Πανεπιστημιο Πατρων
Πολυτεχνικη Σχολη
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών
και Πληροφορικής

Διπλωματική Εργασία

Υλοποίηση πλήρους συστήματος αναγνώρισης λογοτύπων

Φώτης Γ. Χαντζής
Α.Μ. 3771

Επιβλέπων: Παύλος Σπυράκης, Καθηγητής
Συνεπιβλέπων: Ιωάννης Χατζηγιαννάκης

Πάτρα, Ιούνιος 2012

Περίληψη

Η Υπολογιστική Όραση (Computer Vision) είναι ένας τομέας της επιστήμης των υπολογιστών που διερευνά την ανάπτυξη οπτικών δυνατότητων σε μια μηχανή. Περιλαμβάνει μεθόδους για την ανάκτηση, επεξεργασία, ανάλυση και κατανόηση εικόνων από τον πραγματικό κόσμο ώστε να παραχθεί συμβολική πληροφορία, κατάλληλη για την αντίστοιχη λήψη αποφάσεων από τον υπολογιστή. Είναι ένα πεδίο που γνωρίζει τα τελευταία χρόνια τεράστια άνθιση, κυρίως λόγω της συνεχώς αυξανόμενης υπολογιστικής ισχύος που επιτρέπει την ανάπτυξη αλγορίθμων που άλλοτε θα ήταν απαγορευτικά χρονοβόροι. Παραμένει όμως ένα σχετικά μεγάλο χάσμα μεταξύ της θεωρητικής έρευνας και της πρακτικής υλοποίησης και χρήσης συστημάτων που βασίζονται στις τελευταίες αυτές εφευρέσεις της υπολογιστικής όρασης. Η εργασία αυτή είναι μια απόπειρα να καλυφθεί ένα μέρος αυτού του κενού, με την υλοποίηση ενός πλήρους συστήματος αναγνώρισης λογοτύπων με επεκτάσιμη και εύκολα κλιμακούμενη αρχιτεκτονική και με αντίστοιχη εφαρμογή σε κινητά βασισμένα στην ευρέως διαδεδομένη πλατφόρμα Android της Google. Είναι γεγονός πως με την τεράστια διείσδυση των έξυπνων κινητών συσκευών στην καθημερινή ζωή μας, είναι ιδιαίτερα σημαντική η ύπαρξη δυνατότητας χρήσης μιας υπηρεσίας ανά πάσα στιγμή μέσω του κινητού. Καθώς η υπολογιστική ισχύς των κινητών συσκευών όμως παραμένει σαφώς κατώτερη συγκριτικά με αυτήν του νέφους (cloud) και οι αλγόριθμοι υπολογιστκής όρασης είναι ιδιαίτερα χρονοβόροι, η εφαρμογή επικοινωνεί με το cloud για την κάλυψη των αναγκών της αναγνώρισης του λογοτύπου. Το συνολικό σύστημα μας καθιστά δυνατή την ανάπτυξη παιγνίων με τη λογική του κυνηγιού θησαυρού, όπου η εύρεση

και φωτογράφιση συγκεκριμένου λογοτύπου σε μια τοποθεσία αποτελεί το στόχο του παιχνιδιού.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εφαρμοσμένη Υπολογιστική Όραση, Android, Software Engineering

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : SURF, SIFT, FLANN, Android development, OpenCV, C/C++, Java, Python

Ευχαριστίες

Ο κύκλος σπουδών φτάνει στο τέλος του και σηματοδοτεί την έναρξη νέων αναζητήσεων και στόχων. Στο μακρύ ταξίδι αυτό μέχρι τώρα όμως συνέβαλαν άτομα πέραν του συγγραφέα και οφείλω να τους ευχαριστήσω στο εδάφιο αυτό.

Κατάρχήν, θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Ιωάννη Χατζηγιαννάκη, για την πολύτιμη βοήθεια του σε όλη την πορεία των σπουδών μου. Η συνεργασία μας ειδικότερα στα μαθήματα Λειτουργικών Συστημάτων και Κατανεμημένων Συστημάτων υπήρξε ιδιαίτερα καρποφόρα καθώς πέραν των εκπαιδευτικών εργασιών που προέκυψαν, πυροδοτήθηκαν ενδιαφέρουσες επιστημονικές συζητήσεις και ανταλλαγές απόψεων.

Ένα τεράστιο ευχαριστώ οφείλω στο φίλο και συνάδελφο Φοίβο Κεφαλλωνίτη, με τον οποίο η εκτενέστατη και εις βάθος ανάλυση τόσο θεμάτων της επιστήμης των υπολογιστών όσο και γενικότερα της ζωής που κάναμε όλα αυτά τα χρόνια, έπαιξε καθοριστικό ρόλο στη διαμόρφωση απόψεων και στο άνοιγμα οριζόντων. Η νοοτροπία συνεχούς αναζήτησης της γνώσης, η αμφισβήτηση κάθε επιβεβλημένου στερεοτύπου και κομφορμισμού και η χρήση της φαντασίας για την επίλυση προβλημάτων ήταν αυτά που καθόρισαν τις συζητήσεις μας και διαμόρφωσαν την πορεία μας.

Θερμά ευχαριστώ από τα βάθη της καρδιάς μου θέλω να δώσω και στο άτομο (Ν.Τ) που πυροδότησε, ενίσχυσε και ενεθάρρυνε αισθήματα συμπόνοιας, ενσυναίσθησης και αγάπης προς τον κόσμο γενικότερα. Θα ήμουν τελείως διαφορετικός αν δε σε είχα γνωρίσει.

Κλείνοντας, θα ήθελα να δώσω πολλές ευχαριστίες στους γονείς μου Γι-

ώργο και Μαρία και στην αδερφή μου Ιωάννα που όλα αυτά τα χρόνια με στήριξαν ποικιλοτρόπως και μου έδωσαν τη δυνατότητα να αποκτήσω ένα ισχυρό υπόβαθρο χρήσιμο στην επιστήμη που διάλεξα να ακολουθήσω αλλά και κυριότερα στην Οδύσσεια της ζωής.

Φώτης Χαντζής

Πάτρα, 7 Μαΐου 2012

Αφιερώνεται στην Οδύσσεια του καθενός που προσπαθεί για ένα συνολικό καλύτερο αύριο.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

> It had long since come to my attention that people of accomplishment rarely sat back and let things happen to them. They went out and happened to things.
>
> Leonardo da Vinci

1.1 Πρόβλημα και Σημασία

Ως άνθρωποι αντιλαμβανόμαστε την τρισδιάστατη δομή του κόσμου γύρω μας με φαινομενική ευκολία. Μπορούμε πολύ εύκολα να διακρίνουμε τα πρόσωπα από τα αντικείμενα, τα αντικείμενα από το φόντο, να αναγνωρίσουμε μοναδικά χαρακτηριστικά του εκάστοτε προσώπου ή αντικειμένου και να τα αντιστοιχίσουμε με βάση την προϋπάρχουσα γνώση για τον κόσμο γύρω μας. Σε αντίθεση με μας, ο υπολογιστής δεν μπορεί να επιτελέσει τις παραπάνω λειτουργίες με την ίδια ευχέρεια. Η Υπολογιστική Όραση, ως αντικείμενο μελέτης έχει την ανάκτηση από εικόνες και φωτογραφίες εκείνης της συμβολικής πληροφορίας που θα καθιστά δυνατό για μια μηχανή να προσεγγίσει τις δυνατότητες που έχει ο συνδυασμός του ανθρώπινου οπτικό συστήματος και ο νους. Χρησιμοποιεί μαθηματικές τεχνικές και στατιστικά μοντέλα για να αναλύσει την τρισδιάστατη σκηνή που συνήθως αναπαρίσταται σε μια δυσδιάστατη εικόνα και να εξάγει χρήσιμη πληροφορία με σκοπό να γίνει όσο

το δυνατόν μεγαλύτερη ¨κατανόηση¨ εκ μέρους της μηχανής για το αντικείμενο που παρουσιάζεται. Μερικές από τις εφαρμογές της επιστήμης είναι η Αναγνώριση Αντικειμένων, η Οπτικη αναγνώριση χαρακτήρων, η 3-D ανακατασκευή μοντέλων από εικόνες, η συρραφή πολλαπλών εικόνων, η ανίχνευση και αναγνώριση προσώπων και πολλές ακόμα. Γίνεται αντιληπτό πως ο κλάδος αυτός διεισδύει όλο και περισσότερο στην καθημερινότητα μας και στο εγγύς μέλλον η σημασία και επιρροή του θα είναι αδιαμφισβήτητη, αν δεν είναι ήδη σήμερα. Επιπλέον, η ευκολία με την οποία μπορεί ο καθένας να ψηφιοποιήσει οποιαδήποτε σκηνή της αρεσκείας του με το απλό πάτημα ενός πλήκτρου, φέρνει τον απλό καθημερινό χρήστη όλο και πιο κοντά στη σφαίρα επιρροής των εφαρμογών της Υπολογιστικής Όρασης.

Μια από τις πιο ευρέως χρησιμοποιούμενες εφαρμογές της Υπολογιστικής Όρασης είναι η *Αναγνώριση Αυτικειμένων*. Ο κλάδος αυτός ασχολείται με την ανάπτυξη των τεχνικών εκείνων που επιτρέπουν σε μιας μηχανή να ανιχνεύσει και να αναγνωρίσει συγκεκριμένα σε μια εικόνα ή βίντεο. Αν η μηχανή έχει αρχικά ¨μάθει¨ βάση ενός μοντέλου αναπαράστασης, λόγου χάριν πως ένα βάζο πάνω σε ένα τραπέζι έχει ένα συγκεκριμένο σχήμα και έχει ορισμένα μοναδικά χαρακτηριστικά που το διαχωρίζουν από την υπόλοιπη σκηνή, αν αργότερα ξαναπαρουσιαστεί το ίδιο βάζο από μια άλλη φωτογραφία, θα μπορέσει να ξανα-αναγνωρίσει εκείνα τα χαρακτηριστικά που το κάνουν μοναδικό ώστε να το καταχωρήσει ως το βάζο που είδε στην αρχική εικόνα; Η δυσκολία έγκειται στο ότι η φωτογραφία του αντικειμένου μπορεί να προέρχεται από διαφορετική γωνία λήψης, υπό διαφορετικές συνθήκες φωτισμού, με παρουσία θορύβου και γεωμετρικών και άλλων αλλοιώσεων. Τα παραπάνω κάνουν ιδιαίτερα επίπονη τη διαδικασία εξαγωγής πληροφορίας που μένει αναλοίωτη κάτω από όλες αυτές τις συνθήκες. Η έρευνα που γίνεται επικεντρώνεται αφενός στην ακρίβεια του αλγορίθμου εξαγωγής αυτής της πληροφορίας αφετέρου στον όσο το δυνατόν ταχύτερο υπολογισμού της. Όπως θα δούμε παρακάτων ο SURF - Speeded Up Robust Features είναι ένας τέτοιος αλγόριθμος που συνδυάζει και τα δύο αυτά προσόντα. Κατόπιν, τίθεται επίσης το ζήτημα της γρήγορης αντιστοίχισης των εικόνων από μια τράπεζα εικόνων βάση της προηγούμενης εξαχθείσας πληροφορίας.

Στη σημερινή εποχή, κατακλυζόμαστε καθημερινώς από την παρουσία των εικόνων εκείνων που διαχωρίζουν την μια εταιρεία από την άλλη και που

Σχήμα 1.1: Τα λογότυπα αποτελούν σημαντικό σημείο για την προώθηση και διαφήμιση μιας εταιρείας

κωδικοποιούν την αναγνωρίσιμη ταυτότητα τους. Πρόκειται για τα λεγόμενα λογότυπα (logos) που αποτελούν το πιο σημαντικό ίσως σημείο αναφοράς για την προώθηση και διαφήμιση μιας εταιρείας (1.1). Η αναγνώριση της εκάστοτε ¨μάρκας¨ γίνεται άμεσα από τον ανθρώπινο νου, αφού έχει συνδυάσει και συνήθως ταυτίσει το βασικό προϊόν της εταιρείας με το σχήμα, το χρώμα, τα γράμματα και οτιδήποτε άλλο κάνει ξεχωριστό το λογότυπο.

Το λογότυπο από την πλευρά του υπολογιστή δεν είναι τίποτα παραπάνω από ένα ακόμα αντικείμενο. Δηλαδή, είναι μια εικόνα που διέπεται από συγκεκριμένο πρότυπο pattern κατανομή πληροφορίας. Συνεπώς, οι ίδιοι κανόνες και τεχνικές που εφαρμόζονται για την αναγνώριση αντικειμένων μπορούν πρακτικά να εφαρμοστούν και για την *Αναγνώριση Λογοτύπων* με την οποία ασχολείται η παρούσα Διπλωματική Εργασία.

Είναι γεγονός πως μέχρι τώρα δεν υπάρχει διαθέσιμο στο ευρύ κοινό κάποιο ολοκληρωμένο σύστημα αναγνώρισης λογοτύπων. Φυσικά, υπάρχουν εταιρείες που ενδεχομένως αναπτύσσουν εσωτερικά για δική τους χρήση παρόμοια συστήματα. Παραμένουν όμως σχεδόν πάντα λογισμικά κλειστού κώδικα και διαθέσιμα μόνο κατά παραγγελία από άλλες εταιρείες. Η Διπλωματική Εργασία αυτή θα αποτελέσει μάλλον το πρώτο ολοκλήρωμενο σύστημα αναγνώρισης λογοτύπων που σε συνδυασμό με την εφαρμογή για κινητά

Ανδροιδ που αναπτύξαμε, προετοιμάζουν το έδαφος για ακόμα και μια ευρεία διάθεση του στο κοινό.

1.2 Στόχος Διπλωματικής Εργασίας

Ο βασικός στόχος είναι η μελέτη της πρακτικότητας των πιο σύγχρονων αλγορίθμων αναγνώρισης αντικειμένων και επομένως αναγνώρισης λογοτύπων με την εφαρμογή τους στο πλήρες σύστημα που αναπτύξαμε. Ο καιρός είναι ο πλέον κατάλληλος για την άμεση χρήση των αλγορίθμων αυτών, αφού πλέον η υπολογιστική ισχύς των ημερών μας, μας επιτρέπει διαδραστικότητα σχεδόν πραγματικού χρόνου ακόμα και για τόσο απαιτητικές διαδικασίες όπως η αναγνώριση αντικειμένων. Θα δουμε έμπρακτα την καταλληλότητα του νεότερου αλγορίθμου του κλάδου αυτού και τις δυνατότητες που ανοίγονται στο χώρο της εφαρμοσμένης Υπολογιστικής Όρασης.

1.3 Σκοπός και Συνεισφορά της Διπλωματικής Εργασίας

Όπως αναφέρθηκε προηγούμενως, το λογισμικό που αναπτύξαμε πρόκειται για το πρώτο *ολοκληρωμένο* σύστημα αναγνώρισης λογοτύπων που θα είναι διαθέσιμο προς το ευρύ κοινό. Η συνεισφορά έγγυται όχι τόσο στην επισκόπηση του θεωρητικού υποβάθρου που καθιστά δυνατή τη λειτουργία ενός τέτοιου συστήματος (το οποίο προφανώς έχει τη δική του βαρύτητα και εξετάζεται ξεχωριστά παρακάτω), όσο πρωτίστως στην ανάλυση της χρηστικότητας ενός προγράμματος που μόλις ελάχιστα χρόνια πριν δεν θα μπορούσε να υπάρχει βάση των τότε υπολογιστικών δυνατοτήτων και της τοτε υπάρχουσας έρευνας. Εξετάζονται σημαντικές έννοιες για την αρχιτεκτονική και την τεχνολογία του λογισμικού που αναπτύξαμε, κάτι που δοκιμάζει ουσιαστικά το συνδυασμό βασικών αρχών software engineering με πρωτοπόρα θεωρητική έρευνα (βλέπε αλγόριθμο SURF) και το δέσιμο των παραπάνω σε ένα πλήρες πρακτικό σύστημα. Παρουσιάζονται παράλληλα, σύγχρονα πακέτα λογισμικού όπως το OpenCV, που περιέχει βιβλιοθήκες με αλγορίθμους Υπολογιστικής Όρασης και το Android, η διάσημη πλατφόρμα της Google για

έξυπνες κινητές συσκευές.

