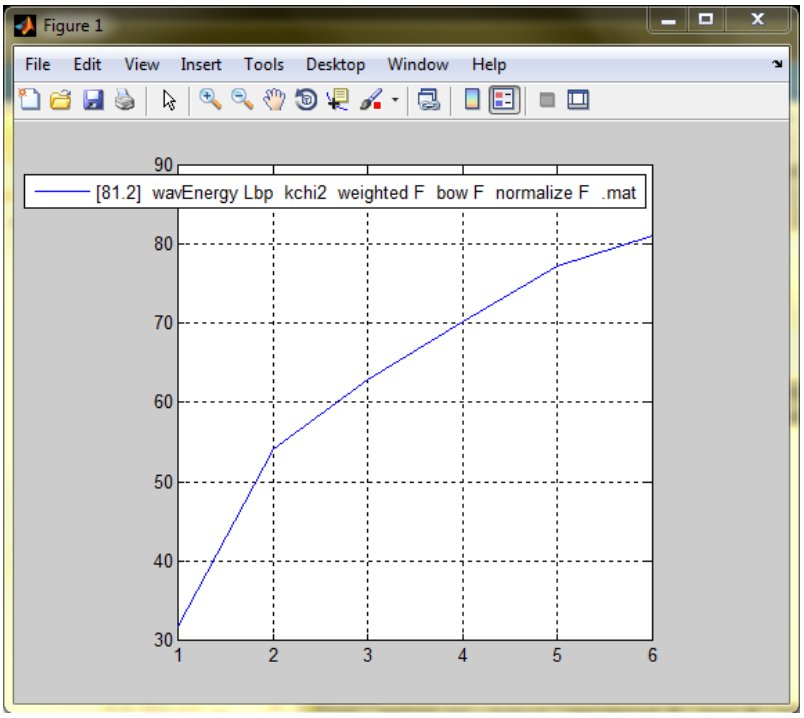
Τα αποτελέσματα του αλγορίθμου στο προηγούμενο Κεφάλαιο, ήταν αρκετά υψηλά, και έδωσαν το βήμα για την εφαρμογή μιας παραλλαγής του αλγορίθμου αυτού. Συγκεκριμένα, οι ενέργειες εφαρμόστηκαν ως έχουν, δηλαδή σε συνδυασμό με τον

μετασχηματισμό wavelet.Η αλλαγή έγινε στα LBP.Δοκιμάσαμε, να εφαρμόσουμε LBP πάνω σε wavelets και τα αποτελέσματα αυτά να αποθηκευτούν σε μία ενιαία δομή, μαζί με τα αποτελέσματα των ενεργειών που εφαρμόστηκαν στα wavelets.Εδώ, χρησιμοποιήσαμε επιλεκτικά τους πυρήνες που έδωσαν τα πιο υψηλά αποτελέσματα στον προηγούμενο αλγόριθμο, δηλαδή τους echi2 και kchi2.

Κατά την εκτέλεση του αλγορίθμου με τον πυρήνα echi2 , οι θεωρητικές υποθέσεις μας , πραγματοποιήθηκαν κατά το δοκούν, καθώς ,η εκτέλεση των χαρακτηριστικών που εξήχθησαν, με τη χρήση του αλγορίθμου αυτού, πάνω στο SVM, απέδωσε κατά 80.6% σε κανονικοποιημένη μορφή και 79.6% σε μη κανονικοποιημένη μορφή, δίνοντας μας, τα καλύτερα αποτελέσματα ,συγκριτικά με τα προηγούμενα, ως τώρα. Παρακάτω παρουσιάζεται και το διάγραμμα της εκτέλεσης αυτής.



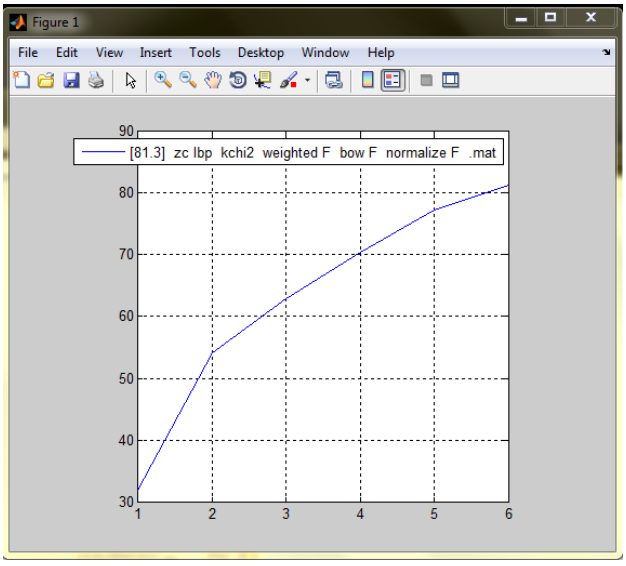
Η εκτέλεση με τη χρήση του kchi2 ,ανέβασε ακόμη περισσότερο τον πήχη των αποτελεσμάτων, αποδίδοντας 73.8% σε κανονικοποιημένη μορφή και 81.3% σε μη κανονικοποιημένη μορφή. Η απόδοση, δίνεται και με το διάγραμμα των αποτελεσμάτων, που φαίνεται παρακάτω.



3.6 ZERO CROSSINGS ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ LBP ΠΑΝΩ ΣΕ WAVELETS

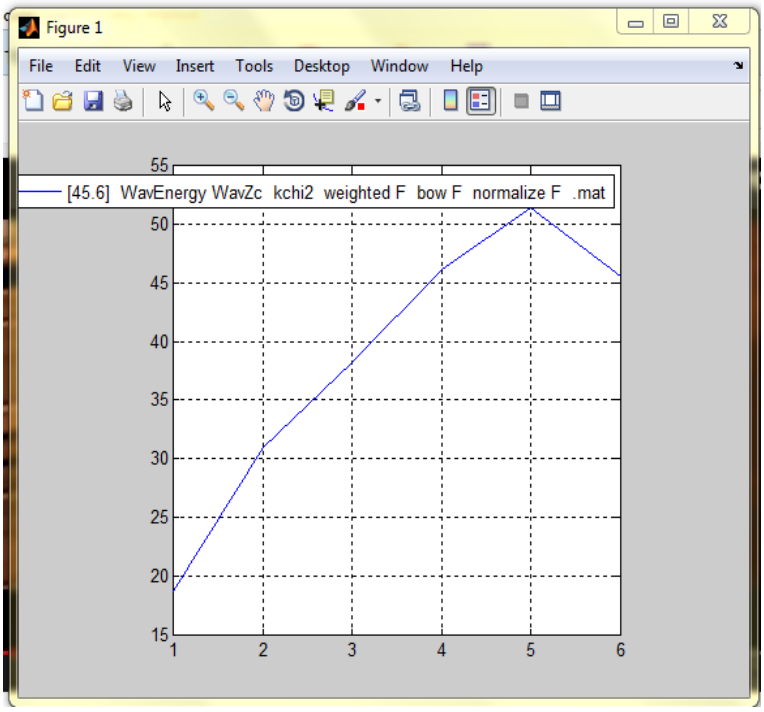
Μία ακόμα νέα έννοια που εισήχθη είναι εκείνη των zero crossings.Ύστερα από μελέτη των zero crossings και του τρόπου λειτουργίας τους, έγινε συνδυασμός με τον αλγόριθμο που εξάγει χαρακτηριστικά από τις εικόνες ,εφαρμόζοντας LBP σε μετασχηματισμό wavelet. Παραδόξως, σε όλες τις προσπάθειες εκτέλεσης του αλγορίθμου αυτού, και ενώ ο κώδικας ελέγχθηκε για τυχόν λογικά λάθη, τα αποτελέσματα ήταν σχεδόν ίδια (τα τελικά αποτελέσματα ήταν επακριβώς ίδια) με εκείνα του αλγορίθμου στο Κεφάλαιο 3.5.