1.4 Οργάνωση Διπλωματικής Εργασίας

Η παρούσα Διπλωματική Εργασία δομείται ως εξής:

Στο Κεφάλαιο 2: Speeded-Up Robust Features παραθέτουμε το θεωρητικό υπόβαθρο πίσω από την εφεύρεση του βασικού αλγορίθμου (SURF) που χρησιμοποιεί το σύστημα μας. Το κεφάλαιο αυτό δεν απαιτείται για την κατανόηση της υπόλοιπης ανάλυσης αφού η εργασία μας εστιάζει στο πιο πρακτικό μέρος της ανάπτυξης του συστήματος. Παρ'όλα αυτά, για μια πιο καθολική επισκόπηση του αντικειμένου, ο αναγνώστης καλείται να το μελετήσει ξεχωριστά. Η συγγραφή του συγκεκριμένου κεφαλαίου δανείζεται εκτενώς από την επίσημη δημοσίευση του SURF [4].

Στη συνέχεια, στο Κεφάλαιο 3: Byakugan - ένα πλήρες σύστημα αναγνώρισης λογοτύπων αναλύεται η αρχιτεκτονική του συστήματος μας. Παρουσιάζονται και μελετώνται λεπτομερώς τα τμήματα εκείνα του κώδικα που χρίζουν σημασίας για την κατανόηση της λειτουργίας του λογισμικού.

Το Κεφάλαιο 4: Εκτέλεση και Αποτελέσματα παρουσιάζει την έμπρακτη λειτουργία του συστήματος, δίνοντας έμφαση στους χρόνους διαδρασης και αλληλεπίδρασης καθώς και τα αποτελέσματα της συνολικής χρηστικότητας και καταλληλότητας του για ευρεία χρήση. Παρουσιάζονται επίσης οι δυνατότητες χρήσης σε παίγνια και οι πιθανές επεκτάσεις του.

Το Κεφάλαιο 5: Υλοποίηση αποτελεί την παράθεση του συνολικού κώδικα του συστήματος - τόσο του εξυπηρετητή server όσο και του πελάτη client.

Κεφάλαιο 2

Speeded-Up Robust Features

> You don't understand anything until you learn it more than one way.
>
> Marvin Minsky

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τη δομή και λειτουργία του αλγορίθμου που προτιμήθηκε για το σύστημα αναγνώρισης λογοτύπων που δημιουργήσαμε. Ο αλγόριθμος λέγεται SURF από τα αρχικά Speeded-Up Robust Features και εφευρέθηκε απο τους Herbert Bay, Andreas Ess, Tinne Tuytelaars και Luc Van Gool το 2006 [4]. Παρ'όλη τη σχετική νεότητα του, χρησιμοποιείται ήδη ευρέως σε ποικίλες εφαρμογές του τομέα της υπολογιστικής όρασης όπως η αναγνώριση αντικειμένων και η 3D-ανακατασκευή.

Ο βασικός στόχος του αλγορίθμου είναι η εύρεση διακριτών σημείων αντιστοίχισης μεταξύ εικόνων. Η διαδικασία απαρτίζεται από τρία βασικά βήματα.

1. Ίσημεία ενδιαφέροντος΄ επιλέγονται σε συγκεκριμένες τοποθεσίες της εικόνας όπως γωνίες, blobs, T-διασταυρώσεις. Η πιο σημαντική ιδιότητα ενός άνιχνευτή΄ σημείων ενδιαφέροντος είναι η επαναληψιμότητα. Η επαναληψιμότητα εκφράζει την αξιοπιστία ενός ανιχνευτή στην εύρε-

ση των ίδιων σημείων ενδιαφέροντος κάτω από διαφορετικές συνθήκες προβολής.

2. Η περιοχή κάθε σημείου ενδιαφέροντος εκπροσωπείται από ένα χαρακτηριστικό διάνυσμα. Ο 'περιγραφεας' πρέπει να είναι διακριτικός και παράλληλα ανθεκτικός στο θόρυβο, σε μετατοπίσεις ανίχνευσης και γεωμετρικές και φωτομετρικές παραμορφώσεις.

3. Τέλος, τα χαρακτηριστικά διανύσματα αντιπαραβάλλονται μεταξύ διαφορετικών εικόνων. Η αντιστοίχιση βασίζεται σε μια απόσταση μεταξύ των διανυσμάτων όπως την Mahalanobis ή την Ευκλειδιανή. Η διάσταση του περιγραφέα έχει άμεση επίδραση στο χρόνο και μικρότερες διαστάσεις είναι πιο επιθυμητές για γρήγορη αντιστοίχιση των σημείων ενδιαφέροντος. Όμως, όπως αναμενόταν, τα μικρότερων διαστάσεων χαρακτηριστικά διανύσματα είναι γενικά λιγότερα ξεχωριστά σε σύγκριση με αυτά που έχουν παραπάνω διαστάσεις.

Ο στόχος είναι τόσο ο ανιχνευτής όσο και ο περιγραφέας να είναι αρκετά γρήγορο να υπολογιστούν χωρίς να θυσιάζεται η ακρίβεια. Η ισορροπία επιτυγχάνεται με το να απλοποιείται ο τρόπος ανίχνευσης διατηρώντας τον ακριβή και μειώνοντας το μέγεθος του περιγραφέα διατηρώντας τον ικανοποιητικά διακριτικό.

Ο αλγόριθμος SURF εστιάζει σε ανιχνευτές και περιγραφείς που είναι αμετάβλητοι σε σχέση με την κλίμακα και την περιστροφή της εικόνας. Αυτη η ιδιότητα προσφέρει έναν καλό συμβιβασμό μεταξύ της συνθετότητας των χαρακτηριστικών σημείων και της ανθεκτικότητας στις πιο συχνά εμφανιζόμενες παραμορφώσεις.

2.2 Σχετική Έρευνα

2.2.1 Ανίχνευση σημείων ενδιαφέροντος

Ο πιο ευρέως χρησιμοποιούμενος ανιχνευτής είναι ο ανιχνευτής γωνιών Harris που προτάθηκε το 1988 [3]. Βασίζεται στις ιδιοτιμές του μητρώου ροπής δεύτερης τάξης. Όμως οι γωνίες Harris δεν είναι αμετάβλητες ως προς

την κλίμακα. Ο Lindeberg [7] εισήγαγε την έννοια της αυτόματης επιλογής κλίμακας. Αυτό επιτρέπει την ανίχνευση σημείων ενδιαφέροντος σε μια εικόνα, το καθένα με τη δική του χαρακτηριστική κλίμακα. Πειραματίστηκε τόσο με την ορίζουσα ενός Εσσιανού μητρώου όσο και με το Λαπλασιανό, που αντιστοιχεί με το ίχνος ενός Εσσιανού. Οι Mikolajczyk και Schmid [10] βελτίωσαν αυτή τη μέθοδο δημιουργώντας ανθεκτικούς και αμετάβλητους ως προς την κλίμακα ανιχνευτές με υψηλή επαναλειψιμότητα. Χρησιμοποίησαν ένα μέτρο Harris ή την ορίζουσα ενός Εσσιανού μητρώου για την επιλογή τοποθεσίας και τη Λαπλασιανή για την επιλογή της κλίμακας. Εστιάζοντας στην ταχύτητα, ο Lowe [8] πρότεινε την προσέγγιση της Λαπλασιανής των Γκαουσιανών χρησιμοποιώντας ένα φίλτρο Διαφοράς Γκαουσιανών.

Από μελέτη των υπαρκτών ανιχνευτών που έγινε από τους Herbert Bay et al, προέκυψε πως οι βασιζόμενοι σε Εσσιανά μητρώα ανιχνευτές είναι πιο σταθεροί και επαναληπτοί σε σύγκριση με τους αντίστοιχους βασιζόμενους σε Harris μητρώα. Επιπλέον, χρησιμοποιώντας την ορίζουσα του Εσσιανού μητρώου αντί για το ίχνος φαίνεται επωφελές. Επίσης, οι προσεγγίσεις όπως οι Διαφορές Γκαουσιανών μπορού να επιφέρουν ταχύτητα με μικρό κόστος ως προς την ακρίβεια.

2.2.2 Περιγραφή σημείων ενδιαφέροντος

Μια ακόμα μεγαλύτερη ποικιλία από περιγραφείς χαρακτηριστικών έχει προταθεί. Οι περιγραφείς που εκπροσωπούν την κατανομή των μικρότερης κλίμακας χαρακτηριστικών μέσα στη γειτονιά των σημείων ενδιαφέροντος εισήχθηκαν από τον Lowe [9] και έχει δειχθεί ότι υπερτερούν έναντι των υπολοίπων [11]. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι συλλαμβάνουν ένα μεγάλο ποσό πληροφορίας για τα πρότυπα της χωρικής έντασης, ενώ παράλληλα ώντας ανθεκτικά σε μικρές παραμορφώσεις και τοπικά λάθη. Ο περιγραφέας του άλλου γνωστού αλγορίθμου SIFT - Scale Invariant Feature Transform [9] υπολογίζει ένα ιστόγραμμα των τοπικών κλίσεων γύρω από τα σημεία ενδιαφέροντος και αποθηκεύει τα αποτελέσματα σε 128-διάστατα διανύσματα.

Ο περιγραφέας SIFT είναι ακόμα από τους πιο ελκυστικούς περιγραφείς για πρακτική χρήση και ένας από τους πιο ευρέως χρησιμοποιούμενους πριν την εμφάνιση του SURF. Είναι διακριτικός και σχετικά γρήγορος, πράγμα

κρίσιμο για εφαρμογές πραγματικού χρόνου. Όμως η υψηλή διαστατικότητα του συγκεκριμένου περιγραφέα είναι μειονέκτημα για το βήμα της αντιστοίχισης. Για τη βελτίωση της ταχύτητας της αντιστοίχισης ο SURF χρησιμοποιεί το ίχνος του Εσσιανού μητρώου.

2.3 Ανίχνευση σημείων ενδιαφέροντος

Η προσέγγιση για την ανίχνευση σημείων ενδιαφέροντος χρησιμοποιεί μια πολλή βασική προσέγγιση Εσσιανού μητρώου. Κάνει χρήση των έικόνων ολοκλήρωμα′ (integral images) που έγιναν γνωστές από τους Viola και Jones [14] που μειώνει δραματικά το χρόνο υπολογισμού.

2.3.1 Εικόνες ολοκλήρωμα

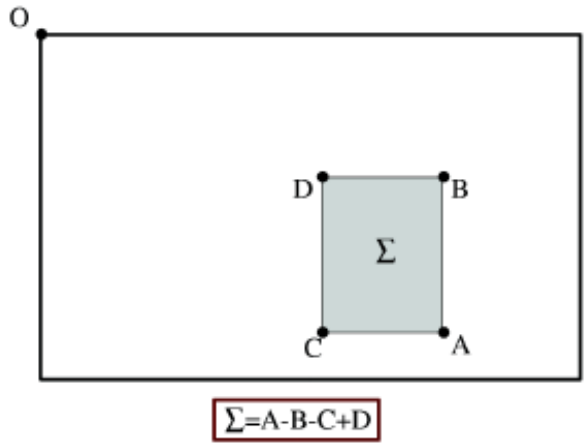
Οι εικόνες ολοκλήρωμα χρησιμοποιούνται για το γρήγορο υπολογισμό συνελικτικών φίλτρων. Η είσοδος μιας εικόνας ολοκλήρωμα $I_\Sigma(x)$ σε μια τοποθεσία $x = (\chi, y)^T$ αντιπροσωπεύει το άθροισμα όλων των εικονοκυττάρων στην εικόνα εισόδου I μέσα σε μια ορθογώνια περιοχή σχηματισμένη από την αρχή και το x.

$$I_\Sigma(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{2.1}$$

Όταν η εικόνα ολοκλήρωμα υπολογιστεί, παίρνει επιπρόσθετα τρείς προσθέσεις για να υπολογιστεί το άθροισμα των εντάσεων πάνω σε μια ορθογώνια περιοχή. (δες εικόνα 2.1) Έτσι ο χρόνος υπολογισμού είναι ανεξάρτητος από το μέγεθος. Αυτό είναι ιδιαίτερα σημαντικό καθώς στην περίπτωση του SURF χρησιμοποιούνται φίλτρα μεγάλου μεγέθους.

2.3.2 Σημεία ενδιαφέροντος βασισμένα σε Εσσιανά μητρώα

Ο ανιχνευτής του SURF βασίζεται στο Εσσιανό μητρώο λόγω της καλής απόδοσης στην ακρίβεια. Ανιχνεύει blobs σε περιοχές όπου η ορίζουσα είναι μέγιστη. Σε σύγκριση με τον Εσσιανό-Λαπλασιανό ανιχνευτή των Mikolajczyk



Σχήμα 2.1: Χρησιμοποιώντας εικόνες ολοκλήρωμα, παίρνει μόνο 3 προσθέσεις και 4 προσβάσεις στη μνήμη για τον υπολογισμό των εντάσεων μιας ορθογώνιας περιοχής οποιουδήποτε μεγέθους

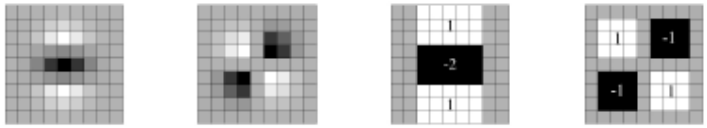
και Schmid [10], βασίζεται στην ορίζουσα του Εσσιανού και για την επιλογή κλίμακας όπως η εργασία του Lindeberg [7].

Δοθέντος ενός σημείου $x = (\chi, y)$ σε μια εικόνα I, το Εσσιανό μητρώο $H(x, \sigma)$ στην κλίμακα σ ορίζεται ως εξής:

$$H(x, \sigma) = \begin{bmatrix} L_{\chi\chi}(x, \sigma) & L_{\chi y}(x, \sigma) \\ L_{\chi y}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{2.2}$$

όπου το $L_{\chi\chi}(x, \sigma)$ είναι η συνέλιξη της δεύτερης τάξης παραγώγου της Γκαουσιανής $\frac{\partial^2}{\partial(x)^2} g(\sigma)$ με την εικόνα I στο σημείο x και ομοίως για τα $L_{\chi y}(x, \sigma)$ και $L_{yy}(x, \sigma)$.

Οι Γκαουσιανές είναι βέλτιστες για ανάλυση του χώρου κλίμακας [5] [6] αλλά στην πράξη πρέπει να διακριτοποιηθούν και να περικοπούν. Αυτό οδηγεί σε απώλεια στην επαναληψιμότητα σε περιστροφές της εικόνας κατά περιττά πολλαπλάσια του $\frac{\pi}{4}$. Αυτή η αδυναμία ισχύει γενικά για ανιχνευτές βασισμένους σε Εσσιανά μητρώα. Η επαναληψιμότητα φθάνει στο μέγιστο σε πολλαπλάσια του $\frac{\pi}{2}$. Αυτό είναι λόγω του τετράγωνου σχήματος του φίλτρου. Παρ'όλα αυτά, οι ανιχνευτές αποδίδουν καλά και η μικρή μείωση της επίδοσης δεν επισκιάζει το πλεονέκτημα των γρήγορων συνελίξεων που επιφέρονται από την διακριτοποίηση και την περικοπή. Αφού τα πραγματικά φίλτρα ε-



Σχήμα 2.2: Αριστερά προς δεξιά: ο διακριτοποιημένες και περικομμένες δεύτερης τάξης Γκαουσιανές παράγωγοι στην y και xy-κατεύθυνση αντίστοιχα· η προσέγγιση με χρήση φίλτρων-κουτιά για τις ίδιες κατευθύνσεις.

ίναι μη ιδανικά σε κάθε περίπτωση και δεδομένης της επιτυχίας του Lowe με τις προσεγγίσεις των Λαπλασιανών των Γκαουσιανών, η προσέγγιση για το Εσσιανό μητρώο πάει ένα βήμα πιο πέρα με τη χρήση φίλτρων-κουτιά (δεξί μέρος της 2.2). Αυτά προσεγγίζουν δεύτερης τάξης Γκαουσιανές παραγώγους και μπορούν να εκτιμηθούν με χαμηλό υπολογιστικό κόστος χρησιμοποιώντας εικόνες-ολοκλήρωμα. Έτσι, ο χρόνος υπολογισμού είναι ανεξάρτητος του μεγέθους φίλτρου.

Τα $9\chi9$ φίλτρα κουτιά στην 2.2 είναι προσεγγίσεις μιας Γκαουσιανής με διασπορά $\sigma = 1.2$ και αναπαριστούν τη χαμηλότερη κλίμακα για τον υπολογισμό των blob χαρτών. Ονομάζονται D_{xx}, D_{yy} και D_{xy}. Τα βάρη που εφαρμόζονται στις ορθογώνιες περιοχές διατηρούνται απλά για λόγους υπολογιστική επίδοσης. Αυτό δίνει:

$$det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.3}$$

Το σχετικό βάρος w των αποκρίσεων των φίλτρων χρησιμοποιείται για την εξισορρόπηση της έκφρασης για την Εσσιανή ορίζουσα. Αυτό χρειάζεται για την διατήρηση ενέργειας μεταξύ των Γκαουσιανών πυρήνων και των προσεγγισμένων Γκαουσιανών πυρήνων,

$$w = \frac{|L_{xy}(1.2)|_F \, |D_{yy}(9)|_F}{|L_{yy}(1.2)|_F \, |D_{xy}(9)|_F} = 0.912... \simeq 0.9 \tag{2.4}$$

όπου $|\chi|_F$ είναι η νόρμα Frobenius. Το βάρος μπορεί να αλλάξει ανάλογα με την κλίμακα, όμως και αν διατηρηθεί σταθερό, πρακτικά δεν θα έχει μεγάλη επίδραση.

Επιπρόσθετα, οι αποκρίσεις των φίλτρων κανονικοποιούνται σε σχέση με

το μέγεθός τους. Αυτό εξασφαλίζει μια σταθερή νόρμα Frobenius για κάθε μέγεθος φίλτρου, κάτι σημαντικό για την ανάλυση στο χώρο κλίμακας.

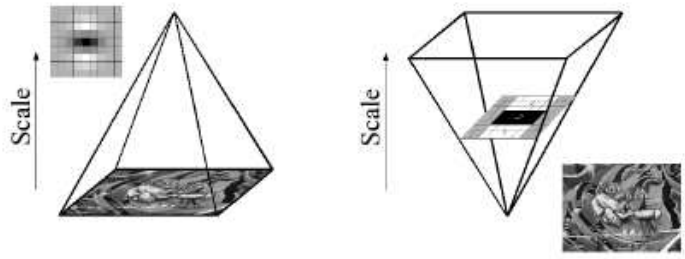
Η προσεγγισμένη Εσσιανή ορίζουσα εκπροσωπεί την απόκριση blob της εικόνας στο σημείο χ. Αυτές οι αποκρίσεις αποθηκεύονται σε ένα χάρτη απόκρισης blob για διαφορετικές κλίμακες και ανιχνεύονται τοπικά μέγιστα όπως εξηγείται παρακάτω.

2.3.3 Αναπαράσταση χώρου κλίμακας

Σημεία ενδιαφέροντος χρειάζεται να βρεθούν για διαφορετικές κλίμακες, αν μη τι άλλο διότι η αναζήτηση αντιστοιχιών συχνά απαιτεί τη σύγκριση σε εικόνες με διαφορετικές κλίμακες. Οι χώροι κλίμακας συνήθως υλοποιούνται ως εικόνες πυραμίδα. Οι εικόνες εξομαλύνονται επαναλαμβανόμενα με μια Γκαουσιανή και μετά υποδειγματολειπτούνται ώστε να επιτευχθεί ένα ανώτερο επίπεδο της πυραμίδας. Ο Lowe στο [9] αφαιρεί αυτά τα επίπεδα πυραμίδας ώστε να πάρει τις εικόνες Διαφορά των Γκαουσιανών όπου οι ακμές και τα blobs μπορούν να βρεθούν.

Λόγω της χρήσης των φίλτρων κουτιά και των εικόνων-ολοκλήρωμα, δε χρειάζεται να εφαρμοστεί επαναληπτικά το ίδιο φίλτρο στην έξοδο ενός προηγουμένως φιλτραρισμένου επιπέδου, αλλά αντί αυτού μπορεί να εφαρμοστούν φίλτρα κουτιά οποιουδήποτε μεγέθους στην ακριβώς ίδια ταχύτητα απεύθειας στην αρχική εικόνα. Επομένως, ο χώρος κλίμακας αναλύεται με το να αυξάνεται η κλίμακα του μεγέθους του φίλτρου αντί να μειώνεται επαναληπτικά το μέγεθος της εικόνας (δες εικόνα 2.3). Η έξοδος του $9\chi9$ φίλτρου, όπως αναφέρθηκε πριν, θεωρείται ως το αρχικό επίπεδο κλίμακας το οποίο θα αναφέρεται ως κλίμακα $s = 1.2$ (προσεγγίζοντας Γκαουσιανές παραγώγους με $\sigma = 1.2$). Τα ακόλουθα επίπεδα αποκτώνται με το να φιλτράρεται η εικόνα με σταδιακά μεγαλύτερες μάσκες, λαμβάνοντας υπόψιν τη διακριτή φύση των εικόνων-ολοκλήρωμα και την συγεκριμένη δομή των φίλτρων.

Το κύριο κίνητρο για αυτό το είδος της δειγματοληψίας είναι η υπολογιστική επίδοση. Επιπλέον, αφού δε χρειάζεται να υποδειγματοληπτηθεί η εικόνα, δεν προκύπτει αναδίπλωση συχνοτήτων. Το μειονέκτημα είναι ότι τα φίλτρα κουτιά διατηρούν τις συνιστώσες υψηλών συχνοτήτων που μπορούν να χαθούν σε παραλαγγές της ίδιας σκηνής που έχουν υποστεί σμίκρυνση,

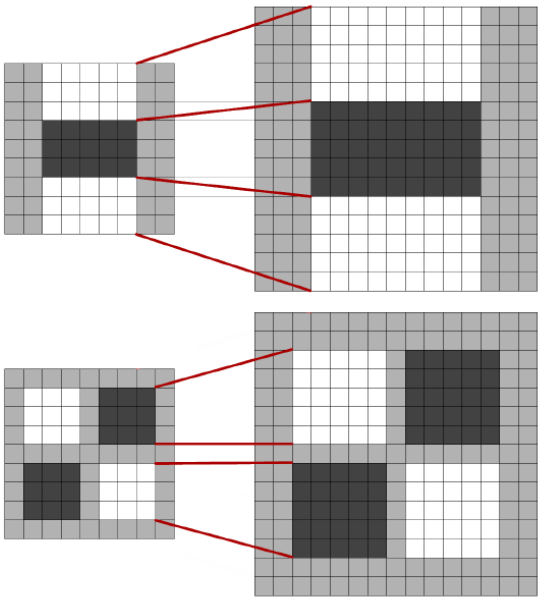


Σχήμα 2.3: Αντί για επαναληπτική μείωση της εικόνας (αριστερά), η χρήση εικόνων-ολοκλήρωμα επιτρέπει την αύξηση της κλίμακας του φίλτρου με σταθερό κόστος (δεξιά)

πράγμα που μπορεί ενδεχομένως να περιορίσει την αμεταβλητότητα ως προς την κλίμακα. Παρ'όλα αυτά αυτό δεν είναι ιδιαίτερα παρατηρήσιμο.

Ο χώρος κλίμακας διαιρείται σε οκτάβες. Μια οκτάβα εκπροσωπεί μια σειρά από χάρτες απόκρισης φίλτρων που έχουν αποκτηθεί από τη συνέλιξη της ίδιας εικόνας με ένα φίλτρο αυξανόμενου μεγέθους. Συνολικά, μια οκτάβα περικλείει έναν παράγοντα κλίμακας (που υποννοεί πώς κάποιος χρειάζεται κάτι περισσότερο από το να διπλασιάσει το μέγεθος του φίλτρου). Κάθε οκτάβα υποδιαιρείται σε σταθερό αριθμό από επίπεδα κλίμακας. Λόγω της διακριτής φύσης των εικόνων-ολοκλήρωμα, η μικρότερη διαφορά κλίμακας μεταξύ δύο διαδοχικών κλιμάκων εξαρτάται από το μήκος l_0 των θετικών ή αρνητικών λοβών της μερικής δεύτερης τάξης παραγώγου στην κατεύθυνση της παραγώγισης (x ή y), που τίθεται στο $\frac{1}{3}$ του μεγέθους του μήκους του φίλτρου. Για το $9x9$ φίλτρο, αυτό το μήκος l_0 είναι 3. Για δύο διαδοχικά επίπεδα, πρέπει να αυξηθεί το μέγεθος κατά ελάχιστο 2 εικονοστοιχείων (1 εικονοστοιχείο σε κάθε πλευρά) ωτε να διατηρηθεί το μέγεθος άνισο και έτσι να εξασφαλίσει την παρουσία του κεντρικού εικονοστοιχείου. Αυτό συνεπάγεται σε συνολική αύξηση του μεγέθους της μάσκας κατά 6 εικονοστοιχεία (δες 2.4). Για διαστάσεις διαφορετικές από l_0 (π.χ το πλάτος της κεντρικής ζώνης του κατακόρυφου φίλτρου στην 2.4), αν αλλάξει η κλίμακα της μάσκας θα εισαχθούν σφάλματα στρογγυλοποίησης. Παρ'όλα αυτά, αφού αυτά τα σφάλματα είναι τυπικά αρκετά μικρότερα από το l_0, είναι αποδεκτή προσέγγιση.

Η κατασκευή του χώρου κλίμακας ξεκινάει με το $9x9$ φίλτρο, το οποίο υπολογίζει την απόκριση σταγόνα (blob) της εικόνας στη μικρότερη κλίμα-



Σχήμα 2.4: Φίλτρα D_{yy} (πάνω) και D_{xy} (κάτω) για δύο διαδοχίκα επίπεδα κλίμακας (9 x 9 και15 x 15). Το μήκος του σκοτεινού λοβού μπορεί να αυξηθεί μόνο από ένα άρτιο αριθμό εικονοστοιχείων ώστε να εξσσφαλίζει την παρουσία κεντρικού εικονοστοιχείου (πάνω).

κα. Μετά τα φίλτρα με μεγέθη 15x15, 21x21 και 27x27 εφαρμόζονται, με τα οποία ακόμα περισσότερο από μια αλλαγή κλίμακας κατά 2 έχει επιτευχθεί. Αλλά αυτό χρειάζεται, αφού μια 3D μη-μέγιστη κατάπνιξη εφαρμόζεται τόσο χωρικά όσο και σε γειτονικές κλίμακες. Έτσι, οι πρώτοι και τελευταίοι χάρτες απόκρισης Εσσιανής στη στοίβα περιέχουν τέτοια μέγιστα, αφού χρησιμοποιούνται για λόγους σύγκρισης μόνο. Συνεπώς, μετά από παρεμβολή, η μικρότερη δυνατή κλίμακα είναι $\sigma = 1.6 = 1.2\frac{12}{9}$ που αντιστοιχεί σε ένα φίλτρο μεγέθους 12x12 και η υψηλότερη σε $\sigma = 3.2 = 1.2\frac{24}{9}$.

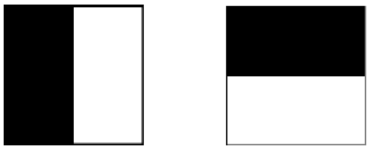
2.3.4 Εντοπισμός σημείων ενδιαφέροντος

Προκειμένου να εντοπιστούν σημεία ενδιαφέροντος στην εικόνα και σε διάφορες κλίμακες, εφαρμόζεται μια μη-μέγιστη κατάπνιξη σε μια 3x3 γειτονιά. Συγκεκριμένα χρησιμοποιείται μια παραλλαγή των Neubeck και Van Gool [13]. Τα μέγιστα της ορίζουσας του Εσσιανού μητρώου παρεμβάλονται μετά στο χώρο κλίμακας και εικόνας με τη μέθοδο που πρότειναν οι Brown et al [1].

Η παρεμβολή στο χώρο κλίμακας είναι ιδιαίτερα σημαντική, αφού η διαφορά μεταξύ των πρώτων επιπέδων της κάθε οκτάβας είναι σχετικά μεγάλη.

2.4 Περιγραφή σημείων ενδιαφέροντος και Αντιστοίχιση

Ο περιγραφέας του SURF περιγράφει την κατανομή του περιεχομένου έντασης μέσα στην γειτονία του σημείου ενδιαφέροντος, παρόμοια με την πληροφορία κλίσης που εξάγεται από τον SIFT [9] και τις παραλλαγές του. Χρησιμοποιεί την κατανομή των αποκρίσεων πρώτης τάξης Haar κυματιδίων στην x και y διεύθυνση αντί για την κλίση, εκμεταλλεύεται τις εικόνες ολοκλήρωμα για ταχύτητα και χρησιμοποιεί μόνο 64 διαστάσεις. Αυτό μειώνει το χρόνο για τον υπολογισμό χαρακτηριστικών και την αντιστοίχιση και έχει αποδειχτεί πως παράλληλα αυξάνει τη στιβαρότητα. Επιπλέον, κάνει χρήση ενός νέου βήματος ευρετηρίασης βασισμένο στο πρόσημο της Λαπλασιανής, που αυξάνει όχι μόνο τη στιβαρότητα του περιγραφέα, αλλά και την ταχύτητα αντιστοίχισης (κατά ένα παράγοντα 2 στην καλύτερη περίπτωση).



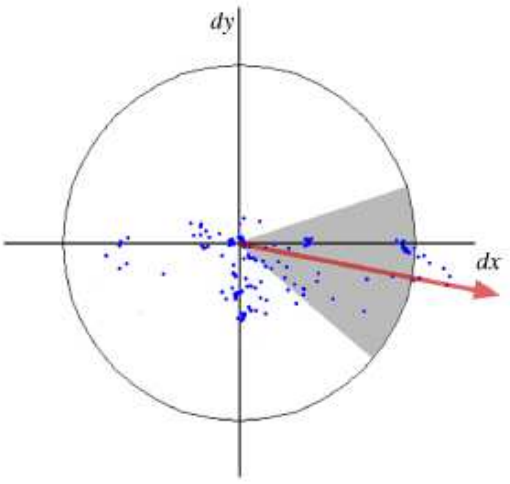
Σχήμα 2.5: Φίλτρα κυματιδίων Haar για τον υπολογισμό των αποκρίσεων στην x κατεύθυνση (αριστερά) και στην y κατεύθυνση (δεξιά). Τα σκοτεινά σημεία έχουν βάρος -1 και τα φωτεινά +1.

Το πρώτο βήμα αποτελείται από τον καθορισμό ενός αναπαραγώγιμου προσανατολισμού βασισμένου στην πληροφορία από μια κυκλική περιοχή γύρω από το σημείο ενδιαφέροντος. Ύστερα, κατασκευάζεται μια τετράγωνη περιοχή στοιχισμένη στον επιλεγμένο προσανατολισμό και εξάγεται ο περιγραφέας SURF από αυτή. Τέλος, τα χαρακτηριστικά αντιστοιχίζονται μεταξύ δύο εικόνων. Τα βήματα αυτά εξηγούνται παρακάτω.

2.4.1 Ανάθεση προσανατολισμού

Για να είναι αμετάβλητη η περιστροφή της εικόνας, αναγνωρίζεται ένας αναπαραγώγιμος προσανατολισμός για τα σημεία ενδιαφέροντος. Για αυτό τον σκοπό, υπολογίζονται πρώτα οι αποκρίσεις των Haar κυματιδίων στην x και y κατεύθυνση μέσα σε μια κυκλική περιοχή ακτίνας $6s$ γύρω από το σημείο ενδιαφέροντος, όπου s είναι η κλίμακα στην οποία το σημείο ενδιαφέροντος ανιχνεύτηκε. Το βήμα δειγματοληψίας είναι εξαρτώμενο από την κλίμακα και επιλέγεται να είναι s. Το μέγεθος των κυματιδίων είναι επίσης εξαρτώμενο από την κλίμακα και τίθεται ως $4s$. Έτσι, μπορούν να χρησιμοποιηθούν πάλι οι εικόνες-ολοκλήρωμα για γρήγορο φιλτράρισμα. Τα φίλτρα αυτά φαίνονται στην εικόνα 2.5. Μόνο 6 πράξεις χρειάζονται για να υπολογιστεί η απόκριση στην x ή y διεύθυνση σε κάθε κλίμακα.