Έτσι λοιπόν, για τον πυρήνα kchi2 , έχουμε ακρίβεια αναγνώρισης 73.8% σε κανονικοποιημένη μορφή ενώ για μη κανονικοποιημένη μορφή έχουμε 81.3%.

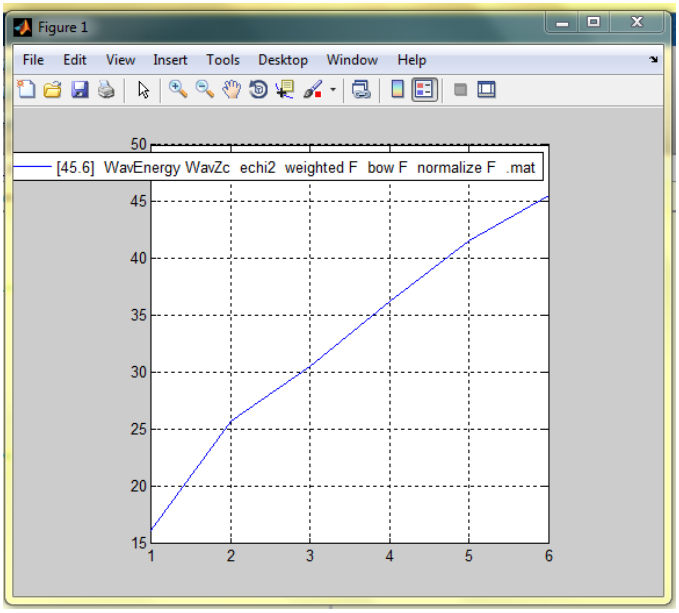


3.7 ΕΝΕΡΓΕΙΕΣ ΕΦΑΡΜΟΣΜΕΝΕΣ ΣΕ WAVELET ΣΕ ΣΥΝΔΙΑΣΜΟ ΜΕ ZERO CROSSINGS

Εφόσον είδαμε πόσο καλά λειτουργούν τα zero crossings (τουλάχιστον σε συνδυασμό με τα LBP), αποφασίσαμε να τα χρησιμοποιήσουμε συνδυάζοντας τα με τις ενέργειες. Φυσικά και τα δύο χαρακτηριστικά εφαρμόζονται πάνω σε εικόνες που πρώτα έχουν μετασχηματιστεί κατά wavelet.Παρόλο που πρόκειται για έναν αρκετά γρήγορο και απλό αλγόριθμο, τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά σε σχέση με αυτά που είχαμε δει ως τώρα. Συγκεκριμένα, όταν χρησιμοποιήθηκε ο πυρήνας kchi2 το σύστημα είχε απόδοση 31.8% και 45.6% για κανονικοποιημένη και μη κανονικοποιημένη μορφή αντίστοιχα.



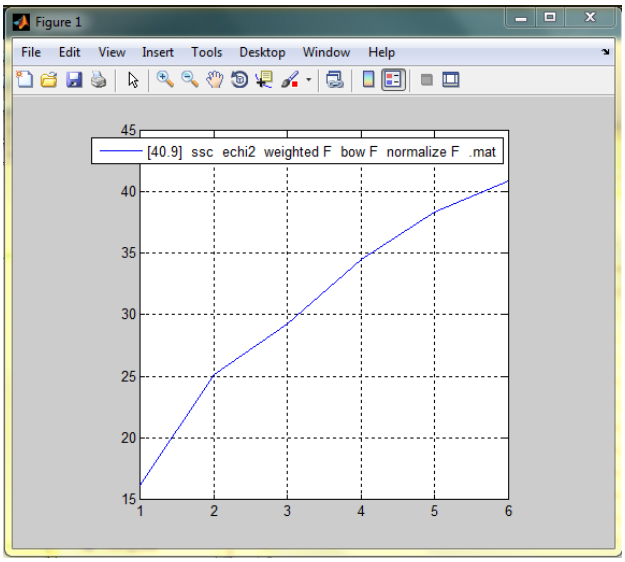
Αλλάζοντας τον πυρήνα σε echi2, δεν πήραμε κάποιο καλύτερο αποτέλεσμα ,απλά λίγο πιο αυξημένο από τον προηγούμενο πυρήνα. Έτσι, λοιπόν , έχουμε απόδοση 47.0% και 45.6% για κανονικοποιημένη και μη κανονικοποιημένη μορφή αντίστοιχα.
Τα αποτελέσματα μπορούμε να τα δούμε και διαγραμματικά στην παρακάτω γραφική παράσταση που προκύπτει από την εκτέλεση του κώδικα με τα παραπάνω στοιχεία.