Όταν οι αποκρίσεις κυματιδίων υπολογιστούν και ζυγιστούν με την Γκαουσιανή ($\sigma = 2s$) επικεντρωμένα στο σημείο ενδιαφέροντος, οι αποκρίσεις εκπροσωπούνται ως σημεία σε ένα χώρο με οριζόντια δύναμη απόκρισης κατά μήκος της τετμημένης και την κατακόρυφη δύναμη απόκρισης κατά μήκος της τεταγμένης. Ο κυρίαρχος προσανατολισμός υπολογίζεται από τον υπολογισμό του αθροίσματος όλων των αποκρίσεων μέσα σε ένα ολισθαίνον



Σχήμα 2.6: Ανάθεση προσανατολισμού: ένα ολισθαίνον παράθυρο προσανατολισμού με μέγεθος $\frac{\pi}{3}$ ανιχνεύει τον κυρίαρχο προσανατολισμό της απόκρισης του Γκαουσιανού ζυγισμένου κυματιδίου σε κάθε σημείο δειγματοληψίας μέσα σε κυκλική περιοχή γύρω από το σημείο ενδιαφέροντος.

παράθυρο προσανατολισμού με μέγεθος $\frac{\pi}{3}$ (δες 2.6). Οι οριζόντιες και κατακόρυφες αποκρίσεις μέσα στο παράθυρο αθροίζονται κι αυτό δίνει ένα διάνυσμα τοπικού προσανατολισμού. Το μεγαλύτερο τέτοιο διάνυσμα πάνω σε όλα τα παράθυρα καθορίζει τον προσανατολισμό του σημείου ενδιαφέροντος. Το μέγεθος του ολισθαίνοντος παραθύρου είναι μια παράμετρος που επιλέγεται προσεκτικά. Τα μικρά μεγέθη έχουν πρόβλημα με μεμονωμένες κυρίαρχες κλίσεις, ενώ τα μεγάλα μεγέθη τείνουν να δίνουν μέγιστα στο μήκος διανύσματος τα οποία δεν είναι αντιπροσωπευτικά. Και τα δύο καταλήγουν σε λανθασμένο προσανατολισμό του σημείου ενδιαφέροντος.

2.4.2 Περιγραφέας βασισμένος σε άθροισμα αποκρίσεων Haar κυματιδίων

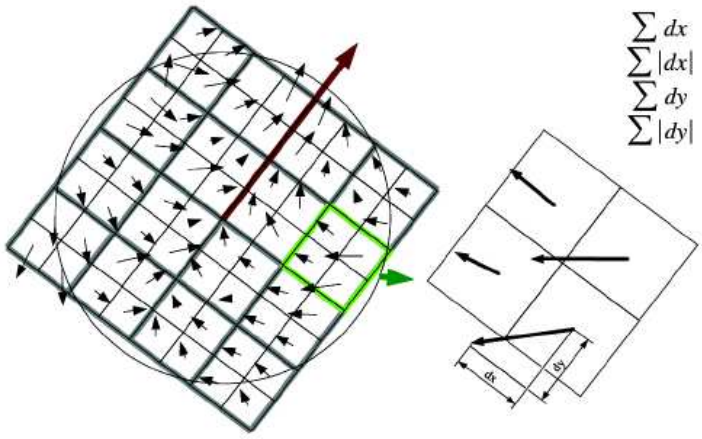
Για την εξαγωγή του περιγραφέα, το πρώτο βήμα αποτελείται από την κατασκευή μιας τετράγωνης περιοχής επικεντρωμένης γύρω από το σημείο ενδιαφέροντος και προσανατολισμένη κατά μήκος του προσανατολισμού που επιλέχθηκε στο προηγούμενο βήμα. Το μέγεθος του παραθύρου είναι $20s$. Παραδείγμα από τέτοιες τετράγωνες περιοχές φαίνονται στην 2.7.

Σχήμα 2.7: Διαφορετικές κλίμακες προσανατολισμένων παραθύρων περιγραφέων σε μια εικόνα.

Η περιοχή χωρίζεται σε μικρότερες $4x4$ τετράγωνες υπο-περιοχές. Αυτό διατηρεί σημαντική χωρική πληροφορία. Για κάθε υπο-περιοχή, υπολογίζονται οι αποκρίσεις των κυματιδίων Haar σε $5x5$ κατανεμημένα σημεία δειγματοληψίας. Για λόγους απλότητας, ονομάζεται d_x η απόκριση του κυματιδίου Haar στην οριζόντια διεύθυνση και d_y η αντίστοιχη απόκριση στην κατακόρυφη διεύθυνση (μέγεθος φίλτρου $2s$). Το ¨οριζόντιο¨ και ¨κατακόρυφο¨ εδώ ορίζεται σε σχέση με τον επιλεγμένο προσανατολισμό του σημείου ενδιαφέροντος (σχήμα 2.8). Για την αύξηση της στιβαρότητας ως προς γεωμετρικές παραμορφώσεις και σφάλματα εντοπισμού, οι αποκρίσεις d_x και d_y ζυγίζονται πρώτα με μια Γκαουσιανή ($\sigma = 3.3s$) επικεντρωμένη στο σημείο ενδιαφέροντος.

Κατόπιν, οι αποκρίσεις κυματιδίων d_x και d_y αθροίζονται πάνω σε κάθε υπο-περιοχή και σχηματίζουν το πρώτο σύνολο καταχωρήσεων στο διάνυσμα του περιγραφέα. Για να υπάρχει πληροφορία για την πόλωση των αλλαγών των εντάσεων, αφαιρούνται επίσης οι απόλυτες τιμές των αποκρίσεων, $|d_x|$ και $|d_y|$. Έτσι, κάθε υπο-περιοχή έχει ένα τετραδιάστατο διάνυσμα περιγραφέα v για την δομή έντασης

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{2.5}$$

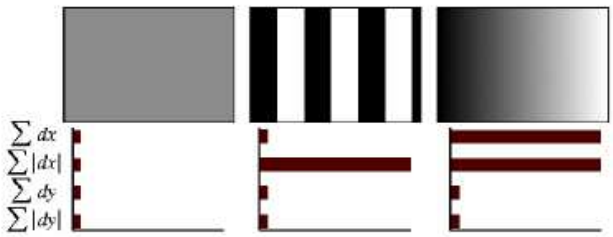


Σχήμα 2.8: Για να δημιουργηθεί ο περιγραφέας, ενα προσανατολισμένο τετραγωνικό πλέγμα με $4x4$ τετράγωνες υπο-περιοχές τίθεται πάνω στο σημείο ενδιαφέροντος (αριστερά). Για κάθε τετράγωνο υπολογίζονται οι αποκρίσεις κυματιδίων. Οι $2x2$ υποδιαιρέσεις κάθε τετραγώνου αντιστοιχούν στα πραγματικά πεδία του περιγραφέα. Αυτά είναι τα αθροίσματα $dx, |dx|, dy$ και $|dy|$, υπολογισμένα σε σχέση με τον προσανατολισμό του πλέγματος (δεξιά).

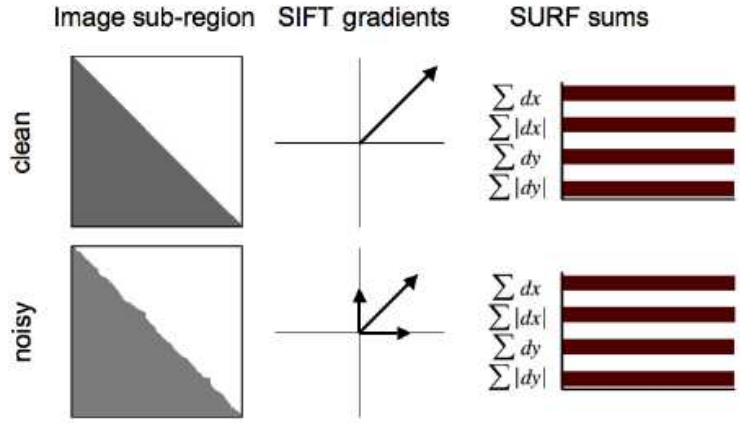
Ενώνοντας αυτό για όλες τις $4x4$ υπο-περιοχές, προκύπτει ένα διάνυσμα περιγραφέα με μήκος 64. Οι αποκρίσεις κυματιδίων είναι αμετάβλητες ως προς προκατάληψη στο φωτισμό. Αμεταβλητότητα ως προς την αντίθεση (contrast) επιτυγχάνεται μετατρέποντας τον περιγραφέα σε μοναδιαίο διάνυσμα.

Το σχήμα 2.9 δείχνει τις ιδιότητες του περιγραφέα για τρία διαφορετικά σχήματα εντάσεων μιας εικόνας μέσα σε μια υπο-περιοχή. Συνδυασμοί τέτοιων τοπικών σχημάτων εντάσεων οδηγούν σε διακριτικό περιγραφέα.

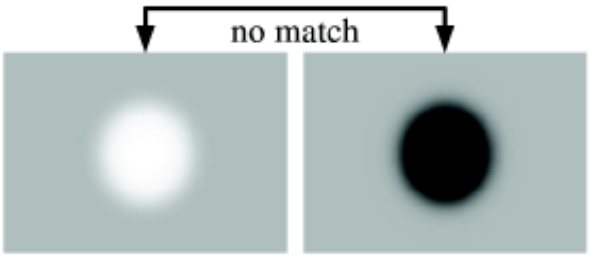
Ο SURF είναι μέχρι κάποιο βαθμό, κοινός στη γενική ιδέα με τον SIFT, αφού και οι δύο εστιάζουν στη χωρική κατανομή της πληροφορίας κλίσης. Παρ'όλα αυτά, ο SURF υπερτερεί του SIFT σε πρακτικά όλες τις περιπτώσεις. Αυτό γίνεται κυρίως γιατί ο SURF ενσωματώνει την πληροφορία κλίσης σε υπό-ομάδα, ενώς ο SIFT εξαρτάται σε προσανατολισμούς των μεμονομένων κλίσεων. Αυτό κάνει τον SURF λιγότερο ευαίσθητο σε θόρυβο, όπως φαίνεται στο σχήμα 2.10.



Σχήμα 2.9: Οι εγγραφές του περιγραφέα μιας υπο-περιοχής αντιπροσωπεύουν τη φύση του σχήματος των εντάσεων. Αριστερά: στην περίπτωση της ομογενούς περιοχής, όλες οι τιμές είναι σχετικά μικρές. Μέση: στην παρουσία συχνοτήτων στην x διεύθυνση, η τιμή του $\sum |d_x|$ είναι υψηλή, αλλά όλες οι άλλες παραμένουν χαμηλές. Αν η ένταση αυξάνεται σταδιακά στην x διεύθυνση, και οι δύο τιμές ,$\sum |d_x|$ και $\sum d_x$ θα είναι υψηλές.



Σχήμα 2.10: Λόγω της καθολικής ενσωμάτωσης του περιγραφέα του SURF, είναι πιο ανθεκτικός σε διάφορες διαταράξεις της εικόνας σε σύγκριση με τον SIFT που δρα πιο τοπικά.



Σχήμα 2.11: Αν η αντίθεση μεταξύ δύο σημείων ενδιαφέροντος είναι διαφορετική (σκοτεινό πάνω σε φωτεινό υπόβαθρο έναντι φωτεινού πάνω σε σκοτεινό υπόβαθρο), το υποψήφιο δε θεωρείται σημαντικό ταίριασμα.

2.4.3 Γρήγορη ευρετηρίαση για αντιστοίχιση

Για γρήγορη ευρετηρίαση κατά τη διάρκεια της φάσης αντιστοίχισης, περιλαμβάνεται το πρόσημο της Λαπλασιανής (δηλαδή το ίχνος ου Εσσιανού μητρώου) για το εξεταζόμενο σημείο ενδιαφέροντος. Τυπικά, τα σημεία ενδιαφέροντος βρίσκονται σε δομές τύπου σταγόνας. Το πρόσημο της Λαπλασιανής διακρίνει φωτεινές σταγόνες πάνω σε σκοτεινά υπόβαθρα. Αυτό το χαρακτηριστικό είναι διαθέσιμο χωρίς επιπλέον κόστος, αφού είχε ήδη υπολογιστεί κατά τη φάση ανίχνευσης. Στο στάδιο της αντιστοίχισης, συγκρίνονται μόνο τα χαρακτηριστικά που έχουν ίδιο τύπο αντίθεσης (δες 2.11). Έτσι, αυτή η ελάαχιστη πληροφορία επιτρέπει γρήγορη αντιστοίχιση, χωρίς να μειώνει την επίδοση του περιγραφέα.

2.5 Επιδόσεις του SURF

Σύμφωνα με πειραματικές μετρήσεις που έγιναν στο [2] και [4] ο αλγόριθμος SURF είναι περίπου 3-5 φορές γρηγορότερος από τον SIFT, ενω παράλληλα είναι πιο ανθεκτικός ως προς τον θόρυβο και προσαρμόζεται πιο εύκολα για παράλληλη επεξεργασία αφού κάθε Εσσιανή εικόνα μπορεί να παραχθεί ανεξάρτητα (σε αντίθεση με τον SIFT). Το μεγάλο κέρδος στην ταχύτητα προέρχεται από τη χρήση των εικόνων-ολοκλήρωμα, που μειώνουν δραστικά τις πράξεις για απλές συνελίξεις-κουτιά ανεξαρτήτου κλίμακας. Η μεγαλύτερη βελτίωση όμως είναι στην ταχύτητα του περιγραφέα. Ο SURF καθιστά δυνατό τον υπολογισμό σε πραγματικό χρόνο χωρίς απώλειες σε επιδόσεις, πράγμα

που αποτελεί ένα μεγάλο πλεονέκτημα για εφαρμογές της Υπολογιστικής Όρασης που απαιτούν άμεση διαδραστικότητα.

Κεφάλαιο 3

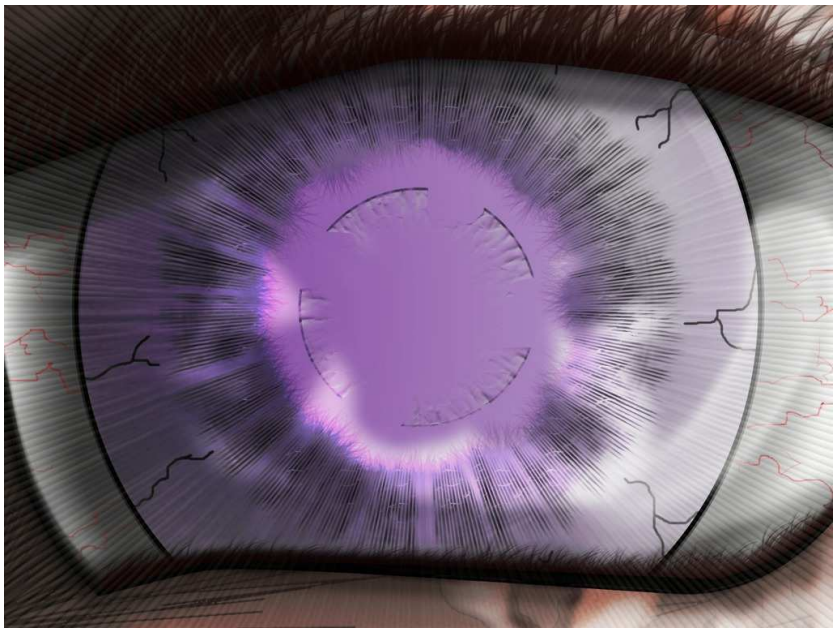
Byakugan - ένα πλήρες σύστημα αναγνώρισης λογοτύπων

> Life is a great big canvas, and you should throw all the paint you can on it.
>
> Danny Kaye

3.1 Εισαγωγή

Στην ενότητα αυτή, θα παρουσιάσουμε το ολοκληρωμένο σύστημα αναγνώρισης λογοτύπων που αναπτύξαμε στα πλαίσια της Διπλωματικής Εργασίας αυτής. Θα γίνει εκτενής ανάλυση της αρχιτεκτονικής του, σχολιασμός και μελέτη των αξιοσημείωτων και πιο σημαντικών τμημάτων κώδικα, περιγραφή των λειτουργιών του και επισήμανση των τεχνικών σημείων εκείνων που χρήζουν σημασίας.

Το σύστημα έχει ονομαστεί Byakugan, όρος που προέρχεται από το δημοφιλές Ιαπωνικό anime Naruto παραπέμποντας στη δυνατότητα μετατροπής του ματιού σε όργανο ισχυρής ενόρασης (σχήμα 3.1) και αποδίδεται στα δικά μας πλαίσια στην ικανότητα της υπολογιστικής *όρασης* να διακρίνει τα λογότυπα.



Σχήμα 3.1: Το όνομα Byakugan προέρχεται από τη δημοφιλές Ιαπωνικό anime Naruto και παραπέμπει στη δυνατότητα μετατροπής του ματιού σε όργανο ισχυρής ενόρασης.

3.2 Αρχιτεκτονική

Το Byakugan αποτελείται από τον βασικό εξυπηρετητή (server), που επιτελεί τη λειτουργία της αναγνώρισης των λογοτύπων καθώς και της παράλληλης εξυπηρέτησης πελατών ιστού, και του πελάτη (client) που είναι η εφαρμογή στην πλατφόρμα Android και επικοινωνεί με τον εξυπηρετητή στέλνοντας του την εικόνα προς ανάλυση.

3.2.1 Byakugan Server

Ο εξυπηρετητής αποτελείται από δύο βασικά μέρη:

1. Τον πυρήνα του λογισμικού (Byakugan Core) που είναι γραμμένος σε C/C++ και εκτελεί αποκλειστικά και επικεντρωμένα τη διαδικασία ανάλυσης και αναγνώρισης των λογοτύπων μέσα σε εικόνες.

2. Τον εξυπηρετητή ιστού (web server) που είναι γραμμένος σε Python και υποστηρίζει την παράλληλη, μέσω threads, εξυπηρέτηση πελατών

λαμβάνοντας την εικόνα από αυτούς ως μια αίτηση HTTP POST και εκτελώντας τον πυρήνα του Byakygan για κάθε μία από αυτές.

3.2.2 Byakugan Client

Ο πελάτης αποτελείται από την εφαρμογή που προορίζεται για έξυπνες κινητές συσκευές βασισμένες στην πλατφόρμα Android και είναι γραμμένη σε Java μαζί με τις κατάλληλες εκείνες βιβλιοθήκες που χρειάζονται για την ανάπτυξη σε Android περιβάλλον. Η εφαρμογή αυτή δίνει τη δυνατότητα στο χρήστη να χρησιμοποιήσει την camera του κινητού του για να τραβήξει μια φωτογραφία, την οποία μετα θα στείλει αυτόματα μέσα μια αίτηση HTTP POST στον Byakygan web server. Σαν απάντηση, θα πάρει το όνομα της εικόνας/λογοτύπου από μία βάση λογοτύπων που ταίριαξε καλύτερα με την σταλθείσα φωτογραφία σύμφωνα με την ανάλυση που θα γίνει από το Byakugan Core.

3.3 Byakugan Core

Ο πυρήνας του συστήματος αποτελείται από το λογισμικό εκείνο που έχει ως σκοπό την αναγνώριση των λογοτύπων. Γράφτηκε σε ένα μεικτό είδος C/C++ χρησιμοποιώντας τη λογική των κλάσεων και τις ευκολίες που παρέχει η C++ (όπως για παράδειγμα η βιβλιοθήκη Boost) χωρίς όμως την αυστηρή αντικειμενοστρέφεια που τη διακατέχει. Το κομμάτι που αφορά το αλγοριθμικό μέρος της Υπολογιστικής Όρασης βασίζεται στο OpenCV. Το OpenCV - Open Source Computer Vision Library αποτελεί μια ανοιχτού κώδικα βιβλιοθήκη από έτοιμες συναρτήσεις, επικεντρωμένες στον τομέα της Επεξεργασίας Εικόνας και της Υπολογιστικής Όρασης.

3.3.1 Βασικές κλάσεις του πυρήνα

Παρακάτω ακολουθεί μια πρώτη επισκόπηση των βασικών κλάσεων του συστήματος μας. Η σειρά που δίνονται ακολουθεί προσέγγιση bottom-up, δηλαδή από κάτω προς τα πάνω, αναφέροντας πρώτες τις λιγότερο αφαιρετικές κλάσεις.

1. *Image*: Η κλάση αυτή ενσωματώνει τη λογική της εικόνας ως μια δομή 2-διάστατου πίνακα, όπως αυτό ορίζεται και στο ίδιο το OpenCV μέσω της δομής Mat (cv::Mat) και περιέχει συναρτήσεις για τη διαχείριση της: εφαρμογή βασικών φίλτρων (canny, soebel, blur, dilate, grayscale κλπ), ανάθεση του path του συστήματος αρχείων στο οποίο βρίσκεται η πρωτότυπη εικόνα στην οποία βασίζεται η κλάση καθώς και της περιοχής ενδιαφέροντος RoI - Region of Interest που αποτελεί το τμήμα εκείνο της εικόνας που περιέχει το λογότυπο. Η κλάση αυτή βασίζεται σε κώδικα του project Unlogo, παρ'όλο που πρακτικά η πλειοψηφία των δανεισμένων μεθόδων δε χρησιμοποιούνται πουθενά στην παρούσα έκδοση του Byakugan αλλά η χρήση τους επιφυλάσσεται για ενδεχόμενη μελλοντική εξέλιξη.

2. *TrainingImage*: Πρόκειται για υπερκλάση της Image και περιέχει συναρτήσεις που αφορούν στην ¨εκπαίδευση¨ του συστήματος ως προς τις εικόνες αυτές. Τα πιο σημαντικά εξαρτήματα της είναι:

   (α') fs::path cnrfile: το μονοπάτι για το αρχείο εκείνο το οποίο περιέχει τις συντεταγμένες για το ορθογώνιο/υπο-περιοχή που περικλείει το λογότυπο στην εικόνα, δηλαδή αυτό που ορίζει την περιοχή ενδιαφέροντος (Region of Interest)

   (β') vector<Point> corners: διάνυσμα (vector - δυναμική δομή πίνακα της βιβλιοθήκης STL της C++) που περιέχει τις συντεταγμένες για την υπο-περιοχή ενδιαφέροντος όπως αυτές εισήχθησαν από το αρχείο *cnrfile*

   (γ') vector<KeyPoint> keypoints: το διάνυσμα που περιέχει τα σημεία ενδιαφέροντος όπως εξάγονται από τον αλγόριθμο ανίχνευσης σημείων - στην προκειμένη περίπτωση τον SURF (βλέπε κεφάλαιο 2:Speeded-Up Robust Features).

   (δ') Mat descriptors: μητρώο που περιέχει τα σημεία όπως υπολογίζονται από τον περιγραφέα (πάλι μέσω του SURF) βάση των σημείων ενδιαφέροντος που εξήχθηκαν προηγουμένως.

3. *TrainingSet*: η κλάση αυτή βρίσκεται πιο ψηλά στην ιεραρχία και περιέχει το διάνυσμα με όλες τις εικόνες, ως TrainingImage-s με τις οποίες

θα συγκριθεί η εικόνα-ερώτηση. Το κάθε TrainingSet αποτελείται από το σύνολο εκείνο των εικόνων που έχουν άμεση σχέση μεταξύ τους. Με άλλα λόγια, στη συνηθέστερη περίπτωση, περιέχει διάφορες εκδοχές φωτογραφιών του λογοτύπου μια συγκεκριμένης εταιρείας.

4. *Matcher*: επιτελεί τη βασική λειτουργία της αντιστοίχισης - δηλαδή της διαδικασίας εκείνης που αφορά στη σύγκριση των εικόνων ώστε να βρεθεί εκείνη που μοιάζει και προσεγγίζει περισσότερο την εικόνα-ερώτηση. Η κεντρική του συνάρτηση που κάνει την πλειοψηφία της εργασίας αυτής είναι η *doQuery*. Ο αλγόριθμος που χρησιμοποιείται για την αντιστοίχιση είναι βασισμένος στο FLANN - Fast Library for Approximate Nearest Neighbours [12], μια βιβλιοθήκη για γρήγορη προσεγγιστική αναζήτηση βάση του κοντινότερου γείτονα. Δημιουργείται συνολικά ένα μόνο singleton αντικείμενο αυτής της κλάσης.

5. *ByakuganOps*: είναι κλάση που περιέχει τις επιλογές που θέτει ο χρήστης κατά την κλήση του Byakugan στη γραμμή εντολών (command line). Οι επιλογές αυτές ρυθμίζουν τη συμπεριφορά του Byakygan και καθορίζουν τον βερμπαλισμό της εξόδου του. Η κλάση αυτή ακολουθεί το σχεδιαστικό πρότυπο *singleton*, δηλαδή περιορίζεται στη δημιουργία ενός και μόνο αντικειμένου της. Αυτό γίνεται, διότι οι επιλογές καθορίζονται από το χρήστη μία και μόνο φορά αμέσως πριν την εκτέλεση του προγράμματος.

6. *Byakugan*: ώντας η βασική κλάση του πυρήνα, περιέχει τις συναρτήσεις εκείνες που θέτουν εν ενεργεία το σύστημα καθολικά. Οι 2 βασικές, που διαχωρίζουν τη λειτουργία του Byakugan σε 2 τρόπους (modes) λειτουργίας είναι:

   (α') *void scan(void)*: η συνάρτηση που ενεργοποιείται σε scan mode μέσω του διακόπτη -s (user option) και σαρώνει αναδρομικά τους φακέλους που ορίζει ο χρήστης, αναζητώντας εικόνες μέσα σε αυτούς. Για κάθε φάκελο δημιουργεί ένα TrainingSet από τις εικόνες που περιέχονται σε αυτόν (η κάθε μία αντιστοιχεί στη δημιουργία μιας TrainingImage) και εκτελεί τον αλγόριθμο SURF ώστε να εξάγει τα σημεία ενδιαφέροντος και τους περιγραφείς από κάθε μία.

Στη συνέχεια αποθηκεύει τα δεδομένα αυτά για κάθε σετ εικόνων σε ένα αρχείο XML για τη γρήγορη ανάκτηση αργότερα.

(β') *int process(void)*: αφορά στο δεύτερο τρόπο λειτουργίας (process mode - ενεργοποίηση από τον διακόπτη -p) του συστήματος που είναι η ανάλυση και σύγκριση της εικόνας-ερώτηση (query image) όπως αυτή δίνεται από το χρήστη μέσω του διακόπτη -q με τις εικόνες που βρίσκονται στη βάση μας. Οι εικόνες αυτές ανακτώνται με γρήγορο τρόπο από τα προηγουμένως υπολογισμένα δεδομένα του scan mode που έχουν αποθηκευτεί στα XML αρχεία. Έτσι ο χρονοβόρος υπολογισμός για την πληθώρα εικόνων στη βάση, μπορεί να έχει γίνει εκ των προτέρων, επιτρέποντας έτσι τη γρήγορη εκτέλεση του προγράμματος σε αυτό το mode, αφού ο χρόνος που θα καταναλωθεί θα αφορά στην αντιστοίχιση και στον υπολογισμό βάση του SURF των σημείων από την εικόνα-ερώτηση μόνο.

Η κλάση αυτή περιέχει, όπως είναι αναμενόμενο, το διάνυσμα (vector) με όλα τα TrainingSet-s όπως αυτό θα δημιουργηθεί από το process mode διαβάζοντας κατάλληλα τα δεδομένα από τα αρχεία XML. Ακολουθεί και αυτή, όπως η ByakuganOps, το σχεδιαστικό πρότυπο *singleton* αφού δε χρειάζεται παρά μόνο μια υλοποίηση (instance) αντικειμένου της συγκεκριμένης κλάσης να υπάρχει συνολικά.

Πέραν των παραπάνω βασικών κλάσεων, υπάρχουν και ορισμένες βοηθητικές συναρτήσεις που βρίσκονται εντός των αρχείων *utils.h, utils.cc*. Μεταξύ άλλων βρίσκονται συναρτήσεις που αποτελούν ασφαλείς εκδοχές, με την έννοια των επιπλέον ελέγχων, πρώιμων συναρτήσεων όπως για παράδειγμα οι *strncpy* και *snprintf* που αντικαθιστόνται από τις ασφαλέστερες *Strncpy* και *Snprintf*. Η πιο σημαντική όμως συνάρτηση είναι η *crossCheckMatching* που εφαρμόζει την αντιστοίχιση βάση των περιγραφέων και του αλγορίθμου αντιστοίχισης που λαμβάνει ως ορίσματα. Επιπλέον, υπάρχουν μέθοδοι για την στρογγυλοποίηση αριθμών (Round()), τη μετατροπή από αριθμό κινητής υποδιαστολής σε string (*FloatToString*) και την έξοδο του προγράμματος σε περίπτωση σφάλματος (*fatal*).

Επιπλέον, η διαχείριση του συνόλου των XML αρχείων, που αποτελεί ουσιαστικά μια υποτυπώδης βάση δεδομένων για τα δεδομένα που εξάγονται από τις εικόνες, γίνεται με τη βιβλιοθήκη ανοιχτού κώδικα *TinyXML*. Τα αρχεία *tinystr.cpp, tinystr.h, tinyxml.cpp, tinyxmlerror.cpp, tinyxml.h, tinyxmlparser.cpp* αποτελούν το σύνολο της βιβλιοθήκης αυτής και ενσωματώνονται στο πρόγραμμα μας για την κατάλληλη εγγραφή και ανάγνωση (parsing) με γρήγορο και εύκολο τρόπο και με μηδαμινό φόρτο, αφού πρόκειται για ιδιαίτερα ¨ελαφριά¨ library.

Τέλος, στο *byakugan.cc* βρίσκεται η *main* συνάρτηση του συστήματος η οποία είναι υπεύθυνη για την λεκτική ανάλυση των επιλογών που καθορίζει ο χρήστης κατά την εκτέλεση του προγράμματος, και κατόπιν της κατάλληλης κλήσης του Byakugan σύμφωνα με αυτές.

3.3.2 Ανάλυση κώδικα και ροής εκτέλεσης

Στο παρόν εδάφιο, θα αναλύσουμε τα τμήματα εκείνα του κώδικα του πυρήνα που χρήζουν ιδιαίτερης σημασίας και προσοχής. Η τακτική ανάλυσης που θα ακολουθήσουμε εδώ θα είναι η αντίθετη με προηγουμένως, δηλαδή θα είναι top-down προσέγγιση.

Η εκτέλεση του Byakugan ξεκινάει, όπως αναφέραμε και πριν, από το αρχείο *byakugan.cc* οπου βρίσκεται η *main* συνάρτηση. Αφού λοιπόν, ο χρήστης καθορίσει τις επιλογές εκτέλεσης στη γραμμή εντολών, το σύστημα θα ξεκινήσει να λειτουργεί σε έναν από τους δύο τρόπους λειτουργίας που αναφέρθηκαν πριν: το *scan mode* ή το *process mode*. Θα παρακολουθήσουμε τη ροή εκτέλεσης και με τους 2 τρόπους, σχολιάζοντας τα κρίσιμα σημεία του κώδικα.

3.3.2.1 Scan Mode

Η λειτουργία της σάρωσης (*scan mode*) αποσκοπεί στην αναδρομική αναζήτηση στους φακέλους, που καθορίζονται από το χρήστη, για όλες εκείνες τις εικόνες που θα αποτελέσουν τη ¨βάση εικόνων¨ με τις οποίες θα συγκρίνονται αργότερα οι εικόνες-ερώτηση (query images) που θα λαμβάνει το σύστημα στη λειτουργία επεξεργασίας (*process mode*). Για κάθε εικόνα, εφαρμόζεται ο αλγόριθμος SURF και τα δεδομένα αποθηκεύονται ως XML

αρχεία για μελλοντική απεύθειας ανάκτηση.

Η διαδικασία ξεκινάει με την κλήση της συνάρτησης *scan* του *singleton* αντικειμένου *Byakugan*, η οποία καλεί αναδρομικά την *scan_internal* τόσες φορές όσες αναφέρει ο αριθμός των φακέλων που ορίστηκαν από το χρήστη μέσω της επιλογής *-s* ή μέσω της επιλογής *-i <file_with_list_of_directories>* (όπου το σύνολο των φακέλων ορίζεται ως λίστα μέσα σε ένα αρχείο). Η *scan_internal* δημιουργεί αντίστοιχα για κάθε φάκελο ένα TrainingSet και καλεί δύο βασικές συναρτήσεις της κλάσης αυτής: την *loadFromDirectory* και την *save*.

Η *loadFromDirectory(string _path)* έχει ως στόχο την φόρτωση όλων των εικόνων από τον εκάστοτε φάκελο, του οποίου μονοπάτι στο σύστημα αρχείων παίρνει ως όρισμα, και την αναπαράσταση τους ως *TrainingImage-s*. Η μετατροπή αυτή έχει δύο βασικές πτυχές: τη δημιουργία/φόρτωση του *cnrfile (corner file)* και την εφαρμογή του *SURF* πάνω στην εικόνα.