Τέλος, έγινε ακόμη μία προσπάθεια εκτέλεσης του συγκεκριμένου αλγορίθμου με τον πυρήνα rbf ,όμως τα αποτελέσματα μειώθηκαν στο 39.9% και 39.3%.

3.8 SIGN SLOPE CHANGE

Ένας ακόμη αλγόριθμος εξαγωγής χαρακτηριστικών που χρησιμοποιήσαμε είναι τα sign slope changes, τα οποία εφαρμόστηκαν σε εικόνες μετασχηματισμένες κατά wavelet.Όμως οι πληροφορίες που προκύπτουν από τα sign slopes changes είναι πολύ λίγες (μόλις 6 τιμές για κάθε εικόνα) και όπως είναι λογικό τα αποτελέσματα ήταν αρκετά μειωμένα σε σχέση με αυτά των προηγούμενων αλγορίθμων. Συγκεκριμένα, αποδίδουν 41.4% για κανονικοποιημένο πυρήνα ενώ για μη κανονικοποιημένο 40.9%. Ο πυρήνας που χρησιμοποιήθηκε στα sign slope changes ήταν ο echi2,καθώς είχε την καλύτερη απόδοση στους περισσότερους αλγορίθμους.



3.9 ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Μετά την επιτυχή εξαγωγή των αποτελεσμάτων των πειραματισμών που διεξήχθησαν και την παρουσίαση τους, κρίνεται απαραίτητη η σύγκριση τους με τα αποτελέσματα μεγαλύτερης ακρίβειας όπως προέκυψαν από το πανεπιστήμιο του MIT. Ήδη στα προηγούμενα υποκεφάλαια του Κεφαλαίου 3, παρακολουθήσαμε μία αναλυτική παρουσίαση των αποτελεσμάτων του κάθε αλγορίθμου που δημιουργήθηκε, με διαγράμματα και αποτελέσματα ακρίβειας, με σκοπό την παρουσίαση της συνολικής πορείας (επιτυχούς από άποψη ακρίβειας αναγνώρισης αλλά και μη), των πειραματισμών. Στο υποκεφάλαιο αυτό, επιλέχθηκαν τα πιο υψηλά αποτελέσματα , και από τους πειραματισμούς αλλά και από την παρουσίαση του MIT και γίνεται η μεταξύ τους σύγκριση. Παρακάτω, λοιπόν, βλέπουμε τους πίνακες σύγκρισης των αποτελεσμάτων αυτών.

χαρ/αποτ	comb.nor	comb.un	max
wavEn_wavLbp_echi2	80.6	79.6	80.6
wavEn_wavLbp_kchi2	73.8	81.3	81.3
wavEnergyLbp_echi2	73.9	81.4	81.4
wavEnergyLbp_kchi2	75.5	76.1	76.1
zclbp_kchi2	73.8	81.3	81.3