1. *Κατασκευή/Φόρτωση cnrfile*: Το *cnrfile*, όπως είπαμε και προηγουμένως, πρόκειται για ένα αρχείο το οποίο περιέχει 4 ζευγάρια συντεταγμένων (σε μορφή x y) που ορίζουν το ορθογώνιο/υπο-περιοχή που περικλείει το λογότυπο μέσα στην εικόνα. Σε πολλές περιπτώσεις η φωτογραφία μπορεί να περιέχει και άλλη άσχετη πληροφορία, πέραν του λογοτύπου. Όμως στη βάση εικόνων χρειάζονται ¨καθαρές¨ εκδόσεις των λογοτύπων αφού λειτουργούν ως πρότυπα αναφοράς για τις μετέπειτα συγκρίσεις που θα γίνουν με τις εικόνες-ερώτηση. Έτσι, για να έχουμε τη μέγιστη δυνατή ακρίβεια, δίνεται η δυνατότητα του προσδιορισμού της περιοχής ενδιαφέροντος (Region of Interest) μέσω των αρχείων αυτών. Στο σχήμα 3.2 φαίνεται ένα παράδειγμα σήμανσης περιοχής ενδιαφέροντος σε ένα λογότυπο βάση των συντεταγμένων που βρίσκονται στον Πίνακα 3.1. Το cnrφιλε, είτε είναι ήδη δημιουργημένο και βρίσκεται στον ίδιο φάκελο (με κατάληξη .cnr) με την ομώνυμη εικόνα, της οποίας την υπο-περιοχή προσδιορίζει, οπότε οι συντεταγμένες φορτώνονται απευθείας στη μνήμη διαβάζοντας το αρχείο, είτε αν δεν υπάρχει, ζητείται να κατασκευαστεί χειροκίνητα από το χρήστη εκείνη τη στιγμή. Στην περίπτωση αυτή, εμφανίζεται ένα παράθυρο με την εικόνα και ο χρήστης καλείται να σημειώσει με το ποντίκι πάνω της τα

x	**y**
116	111
91	425
654	74
656	421

Πίνακας 3.1: παράδειγμα συντεταγμένων cnrfile

4 σημεία που προσδιορίζουν τις συντεταγμένες του ορθογωνίου που περιβάλλει την Region of Interest. Η συνάρτηση, υπεύθυνη για αυτή την εργασία είναι η *generate_cnrfile* της *TrainingImage*. Η βοηθητική συνάρτηση *create_outline*, διαβάζει το προϋπάρχον ή το νεοδημιούργητο αρχείο συντεταγμένων και κατασκευάζει με έξυπνο και ασφαλή τρόπο το ορθογώνιο που περικλείει την υπο-περιοχή του λογοτύπου. Συγκεκριμένα, προνοεί ώστε στην πιο πιθανή περίπτωση που οι συντεταγμένες δεν είναι ευθυγραμμισμένες (άλλωστε κάτι τέτοιο σπάνια μπορεί να επιτευχθεί όταν η σήμανση γίνεται χειροκίνητα), να θεωρεί ως έγκυρο το μεγαλύτερο δυνατό ορθογώνιο που ορίζουν αυτές. Έτσι, στο παράδειγμα 3.2 η περιοχή που θα δημιουργήσει η *create_outline* θα είναι αυτή που φαίνεται στην 3.3. Αυτό γίνεται ώστε να μην υπάρξει πιθανή απώλεια πληροφορίας ως προς το λογότυπο σε περίπτωση ¨αστοχείας¨ του χρήστη.

2. *Εφαρμογή SURF*: Η δεύτερη φάση δημιουργίας της *TrainingImage* αφορά στην εφαρμογή του αλγορίθμου *SURF* ώστε να εξαχθούν τα σημεία ενδιαφέροντος και οι περιγραφείς. Αυτό επιτυγχάνεται χρησιμοποιώντας τις βιβλιοθήκες του *OpenCV* οπου ο αλγόριθμος είναι ήδη υλοποιημένος. Η κλήση των συναρτήσεων *detector->detect()* και *extractor->compute()* είναι αρκετή για να γίνει εφικτό το παραπάνω. Να σημειωθεί πως οι μεταβλητές detector και extractor πρόκειται για generic containers, οπότε μπορούν να χρησιμοποιηθούν δυναμικά και άλλοι αλγόριθμοι εκτός του *SURF*, όπως πχ ο *SIFT*, αν είναι επιθυμητό.

Αφού ολοκληρωθούν αυτές οι δύο φάσεις, τότε η TrainingImage είναι έτοιμη να εισέλθει στο διάνυσμα (vector) του εκάστοτε TrainingSet μαζί με τους εξαχθέντες περιγραφείς που υπολογίστηκαν. Αφού αυτό γίνει για κάθε

Σχήμα 3.2: Με κόκκινο χρώμα φαίνονται τα σημεία που ορίζουν την περιοχή ενδιαφέροντος, σύμφωνα με τις συντεταγμένες του ενρφιλε που φαίνονται στον Πίνακα 3.1

Σχήμα 3.3: Προτιμείται η ασφαλέστερη μέθοδος κατασκευής του περιγράμματος για την περιοχή ενδιαφέροντος ώστε να αποφευχθεί πιθανή απώλεια πληροφορίας ως προς το λογότυπο.

εικόνα στον εκάστοτε φάκελο, τότε το TrainingSet που περιέχει όλες αυτές τις φορτωμένες ¨εκπαιδευμένες¨ εικόνες, είναι έτοιμο να αποθηκευτεί στο δίσκο στη βάση εικόνων για μελλοντική χρήση. Αυτό μας φέρνει στη συνάρτηση *save* του TrainingSet.

Η συνάρτηση *save* έχει ως σκοπό την κατάλληλη αποθήκευση όλων των υπολογισμένων δεδομένων των εικόνων (δηλαδή των εξαχθέντων περιγραφέων από τον αλγόριθμο SURF) από την προηγούμενη φάση. Κάνει χρήση της βιβλιοθήκης *TinyXML* για να μετατρέψει τα δεδομένα αυτά σε ¨ευανάγνωστα¨ τόσο από υπολογιστή όσο κι από άνθρωπο, αρχεία XML. Παρακάτω, βλέπουμε τη δομή των XML αρχείων. Τα ¨...¨ παραπέμπουν σε επανάληψη της ίδιας μορφής XML οδηγίων και συμπεριλήφθησαν μόνο εδώ σκοπίμως για λόγους εξοικονόμησης χώρου.

```
1
2  <?xml version="1.0" ?>
3  <total>6</total>
4  <objects>
5      <filename>image_0
6          <path>/home/ithilgore/byakugan/logos/Adidas/variant-1/mod2_small.jpg</path>
7          <roi x="37" y="67" width="712" height="456" />
8          <keypoints>
9              <keypoint pt.x="4.4376577758789062e+02" pt.y="3.6947502136230469e+01" size="↩
                   1.2000000000000000e+01" angle="2.9698422241210938e+02" response="↩
                   4.2199981689453125e+02" octave="1" class_id="-1" /
10             <keypoint pt.x="4.6738388061523438e+02" pt.y="5.3597236633300781e+01" size="↩
                   1.2000000000000000e+01" angle="2.1169067382812500e+02" response="↩
                   4.1786416625976562e+02" octave="1" class_id="-1" />
11           ...
12         </keypoints>
13         <descriptors>
14             <rows>337</rows>
15             <cols>64</cols>
16             <dt>f</dt>
17             <descriptor>2.00264534e-04</descriptor>
18             <descriptor>4.29603038e-04</descriptor>
19             <descriptor>4.70690982e-04</descriptor>
20             ...
21         </descriptors>
22     </filename>
23     <filename>image_1
24         <path>/home/ithilgore/byakugan/logos/Adidas/variant-1/mod4_small.jpg</path>
25         <roi x="200" y="270" width="475" height="258" />
26         <keypoints>
27         ...
```

```
28          </keypoints>
29          <descriptors>
30          ...
31          </descriptors>
32      </filename>
33      ...
34  </objects>
```

Το *total* αναφέρεται στο σύνολο των εικόνων για αυτό το σετ. Όπως μπορεί να παρατηρήσει κανείς, τα δεδομένα που αποθηκεύονται για κάθε εικόνα είναι τα εξής:

1. *path*: το μονοπάτι στο σύστημα αρχείων για την εικόνα στην οποία βασίζονται τα δεδομένα

2. *roi*: η περιοχή ενδιαφέροντος, με τη μορφή των συντεταγμένων x, y της πάνω αριστερά γωνίας του ορθογωνίου και του πλάτους (width) και ύψους (height) του.

3. *keypoints*: τα σημεία ενδιαφέροντος όπως ανιχνεύονται από τον αλγόριθμο SURF.

4. *descriptors*: οι περιγραφείς όπως εξάγονται από τον SURF.

Να σημειώσουμε εδώ, πως οι μέθοδοι απεύθειας αποθήκευσης των δομών του OpenCV στο δίσκο για την τρέχουσα έκδοση του (2.3.1) ήταν ελλιπείς. Επομένως χρειάστηκε η χειροκίνητη εγγραφή τους, κάτι που φαίνεται στη μαρκοσκελή συνάρτηση *save*.

Η διαδικασία τελειώνει, όταν το σύστημα μας, για λόγους εξοικονόμησης χώρου στο δίσκο, συμπιέσει τα XML αρχεία για κάθε TrainingSet. Η συμπίεση γίνεται με χρήση της υλοποίησης *gzip* των *iostreams* της γνωστής βιβλιοθήκης *boost*. Το *gzip* χρησιμοποιεί, ως γνωστόν, μια παραλλαγή του αλγορίθμου *DEFLATE*, που είναι συνδυασμός των *Lempel-Ziv* και *Huffman coding* και είναι αρκετά αποτελεσματικοί για αρχεία XML. Ενδεικτικά ένα αρχείο XML μεγέθους 3.2 MB κατόπιν συμπίεσης πέφτει στα 434 KB.

Τα παραπάνω, ολοκληρώνουν την περιγραφή λειτουργίας του *scan mode*. Μένει να δούμε τώρα το δεύτερο αλλά σημαντικότερο τρόπο λειτουργίας του *Byakugan*.

3.3.2.2 Process Mode

Η λειτουργία επεξεργασίας (*process mode*) αποτελεί την κινητήρια δύναμη του συστήματος, αφού είναι αυτή που ουσιαστικά συγκρίνει την εικόνα-ερώτηση που θέτει ο χρήστης με τη βάση εικόνων που έχει σχηματιστεί νωρίτερα από τη λειτουργία σάρωσης (*scan mode*).

Η ενεργοποίηση του παρόντος τρόπου λειτουργίας γίνεται με τη χρήση του διακόπτη -p και του ορισμού της εικόνας-ερώτηση με το διακόπτη -q (από το query). Απαιτείται επιπλέον να καθοριστεί το σύνολο των XML αρχείων που θα αποτελέσουν τη βάση εικόνων με τις οποίες θα συγκριθεί η query image. Αυτό γίνεται διότι μπορεί να υπάρχουν περιπτώσεις που είναι προτιμητέο να γίνει η σύγκριση με ένα υποσύνολο της βάσης εικόνων. Ο καθορισμός των ονομάτων των αρχείων XML γίνεται είτε απεύθειας στη γραμμή εντολών με το σχετικό ή απόλυτο μονοπάτι τους είτε με τη χρήση του διακόπτη -i <file_with_list_of_XML_files> με το όνομα του αρχείου που λειτουργεί ως λίστα με τα μονοπάτια των XML αρχείων.

Η αρχική και βασική συνάρτηση που καλείται εδώ είναι η *process* του αντικειμένου *Byakugan*. Σκοπός είναι να φορτώσει από το δίσκο τα αρχεία XML που ορίστηκαν προηγουμένως, να ανακατασκευάσει το σύνολο των *TrainingSet-s* και των *TrainingImage-s* καθενός από αυτά και στη συνέχεια να επιτελέσει τη λειτουργία της αντιστοίχισης και να βρει την εικόνα από τη βάση εικόνων που μοιάζει περισσότερο στην εικόνα-ερώτηση.

Η διαδικασία φόρτωσης από το δίσκο των XML αρχείων γίνεται με τη βοήθεια της *loadFromArchive* της κλάσης *TrainingSet*. Το πρώτο πράγμα που ελέγχει είναι αν τα αρχεία XML είναι συμπιεσμένα, που υπό κανονικές συνθήκες θα είναι. Κατόπιν, τα αποσυμπιέζει αν χρειστεί, και τα φορτώνει στη μνήμη όπου πάλι με τη βοήθεια της βιβλιοθήκης *TinyXML* ακολουθεί η αποκωδικοποίηση τους. Φυσικά, τα σημεία ενδιαφέροντος και περιγραφείς που εξάχθηκαν από τον SURF κατά τη διάρκεια της λειτουργίας της σάρωσης που έχει προηγηθεί, είναι έτοιμα και δε χρειάζεται να ξανα-υπολογιστούν. Το γεγονός αυτό επιταχύνει σε μεγάλο βαθμό τη συνολική διαδικασία.

Μετά το πέρας της *loadFromArchive* για κάθε *TrainingSet*, το *Byakugan* διαθέτει στη μνήμη του ανακατασκευασμένες όλες τις *TrainingImage-s* μαζί με τα υπολογισμένα δεδομένα τους. Η ροή εκτέλεσης στην *process* συνεχίζει

και σειρά έχει η πολυπόθητη λειτουργία της αντιστοίχισης. Η εικόνα-ερώτηση περνάει ως όρισμα στη συνάρτηση *match* του εκάστοτε *TrainingSet* και αυτό με τη σειρά του καλεί την *match* από κάθε ενσωματωμένη *TrainingImage*. Η κάθε *TrainingImage* ενεργοποιεί το καθολικό αντικείμενο *Matcher*, φορτώνει σε αυτό τα δεδομένα της, δηλαδή τα σημεία ενδιαφέροντος και τους περιγραφείς και μετά καλεί τη βασική συνάρτηση αντιστοίχισης *doQuery*.

Η *doQuery* κατάρχάς πρέπει να υπολογίσει, μέσω του *SURF* τα σημεία ενδιαφέροντος και τους περιγραφείς από την εικόνα-ερώτηση. Φυσικά, για την εικόνα σύγκρισης που βρίσκεται στην TrainingImage, κάτι τέτοιο δε χρειάζεται αφού τα στοιχεία ήταν ήδη υπολογισμένα από πριν και απλά φορτώθηκαν έτοιμα στη μνήμη από το αντίστοιχο XML αρχείο. Επίσης, ο υπολογισμός των δεδομένων για την εικόνα-ερώτηση χρειάζεται να γίνει μία και μόνο φορά αφού αυτή δεν αλλάζει καθ'όλη τη διάρκεια εκτέλεσης του προγράμματος.

Στη συνέχεια, υπάρχουν δύο επιλογές ως προς τον τρόπο της σύγκρισης των περιγραφέων: η μία αφορά στην αντιστοίχιση εμπρός-πίσω και η άλλη στην κλασσική απλή αντιστοίχιση. Το Byakugan χρησιμοποιεί την πρώτη από προεπιλογή για λόγους προστιθέμενης ακρίβειας. Το έργο αυτό επιτελεί η *crossCheckMatching* του *utils.cc*. Σε κάθε περίπτωση, ο αλγόριθμος αντιστοίχισης είναι βασισμένος στην υλοποίηση του *OpenCV* για το *FLANN - Fast Library for Approximate Nearest Neighbours* [12]. Η βιβλιοθήκη αυτή επιλέγει τον καλύτερο δυνατό αλγόριθμο εσωτερικά για μια γρήγορη προσεγγιστική αναζήτηση βάση του κοντινότερου γείτονα.

Κατόπιν, υπολογίζεται η ομογραφία (homography) βάση του *RANSAC - RANdom SAmple Consensus*, μιας επαναληπτητική μεθόδου ώστε να εξαλειφθούν τα σημεία outliers, δηλαδή τα σημεία εκείνα ακραίων τιμών που στατιστικά μάλλον δεν ανήκουν στο σετ με τα υπόλοιπα. Το βήμα αυτό εξαφανίζει ένα πλήθος λανθασμένων αντιστοιχίων που βρέθηκαν στην προηγούμενη βάση του matching. Τελικά το πλήθος των ¨πραγματικών¨ σημείων αντιστοίχισης, επιστρέφονται ως αποτέλεσμα από την *doQuery*.

Το αποτέλεσμα αυτό το επιστρέφει η TrainingImage στο αντίστοιχο TrainingSet στο οποίο ανήκει. Έτσι, η συνάρτηση match του TrainingSet αφού συλλέξει τα αποτελέσματα αυτά για κάθε ενσωματωμένη TrainingImage του, ξεκινάει τη διαδικασία ταξινόμησης τους. Η διαδικασία εδώ είναι απλή: βρίσκεται η εικόνα που έδωσε τα καλύτερα αποτελέσματα, δηλαδή αυτή με

το μέγιστο πλήθος σημείων αντιστοίχισης. Το κάθε TrainingSet κατάυτόν τον τρόπο θα επιστρέψει στο αντικείμενο Byakugan την ¨καλύτερη¨ εικόνα του, δηλαδή αυτή που προσέγγιζε περισσότερο την εικόνα-ερώτηση.

Το τελευταίο στάδιο του process mode είναι η συλλογή των ¨καλύτερων¨ εικόνων αυτών από το singleton Byakugan και η εφαρμογή μιας επιπλέον ταξινόμησης, με την ίδια πάλι λογική όπως πριν. Αναζητείται η ¨καλύτερη¨ των ¨καλύτερων¨ εικόνων, δηλαδή αυτή που έχει καθολικά τα περισσότερα σημεία ενδιαφέροντος. Η εικόνα αυτή δίνεται ως η τελική απάντηση από το Byakugan προς το χρήστη με τη χαρακτηριστική γραμμή εξόδου:
Best match: <path_to_image>

Εδώ τελειώνει η περιγραφή και του mode επεξεργασίας του συστήματος μας και μαζί με αυτή και η περιγραφή του πυρήνα του Byakugan. Προχωρούμε τώρα, στην ανάλυση των άλλων δύο σημαντικών συστατικών μερών του.

3.4 Byakugan Web Server

Ο εξυπηρετητής ιστού (web server) του συστήματος μας, έχει ως σκοπό την παράλληλη εξυπηρέτηση δικτυακών αιτήσεων για τη μεταφορά εικόνων-ερώτηση ((χυερψ ιμαγες) από τις κινητές συσκευές Android των χρηστών στη διάθεση του Byakugan Core ώστε να γίνει η ανάλυση τους. Η γλώσσα προγραμματισμού Python μας έδωσε έτοιμα modules για την εύκολη και σύντομη συγγραφή του συγκεκριμένου λογισμικού.

Ο server μας, ακούει από προεπιλογή στο port 8000 για HTTP POST αιτήσεις. Όταν λάβει κάποια αίτηση, ξεκινάει ξεχωριστό νήμα εκτέλεσης (thread) για την εξυπηρέτηση της. Αναζητάει το πεδίο *upfile* μέσα στην κεφαλίδα (header) του HTTP πακέτου και αντιγράφει το περιοχόμενο που ακολουθεί, σε ένα προσωρινό αρχείο με μοναδικό όνομα στον φάκελο /tmp του συστήματος. Είναι προσυμφωνημένο στο πρωτόκολλο επικοινωνίας μεταξύ του server μας και των πελατών (clients) που στέλνουν σε αυτόν, πως το περιεχόμενο της εικόνας θα είναι κωδικοποιημένο με το γνωστό σχήμα *Base64*. Αυτό είναι αναγκαίο για την κατάλληλη και σωστή μεταφορά δυαδικού αρχείου (binary file) μέσω του HTTP, αφού το αρχείο κατάυτόν τον τρόπο

αναπαρίσταται μέσω ενός μεγάλου ASCII string. Επομένως, πριν γραφτεί στο δίσκο το αρχείο, γίνεται πρώτα η αποκωδικοποίηση του ως προς το Base64.

Στη συνέχεια, ο εξυπηρετητής εκτελεί στο ίδιο νήμα ένα στιγμιότυπο του Byakugan Core σε process mode δίνοντας ως όρισμα για την εικόνα-ερώτηση το μονοπάτι του αρχείου που μόλις αποκωδικοποίησε και αποθήκευσε προσωρινά στο δίσκο. Παράλληλα, ανοίγει μέσω pipe, του γνωστού τρόπου ενδοεπικοινωνίας των διεργασίων (IPC - Inter-process Communication) σε Unix-based συστήματα, το ρεύμα εξόδου και λαθών (output and error streams) του Byakugan Core. Έτσι, θα μπορέσει να διαβάσει την απάντηση που δίνει ο πυρήνας του Byakugan ως έξοδο ύστερα από την ανάλυση που θα κάνει στην εικόνα-ερώτηση. Χρησιμοποιώντας απλές τεχνικές λεκτικής ανάλυσης (parsing) κανονικών εκφράσεων (regular expressions), εξάγει από τη γραμμή Best match: <path_to_image> που αναφέραμε προηγουμένως, το μονοπάτι της ¨καλύτερης¨ εικόνας που αντιστοιχεί με την εικόνα-ερώτηση.

Το όνομα της καλύτερης εικόνας αυτής, στέλνει ως απάντηση πίσω στον πελάτη που έκανε την αίτηση. Έτσι, ολοκληρώνεται η λειτουργία του κάθε νήματος που εκτελείται παράλληλα στον εξυπηρετητή.

3.5 Byakugan Client

Ο πελάτης αποτελείται από την Android εφαρμογή, η οποία μπορεί να τρέξει σε οποιαδήποτε ¨έξυπνη¨ κινητή συσκευή διάθετει λογισμικό Android έκδοσης 2.2 και άνω. Σκοπός της είναι να δώσει μια εύκολη διεπαφή στο χρήστη ώστε να μπορεί να βγάζει φωτογραφίες μέσω της κάμερας του κινητού, να αυτοματοποιήσει τη διαδικασία αποστολής τους προς τον Byakugan Server και κατόπιν να λάβει την απάντηση, που θα είναι το όνομα της εικόνας που ταιράζει καλύτερα στη φωτογραφία, από αυτόν και να την παρουσιάσει στο χρήστη.

Η εφαρμογή μας είναι γραμμένη σε Java χρησιμοποιώντας παράλληλα τις βιβλιοθήκες που είναι απαραίτητες για τη λειτουργία της στην πλατφόρμα του Android. Το κυρίως πρόγραμμα βρίσκεται σχεδόν όλο στο αρχείο *MainActivity.java* που περιέχει και τη βασική κλάση (MainActivity). Η έννοια του Activity στο Android αναφέρεται πρακτικά σε ένα παράθυρο διεπαφής με το οποίο μπορεί να αλληλεπιδράσει άμεσα ο χρήστης. Η συγκεκριμένη,

ώντας και η κεντρική, περιέχει αρκετές βοηθητικές συναρτήσεις με τις οποίες επιτυγχάνει τις ζητούμενες αυτοματοποιήσεις που προαναφέρθηκαν. Συγκεκριμένα :

1. *onCreate*: η βασικότερη συνάρτηση, που καλείται με το που δημιουργηθεί το εκάστοτε Activity στο οποίο αναφέρεται. Το γραφικό περιβάλλον του παραθύρου προκύπτει από την κλήση της *setContentView(R.layout.main)* η οποία ανασύρει το περιεχόμενο του αρχείου *res/layout/main.xml* και φτιάχνει το παράθυρο σύμφωνα με τις XML οδηγίες.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3  android:orientation="vertical"
4  android:layout_width="fill_parent"
5  android:layout_height="fill_parent" android:background="@drawable/byakugan"
6  >
7
8      <Button
9          android:id="@+id/camera_button"
10         android:layout_width="fill_parent"
11         android:layout_height="wrap_content"
12         android:text="@string/camera_button_text" />
13
14 <ImageView android:layout_weight="2"
15 android:contentDescription="@string/description"
16 android:id="@+id/imageview"
17 android:layout_width="fill_parent"
18 android:layout_height="0dp" />
19
20 <ProgressBar
21     android:id="@+id/pbDefault"
22     android:layout_width="fill_parent"
23     android:layout_height="wrap_content"
24     style="@android:style/Widget.ProgressBar.Horizontal"
25     android:visibility="invisible"
26  />
27 </LinearLayout>
```

Σύμφωνα με τα παραπάνω, δημιουργείται ένα παράθυρο, με background εικόνα που προέρχεται από το αρχείο drawable/byakugan, ένα κουμπί (button) πατώντας το οποίο ο χρήστης εκκινεί την κάμερα του κινητού και μία μπάρα προόδου (progress bar) η οποία θα δείχνει προσεγγιστικά το χρόνο που απαιτείται μέχρι να λάβουμε απάντηση από τον εξυπηρετητή. Στην εικόνα 3.4 φαίνονται όλα τα παραπάνω πλην

της μπάρας, αφού αυτή εμφανίζεται αφού ο χρήστης τραβήξει τη φωτογραφία και η εφαρμογή αρχίσει την επικοινωνία με τον web server.

Οι επόμενες γραμμές κώδικα της παρούσας συνάρτησης δημιουργούν έναν OnClickListener, δηλαδή ορίζουν την call-back function (onClick) που θα κληθεί στην περίπτωση event triggering που στην προκειμένη περίπτωση αφορά στο πάτημα του κουμπιού "Take Picture".

2. *onCreateOptionsMenu*: συνάρτηση που δημιουργεί το μενού με τις διαθέσιμες επιλογές, όταν ο χρήστης πατήσει το (hardware) κουμπί *menu* που υπάρχει σε κάθε συσκευή Android. Το μενού το ανασύρει από το res/menu/menu.xml και στη δική μας περίπτωση αφορά στις προτιμήσεις (Preferences) που μπορεί να θέσει ο χρήστης. Η μόνη διαθέσιμη επιλογή στην παρούσα έκδοση έχει να κάνει με τη ρύθμιση της διεύθυνσης IP και του port του Byakugan web server με τον οποίο θα επικοινωνήσει η εφαρμογή μας.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android" >
3     <item android:id="@+id/prefs" android:title="@string/prefsTitle"></item>
4
5
6 </menu>
```

3. *onOptionsItemSelected*: η συνάρτηση αυτή έχει άμεση σχέση με την προηγούμενη που περιγράψαμε, αφού προσπελάυνει τις διαθέσιμες επιλογές του μενού και καλεί αντιστοίχως τον κατάλληλο handler. Στην περίπτωση των Preferences απλά καλεί την PrefsActivity, μια διεπαφή που παρουσιάζει τις προτιμήσεις που μπορεί να θέσει ο χρήστης. Οι επιλογές αυτές ανασύρονται από το αρχείο res/xml/prefs.xml όπως φαίνεται από το PrefsActivity.java. Αυτό είναι μια ακόμα ευκολία που μας παρέχει η πλατφόρμα του Android ως προς την αυτοματοποίηση τετριμμένων διαδικασιών όπως αυτή.

Σχήμα 3.4: Η βασική διεπαφή με την οποία αλληλεπιδρά ο χρήστης στην Android εφαρμογή.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
3     <EditTextPreference android:title="@string/server_ip_addr" android:key="↩
          server_ip" android:summary="@string/ip_summary"/>
4
5
6 </PreferenceScreen>
```

4. *createHttpClient*: βοηθητική συνάρτηση που αρχικοποιεί καταλλήλως παραμέτρους και μεταβλητές για την κλάση HttpClient που χρησιμοποιούμε από το πακέτο του Apache - org.apache.http για το τμήμα της εφαρμογής που πρέπει να στείλει την HTTP POST αίτηση.

5. *convertResponseToString*: βοηθητική συνάρτηση που μετατρέπει την απάντηση που θα λάβει η εφαρμογή από τον εξυπερητητή ιστού σε μορφή κατάλληλη για εμφάνιση σε παράθυρο της διεπαφής.

6. *onActivityResult*: εδώ επιστρέφει η ροή εκτέλεσης, κατόπιν της επιτυχούς λήψης της φωτογραφίας από το χρήστη. Το αρχείο που περιέχει τη φωτογραφία φορτώνεται, η εικόνα αλλάζει κλίμακα και συμπιέζεται βάση του προτύπου JPEG ώστε το μέγεθος να είναι αρκετά μικρό και τέλος κωδικοποιείται σε ASCII string μέσω του γνωστού σχήματος Base64. Στη συνέχεια, το ρεύμα (stream) αυτό από bytes αναλαμβάνει να το διαχειριστεί η ενσωματωμένη κλάση *PostToServer* που περιγράφουμε ακριβώς από κάτω.

Η ενσωματωμένη κλάση *PostToServer*, όπως μαρτυράει και το ίδιο το όνομα της, έχει ως σκοπό την αποστολή της φωτογραφίας, που στο στάδιο αυτό έχει μετατραπεί σε ένα σύνολο από διαδοχικά bytes κατάλληλα για διαδικτυακή μεταφορά, στον Byakugan web server. Μια αξιοσημείωτη ιδιότητά της είναι ότι λειτουργεί ασύγχρονα, δηλαδή ανεξάρτητα και παράλληλα με την υπόλοιπη εφαρμογή. Αυτό γίνεται εφικτό μέσω της λειτουργίας *AsyncTask* που παρέχει το Android και πρόκειται για μια απλουστευμένη μορφή threading. Η διαδικασία εδώ ξεκινάει με την δημιουργία ενός HttpClient με χρήστη της createHttpClient που περιγράψαμε προηγουμένως. Στη συνέχεια, φορτώνονται η IP address και το port του εξυπηρετητή από τις προτιμήσεις (Preferences) του χρήστη ή αν αυτές δεν έχουν αλλάξει, χρησιμοποιούνται οι ¨εργοστασιακές¨ ρυθμίσεις. Κατόπιν, δημιουργείται το πακέτο της

HTTP POST αίτησης, στο οποίο ενσωματώνεται η Base64-κωδικοποιημένη φωτογραφία στο πεδίο upfile. Η γραμμή *"HttpResponse response = httpclient.execute(httppost);"* κάνει διπλή εργασία αφού στέλνει την αίτηση αυτή και περιμένει την απάντηση του εξυπηρετητή την οποία όταν λάβει θα αποθηκεύσει στη μεταβλητή *"response"*. Εντωμεταξύ, καθ'όλη τη διάρκεια εκτέλεσης των παραπάνω, ο χρήστης είναι ελεύθερος να βγάλει κι άλλες φωτογραφίες, αφού οι εργασίες αυτές γίνονται όπως είπαμε ασύγχρονα, δηλαδή στο παρασκήνιο παράλληλα. Το σημείο αυτό της αναμονής της απάντησης είναι και το πιο χρονοβόρο, αφού πέραν του χρόνου μεταφοράς του αρχείου δικτυακά και στη συνέχεια της μεταφοράς της απάντησης πίσω στον πελάτη, χρειάζεται και κάποιο χρόνο και το ίδιο το Byakugan Core που θα αναλύσει την εικόνα και θα την συγκρίνει με τη βάση εικόνων. Παρ'όλα αυτά, όπως θα δούμε και στο επόμενο κεφάλαιο ο χρόνος μεταφοράς της εικόνας αποτελεί συνήθως το μεγαλύτερο bottleneck. Ενώ γίνονται τα παραπάνω, εμφανίζεται η μπάρα προόδου που δείχνει τον αναμενόμενο χρόνο μέχρι τη λήψη της τελικής απάντησης. Η λειτουργία της υλοποιείται στην επίσης ενσωματωμένη και ασύγχρονη κλάση *ProgressBarAsyncTask*. Εν τέλει, η convertResponseToString θα μετατρέψει, όπως είπαμε και πριν, την HTTP απάντηση του εξυπηρετητή σε κατάλληλη μορφή που θα εμφανιστεί σε πλαίσιο διαλόγου στην κεντρική διεπαφή.

Κεφάλαιο 4

Εκτέλεση και Αποτελέσματα

Action is the foundational key to all success.

Pablo Picasso

4.1 Εισαγωγή

Στο παρόν κεφάλαιο, θα παρουσιάσουμε την έμπρακτη λειτουργία του συστήματος μας και θα αναλύσουμε τα αποτελέσματα της εκτέλεσης του. Θα δούμε πως αλληλεπιδρούμε με το Byakugan τόσο σε τοπικό επίπεδο, μέσω του πυρήνα του, όσο και συνολικά, δηλαδή απομακρυσμένα μέσω της κινητής συσκευής και της εφαρμογής Android που αναπτύξαμε. Τέλος, θα σχολιάσουμε τις ενδεχόμενες προεκτάσεις του και τις δυνατότητες που έχει για χρήση σε ποικιλία παιγνίων.

4.2 Τοπική εκτέλεση του Byakugan Core

Αρχικά, θα δούμε τον πυρήνα του Byakugan εν λειτουργία τοπικά, πράγμα που σημαίνει πως θα εκτελεστεί ανεξάρτητα από τον εξυπερητητή ιστού και την εφαρμογή του πελάτη. Η τοπική εκτέλεση θα μας δείξει τι πραγματικά γίνεται στο παρασκήνιο της ανάλυσης και σύγκρισης των εικόνων, τι επιλογές έχει ο διαχειριστής του συστήματος ως προς τη δημιουργία της βάσης εικόνων