Πινακας α

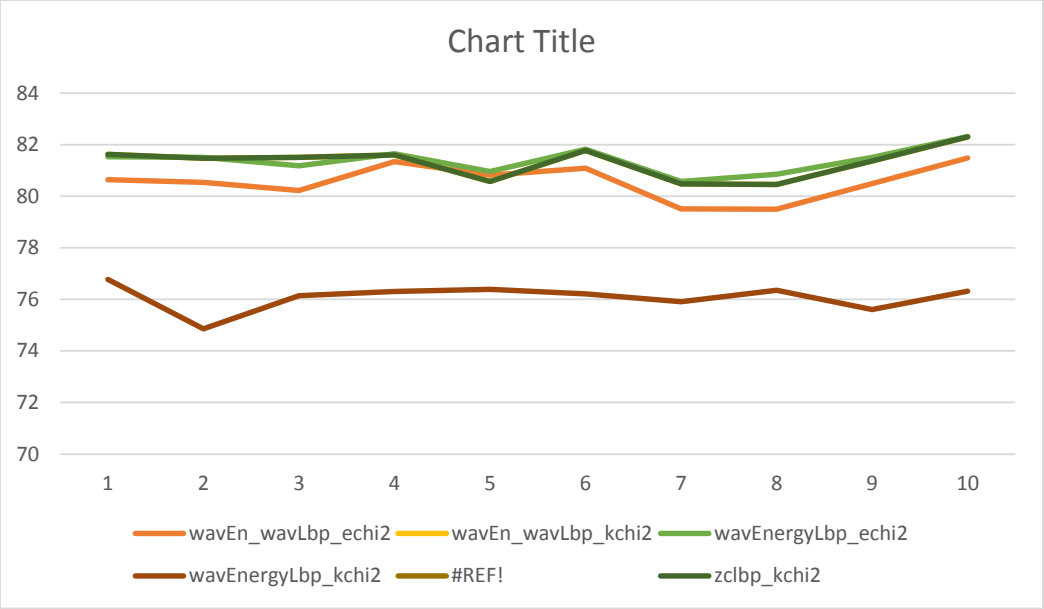
Και από αυτά τα αποτελέσματα κρατάμε τα πιο υψηλά(είτε αναφέρονται σε κανονικοποιημένους ή μη κανονικοποιημένους πυρήνες).Παρακάτω, βλέπουμε, τους αλγορίθμους εξαγωγής χαρακτηριστικών του MIT οι οποίοι έχουν την καλύτερη απόδοση.

χαρ/αποτ	
all	88.1
SIFT	81.2
hog2x2	81.0
texton hist	77.8
ssim	77.2
gist	74.7

Πίνακας β

Η κατασκευή ενός αλγόριθμου ο οποίος θα συγκρίνεται με τον συνδυασμό όλων των αλγορίθμων του MIT αποτελεί μελλοντική επέκταση της εργασίας αυτής, όμως το αποτέλεσμα του εμφανίζεται, καθώς παρουσιάζεται και από το MIT. Οι πειραματισμοί μας κατάφεραν να φέρουν λίγο μεγαλύτερα αποτελέσματα από τον αμέσως επόμενο αλγόριθμο ,τον SIFT όπου τον ξεπέρασαν κατά 0,2%. Συγκεκριμένα γίνεται λόγος για τον αλγόριθμο ο οποίος εφαρμόζει ενέργειες σε μετασχηματισμό wavelet και συνδυάζει με lbp στις αρχικές εικόνες. Ακόμη και τα zero crossings συνδυασμένα με lbp κατάφεραν να φτάσουν την απόδοση του SIFT ξεπερνώντας τον κατά 0,1%. Σε προηγούμενα υποκεφάλαια, είδαμε διαγράμματα με την εξέλιξη του κάθε αλγορίθμου ξεχωριστά. Παρακάτω, βλέπουμε τα διαγράμματα που περιγράφουν την εξέλιξη όλων των αλγορίθμων εν συγκρίσει.

Συγκεκριμένα, στο πρώτο, συγκρίνουμε την πορεία όλων των αλγορίθμων που αναπτύξαμε, παρακολουθώντας παράλληλα στον αμέσως επόμενο πίνακα τα αποτελέσματα που χρησιμοποιήθηκαν για την εξαγωγή του διαγράμματος αυτού. Αναλυτικά, χρησιμοποιούνται τα αποτελέσματα που προέκυψαν από την εκπαίδευση του SVM με την χρήση 100 παραδειγμάτων εκπαίδευσης, στη φάση των συνδυαστικών πυρήνων. Ακόμη, να σημειωθεί ότι δεν εστιάζουμε σε κανονικοποιημένο ή μη κανονικοποιημένο πυρήνα αλλά σε αυτόν που δίνει την μεγαλύτερη ακρίβεια. Στο δεύτερο διάγραμμα παρακολουθούμε την σύγκριση των αποτελεσμάτων μας με τα αποτελέσματα που μας δόθηκαν στη δημοσίευση του MIT.



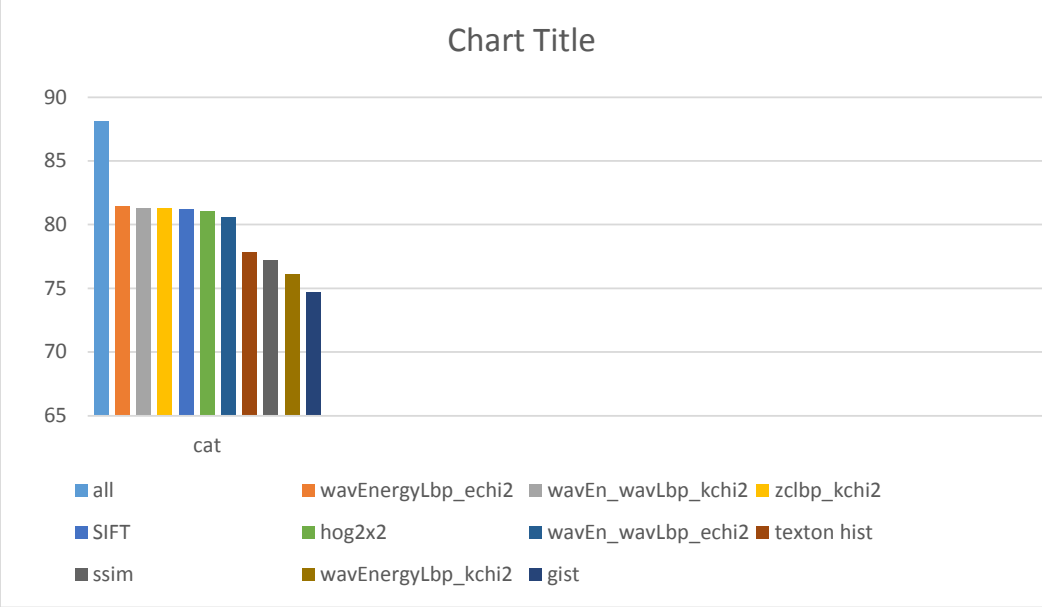
Διάγραμμα α

wavEn_wavLbp_echi2	wavEn_wavLbp_kchi2	wavEnergyLbp_echi2	wavEnergyLbp_kchi2	zclbp_kchi2
80,649871	81,625128	81,534564	76,77216	81,625128
80,537204	81,477153	81,50519	74,857801	81,477153
80,225507	81,508797	81,18564	76,143585	81,508797
81,348991	81,60139	81,652583	76,31253	81,60139
80,823194	80,569548	80,957679	76,386208	80,569548
81,085241	81,781203	81,830848	76,209812	81,781203
79,512818	80,481608	80,572591	75,907034	80,481608
79,496632	80,455906	80,85437	76,360062	80,455906
80,488927	81,367034	81,513675	75,604423	81,367034
81	82,30869	82,312018	76,324553	82,30869

Πίνακας γ

Παρακάτω, γίνεται σύγκριση των δικών μας αποτελεσμάτων με τα αποτελέσματα του MIT.Συγκεκριμένα, παρουσιάζονται με φθίνουσα ταξινόμηση ,όλα τα αποτελέσματα σε ένα διάγραμμα. Όπως, παρατηρούμε , αν εξαιρέσουμε, τον συνδυασμό όλων των αλγορίθμων του MIT μαζί, το οποίο θέτει τον πήχη για μελλοντική επέκταση της πτυχιακής αυτής, όχι μόνο επιτεύχθηκε η αύξηση των αποτελεσμάτων χρησιμοποιώντας τον αλγόριθμο LBP (ο οποίος αρχικά, όταν

εκτελούνταν στη βάση των 15 κλάσεων έδινε κοντά στο 33.1%) σε συνδυασμό με τα wavelets αλλά και με άλλους αλγορίθμους, αλλά ξεπεράσαμε τον SIFT (ο οποίος έρχονταν δεύτερος αμέσως μετά τον all στη δημοσίευση του MIT με ακρίβεια αναγνώρισης στις 15 κλάσεις 81.3% όπως είδαμε και παραπάνω) κατά ένα πολύ μικρό ποσοστό (ουσιαστικά, τα αποτελέσματα είναι ίσα).



Διάγραμμα β.

Το παραπάνω διάγραμμα προέκυψε από τα αποτελέσματα που παρουσιάζονται στους πίνακες 1 και 2.Φυσικά,είναι εμφανής η διαφορά, του αλγορίθμου all με τους υπόλοιπους. Όμως, είναι εμφανή και τα υψηλά αποτελέσματα που απέδωσαν οι πειραματισμοί με τον αλγόριθμο LBP,καθώς μας έφερε στην δεύτερη θέση, μετά τον αλγόριθμο με την καλύτερη απόδοση από όλους.

Στο σημείο αυτό, όπου έχουμε μελετήσει και παρατηρήσει αναλυτικά αλλά και συνολικά τα αποτελέσματα, μπορούμε να πούμε ότι η πτυχιακή αυτή ολοκληρώθηκε επιτυχώς, πετυχαίνοντας την αρχική σκοποθεσία της.

ΚΕΦΑΛΑΙΟ 4: ΕΡΓΑΣΙΕΣ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Ύστερα από διεξοδική μελέτη των αλγορίθμων του MIT αλλά και των αλγορίθμων που δημιουργήθηκαν για να αυξήσουν την απόδοση και την ακρίβεια αναγνώρισης, προκύπτει το συμπέρασμα ότι η δομή ενός τόσο ισχυρού κώδικα, απαιτεί χρόνια εργασίας και μελέτης για να γίνει πλήρως κατανοητή. Όμως κάθε κατανόηση της λειτουργίας του, καθώς και των δυνατοτήτων του.

Υπάρχουν πολυάριθμες εφαρμογές και αλγόριθμοι που μπορούν να δημιουργηθούν και να ενσωματωθούν στον κώδικα αυτό, αυξάνοντας την ακρίβεια με την οποία το SVM αναγνωρίζει και ταξινομεί τις εικόνες.
Καθώς μελετούσαμε τους κώδικες προέκυψαν αρκετές οι ιδέες οι οποίες μπορούν να πραγματωθούν με την επέκταση του κώδικα, εξυπηρετώντας ιατρικούς, μαθηματικούς, τεχνικούς και άλλους σκοπούς. Τέτοιοι μπορεί να είναι:

- Αναγνώριση ανωμαλιών σε ανθρώπινα κύτταρα ή ζωτικά όργανα,
- Αναγνώριση προσώπων σε συστήματα ασφάλειας ή ακόμα και αποτυπωμάτων,
- Εφαρμογές για δημιουργία έξυπνων εγκεφάλων αυτοκινήτων που θα αναγνωρίζουν εμπόδια και θα τα παρακάμπτουν
- Δημιουργία συστημάτων υποβοήθησης ΑΜΕΑ
- Δημιουργία Συστημάτων Παρακολούθησης .κ.ο.κ

Φυσικά οι παραπάνω ιδέες απαιτούν και το κατάλληλο υλικό καθώς και την ύπαρξη εμπειρογνώμονα για την σωστή εκπαίδευση των συστημάτων αυτών.

Εστιάζοντας στο σύστημα της πτυχιακής αυτής εργασίας, το οποίο εξαγάγει χαρακτηριστικά και στη συνέχεια αναγνωρίζει και ταξινομεί εικόνες με βάση τα χαρακτηριστικά αυτά, μπορούμε να πούμε πως μπορεί να επεκταθεί μελλοντικά συνδυάζοντας ακόμα περισσότερες μαθηματικές μεθόδους, ώστε να αυξηθεί η απόδοσή του, όπως για παράδειγμα SIFT με sign slope changes.Επίσης μπορεί να γίνει χρήση διαφορετικών Βάσεων Δεδομένων, μεγαλύτερων ή και μικρότερων, όπως αυτή της SUN με τις 397 κλάσεις εικόνων.

Συμπερασματικά, η ενασχόληση με την επιστήμη της Ψηφιακής Επεξεργασίας Εικόνας σε συνδυασμό με την Τεχνητή Νοημοσύνη, μπορεί να αποδώσει θεαματικά επιστημονικά επιτεύγματα, και να διευρύνει τόσο τις γνώσεις όσο και τους ορίζοντες, να δημιουργήσει ιδέες, οι οποίες κατά την εφαρμογή και ολοκλήρωση τους που μπορούν να αλλάξουν καθοριστικά την ποιότητα της ανθρώπινης ζωής.

ΠΑΡΑΠΟΜΠΕΣ-ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]Εικόνα,el.wikipedia.org, http://el.wikipedia.org/wiki/%CE%95%CE%B9%CE%BA%CF%8C%CE%BD%CE%B1

[2] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ. 1.1 – 1.3

[3] Pixel, en.wikipedia.org, http://en.wikipedia.org/wiki/Pixel

[4] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ. 10.1-10.4

[5] – [6] "SUN Database: Large-scale Scene Recognition from Abbey to Zoo". J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. IEEE Conference on Computer Vision and Pattern Recognition, 2010 , web.mit.edu, http://web.mit.edu/jxiao/Public/publication/2010/CVPR_sun/paper.pdf

[7] Histogram of Oriented Gradients, en.wikipedia.org, http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[8] Scale invariant feature transform-detection, en.wikipedia.org, http://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Scale-invariant_feature_detection

[9]Local Feature Synthesis Draft – Excerpt chapter from Synthesis lecture draft: Visual Recognition – Kristen Grauman and Bastian Leibe , cs.utexas.edu, http://www.cs.utexas.edu/~grauman/courses/fall2009/papers/local_features_synthesis_draft.pdf

[10] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ. 10.74

[11] Canny Edge Detector, en.wikipedia.org, http://en.wikipedia.org/wiki/Canny_edge_detector

[12] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ. 2.14 , 2.44-2.45

[13] Kernels- Image Processing , en.wikipedia.org, http://en.wikipedia.org/wiki/Kernel_%28image_processing%29

[14] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , Παπαμάρκος σελ. 12.87

[15] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ. 12.83

[16] Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας, Νικόλαος Ι. Παπαμάρκος, Έκδοση 3η , σελ 10.74 -10.78

[17]Πανεπιστημιακες σημειώσεις , Δρ. Ι. Κ. Χατζηλάου, Καθηγητής ΣΝΔ, *«ΑΠΟ ΤΑ FOURIER ΣΤΑ WAVELETSΜια Εισαγωγική Παρουσίαση»*

[18] ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ, Χαράλαμπος Ν. Βλατάκης, *«Υλοποίηση Μετασχηματισμών Wavelet σε Υλικό»*

[19] Πανεπιστημιακές Σημειώσεις, Αριστείδη Προσπαθόπουλου , *«WAVELETS (ΚΥΜΑΤΙΔΙΑ) Ένα πανίσχυρο μαθηματικό εργαλείο με πολλές εφαρμογές», Απρίλιος 2009*

[20] Wavelet Functions:dwtmode, mathworks.com, http://www.mathworks.com/help/wavelet/ref/dwtmode.html

[21] Zero Crossings , en.wikipedia.org, http://en.wikipedia.org/wiki/Zero_crossing

[22] Mathematics and Logic - Skill and Concept Development ,Chapter 3: Slope Sign Analysis, whyslopes.com, http://www.whyslopes.com/4060Volume_3_Why_Slopes_-_A_Calculus_Intro_Etc/030_Chapter_3._Slope_Sign_Analysis.html