```
[ithilgore@fitz byakugan_server]$ ./byakugan
byakugan
Usage: two modes of operation:
       [+] SCAN mode
        -  scan given directories for images and compute the needed
        -  keypoints which will be saved in XML format
       ./byakugan -s <directory_1> <directory_2>...
       ./byakugan -s -i <file_with_list_of_directories>
       [+] PROCESS mode
        -  compare <query_image> against all images which have been
        -  previously been decomposed and saved into XML-encoded format
       ./byakugan -g -p <XML_file_1> <XML_file_2>... -q <query_image>
       ./byakugan -g -p -i <file_with_list_of_XML_files> -q <query_image>
OPTIONS:
       -s: scan mode
       -p: process mode
       -q <query_image>: image to compare against
       -i <input_filename: get list of directories/files from this file
       -g: Show correspondence graph with matching keypoints
       -v: Increase verbosity level (use twice or more for greater effect)
       -d[level]: Set or increase debugging level (Up to 10 is meaningful)
       -h: Print this help summary page.
       -V: Print version number
```

Σχήμα 4.1: Οι διαθέσιμες επιλογές που έχει ο διαχειριστής του συστήματος Byakugan κατά την τοπική εκτέλεση

και τον έλεγχο της ορθής λειτουργίας του συστήματος καθώς και τη γραφική απεικόνιση της αντιστοίχισης.

4.2.1 Επιλογές λειτουργίας

Όπως έχει αναλυθεί εκτενώς στο προηγούμενο κεφάλαιο, ο πυρήνας του συστήματος έχει δύο βασικούς τρόπους λειτουργίας: το scan mode και process mode. Στην εικόνα 4.1 φαίνονται το σύνολο των διαθέσιμων επιλογών, όπως αυτά εκτυπώνονται στην οθόνη από το ίδιο το πρόγραμμα.

Ας δούμε αναλυτικά τι επιτελεί το κάθενα:

1. *-s*: ενεργοποίηση του scan mode. Η δημιουργία της βάσης εικόνων γίνεται με αυτό τον τρόπο.

2. *-p*: ενεργοποίηση του process mode. Η ανάλυση και σύγκριση της εικόνας-ερώτηση με τη βάση εικόνων γίνεται με αυτό τον τρόπο.

3. *-q <query_image>*: η εικόνα-ερώτηση που θα αναλυθεί κατά το process mode.

4. *-i <input_filename*: ο διακόπτης αυτός μας επιτρέπει να ορίσουμε ένα αρχείο το οποίο θα λειτουργήσει ως λίστα και θα περιέχει σε κάθε γραμμή το μονοπάτι των φακέλων ή αρχείων που θα χρησιμοποιηθούν είτε

με το scan mode είτε με το process mode. Στην περίπτωση του scan mode αναφέρεται στο σύνολο των φακέλων τα οποία θα αναζητήσει το σύστημα για να βρει εικόνες και να δημιουργήσει τη βάση εικόνων, ενώ στην περίπτωση process mode αναφέρεται στα XML αρχεία που περιέχουν τα προ-υπολογισμένα δεδομένα της βάσης εικόνων.

5. *-g*: ενεργοποίηση της γραφικής απεικόνισης του process mode. Σε κάθε σύγκριση που θα γίνεται, θα αναπαρίστανται σε κοινό παράθυρο το σύνολο των κοινών σημείων της εικόνας-ερώτηση με την εκάστοτε εικόνα από τη βάση εικόνων.

6. -v: διακόπτης που καθορίζει το πόσο αναλυτική θα είναι η έξοδος του συστήματος. Αύξηση του verbosity σημαίνει πως θα εκτυπώνεται παραπάνω πληροφορία ως προς τις εσωτερικές λειτουργίες του συστήματος.

7. *-d[level]*: επιλογή που καθορίζει το επίπεδο της πληροφορίας αποσφαλμάτωσης, δηλαδή πληροφορίες που αποσκοπούν σε καθαρά προγραμματιστικά θέματα και έχουν σκοπό να βοηθήσουν στον εντοπισμό και επίλυση ενός ενδεχόμενου προβλήματος λειτουργίας.

8. *-h*: εκτύπωση του μενού όπως φαίνεται στην εικόνα 4.1

9. *-V*: εκτύπωση της έκδοσης του συστήματος. Η τρέχουσα έκδοση είναι η *byakugan version 1.0alpha*.

4.2.2 Scan mode και δημιουργία βάσης εικόνων

Θα δημιουργήσουμε αρχικά μια μικρή βάση εικόνων για να δούμε αναλυτικά τη διαδικασία του scan mode. Στον φάκελο *byakugan_server/logos* βρίσκονται εικόνες λογοτύπων από ποικίλες μάρκες. Θα ασχοληθούμε αρχικά με τα logos της Adidas και της Coca-Cola. Στον φάκελο *logos/Adidas* έχουμε τοποθετήσει δύο παραλλαγές του λογοτύπου της μάρκας. Στο *logos/Adidas/variant-1/* έχουμε τις εξής εικόνες: 4.2, 4.3, 4.4, 4.5, 4.6, 4.7

Στο *logos/Adidas/variant-2/* έχουμε την εικόνα 4.8

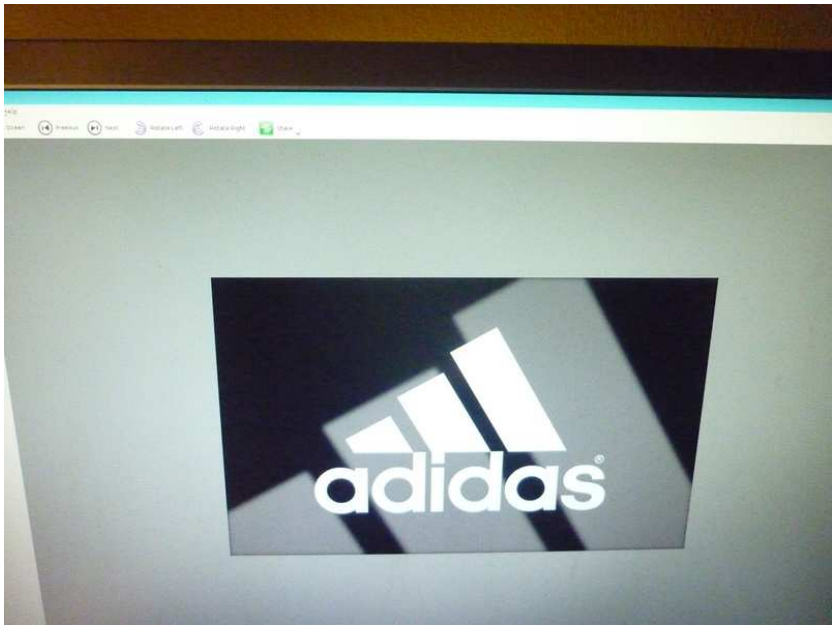
Αντίστοιχα, το λογότυπο της Coca-Cola με το οποίο θα ασχοληθούμε φαίνεται στην εικόνα 4.9

Σχήμα 4.2: Το βασικό λογότυπο Adidas

Σχήμα 4.3: Φωτογραφία του βασικού λογοτύπου Adidas υπό κανονική γωνία λήψης

Σχήμα 4.4: Φωτογραφία του βασικού λογοτύπου Adidas από πιο κοντινή λήψη

Σχήμα 4.5: Φωτογραφία του βασικού λογοτύπου Adidas υπό πλάγια γωνία λήψης



Σχήμα 4.6: Φωτογραφία του βασικού λογοτύπου Adidas από πάνω

Σχήμα 4.7: Φωτογραφία του βασικού λογοτύπου Adidas από κάτω

Σχήμα 4.8: Μια άλλη έκδοση του επίσημου λογοτύπου της Adidas

Σχήμα 4.9: Το επίσημο λογότυπο της Coca-Cola

Τα αντίστοιχα cnrfiles που έχουν προκατασκευαστεί βρίσκονται στον πίνακα 4.1. Να σημειώσουμε εδώ, πως σε μερικές περιπτώσεις, σημειώσαμε σκοπίμως τα πλαίσια των λογοτύπων να είναι τέτοια ώστε να περικλείουν μόνο ένα μέρος τους. Αυτό έγινε για να δούμε πως συμπεριφέρεται το σύστημα μας σε αυτή την περίπτωση και πόσα κοινά σημεία είναι ικανό να βρει τότε. Αυτό θα φανεί και πιο κάτω όταν θα δούμε δίπλα-δίπλα την εικόνα-ερώτηση και τις εικόνες με τις οποίες συγκρίνεται αυτή μέσω του διακόπτη -g που περιγράψαμε πιο πάνω. Τότε θα γίνει αισθητό το ότι μέσω των cnrfiles γίνεται να χρησιμοποιηθεί ένα υποσύνολο των σημείων του λογοτύπου της κάθε εικόνας και επαφίεται στο διαχειριστή που τα δημιουργεί να επιλέξει ποιό θα είναι αυτό.

Θα δούμε τώρα την εκτέλεση του συστήματος σε τοπικό επίπεδο σε scan mode. Ενεργοποιούμε την επιλογή για να βλέπουμε debugging output σε επίπεδο 4 (-d4), που είναι αρκετά κατατοπιστικό για να δούμε τι γίνεται στο παρασκήνιο της εκτέλεσης. Σαν ορίσματα δίνουμε τα σχετικά μονοπάτια των φακέλων καθώς και τον διακόπτη -s που ενεργποιεί το scan mode.

```
1  [ithilgore@fitz byakugan_server]$ ./byakugan -s logos/Adidas logos/Coca-Cola/ -d4
2  [+] Scan mode.
3  Opening logos/Adidas
4  encountered a file: logos/Adidas/variant-2
5  encountered a file: logos/Adidas/variant-1
6  Opening logos/Adidas/variant-2
7  encountered a file: logos/Adidas/variant-2/.directory
8  encountered a file: logos/Adidas/variant-2/adidas_logo_black.cnr
9  encountered a file: logos/Adidas/variant-2/adidas_logo_black.jpg
10 found an image logos/Adidas/variant-2/adidas_logo_black.jpg
11 full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-2/↩
       adidas_logo_black.jpg
12 ----corners----
13 x: 105  y: 56
14 x: 109  y: 240
15 x: 412  y: 45
16 x: 412  y: 234
17 Identifying keypoints
18 Extracting descriptors
19 saving data to logos/Adidas/variant-2.xml
20 Opening logos/Adidas/variant-1
21 encountered a file: logos/Adidas/variant-1/.directory
22 encountered a file: logos/Adidas/variant-1/mod2_small.jpg
23 found an image logos/Adidas/variant-1/mod2_small.jpg
```

	x	y
Adidas - 4.2	116	111
	91	425
	654	74
	656	421

	x	y
Adidas - 4.3	313	239
	328	365
	505	209
	493	364

	x	y
Adidas - 4.4	37	67
	39	520
	748	84
	749	523

	x	y
Adidas - 4.5	523	204
	540	337
	594	205
	592	341

	x	y
Adidas - 4.6	200	270
	218	528
	675	271
	612	510

	x	y
Adidas - 4.7	388	193
	384	357
	480	188
	495	347

	x	y
Adidas - 4.8	105	56
	109	240
	412	45
	412	234

	x	y
CocaCola - 4.9	42	28
	25	291
	838	14
	814	290

Πίνακας 4.1: Συντεταγμένες βάση των cnrfiles

```
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/mod2_small.jpg
----corners----
x: 37  y: 67
x: 39  y: 520
x: 748  y: 84
x: 749  y: 523
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/mod4_small.jpg
found an image logos/Adidas/variant-1/mod4_small.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/mod4_small.jpg
----corners----
x: 200  y: 270
x: 218  y: 528
x: 675  y: 271
x: 612  y: 510
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/logo2.jpg
found an image logos/Adidas/variant-1/logo2.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/logo2.jpg
----corners----
x: 116  y: 111
x: 91  y: 425
x: 635  y: 87
x: 675  y: 436
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/logo2.cnr
encountered a file: logos/Adidas/variant-1/mod4_small.cnr
encountered a file: logos/Adidas/variant-1/mod5_small.jpg
found an image logos/Adidas/variant-1/mod5_small.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/mod5_small.jpg
----corners----
x: 388  y: 193
x: 384  y: 357
x: 480  y: 188
x: 495  y: 347
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/mod1_small.jpg
found an image logos/Adidas/variant-1/mod1_small.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/mod1_small.jpg
----corners----
x: 313  y: 239
x: 328  y: 365
```

```
x: 505  y: 229
x: 493  y: 364
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/mod2_small.cnr
encountered a file: logos/Adidas/variant-1/mod3_small.jpg
found an image logos/Adidas/variant-1/mod3_small.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
    mod3_small.jpg
----corners----
x: 523  y: 204
x: 540  y: 337
x: 594  y: 205
x: 592  y: 341
Identifying keypoints
Extracting descriptors
encountered a file: logos/Adidas/variant-1/mod1_small.cnr
encountered a file: logos/Adidas/variant-1/mod3_small.cnr
encountered a file: logos/Adidas/variant-1/mod5_small.cnr
saving data to logos/Adidas/variant-1.xml
Opening logos/Coca-Cola/
encountered a file: logos/Coca-Cola/coca-cola.jpg
found an image logos/Coca-Cola/coca-cola.jpg
full path: /home/ithilgore/byakugan/code/byakugan_server/logos/Coca-Cola/coca-cola.jpg
----corners----
x: 42  y: 28
x: 25  y: 291
x: 838  y: 14
x: 814  y: 290
Identifying keypoints
Extracting descriptors
encountered a file: logos/Coca-Cola/coca-cola.cnr
saving data to logos/Coca-Cola/.xml
Scanning done.
```

Παρατηρούμε πως γίνεται αναδρομική αναζήτηση για εικόνες στους φακέλους που δώσαμε. Για κάθε μία από αυτές διαβάζεται το αντίστοιχο cnrfile, ενώ βλέπουμε και την έξοδο του καθενός κάτω από την κεφαλίδα "corners". Μετά τον υπολογισμό των περιγραφέων, τα αποτελέσματα αποθηκεύονται στα αντίστοιχα XML αρχεία, τα οποία κατόπιν συμπιέζονται οπότε το τελικά ονόματα των δημιουργηθέντων αρχείων είναι: *logos/Adidas/variant-1.xml.gz*, *logos/Adidas/variant-2.xml.gz*, *logos/Coca-Cola/.xml.gz* . Σε περίπτωση που δεν προϋπήρχε το αντίστοιχο cnrfile, τότε θα καλούμασταν εμείς που τρέξαμε για πρώτη φορά το πρόγραμμα να σημειώσουμε τα 4 σημεία του bounding box μέσω μιας έτοιμης διεπαφής παραθύρου που θα εμφανιζόταν σε κάθε μία εικόνα. Ο χρόνος που πήρε η εκτέλεση του παραπάνω σε έναν

Intel(R) Pentium(R) M processor 2.00GHz με 1 GB RAM είναι μόλις:

real 0m4.722s
user 0m2.626s
sys 0m0.123s

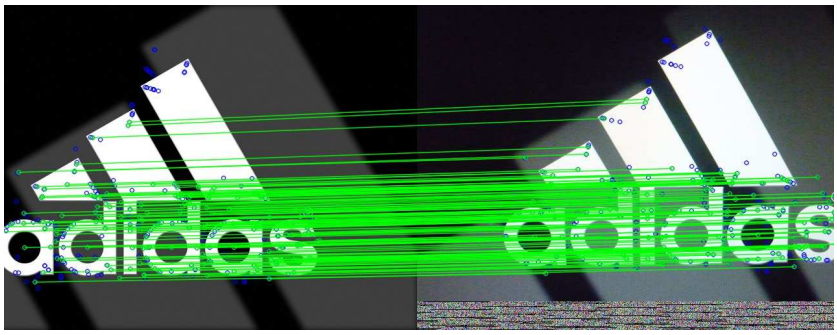
4.2.3 Εκτέλεση process mode

Έχοντας δημιουργήσει μια αρχική βάση εικόνων, θα προχωρήσουμε στην τοπική εκτέλεση του συστήματος σε process mode βλέποντας τη γραφική απεικόνιση της αντιστοίχισης για κάθε συνδυασμό εικόνων. Για ευκολία δημιουργούμε ένα αρχείο-λίστα που περιέχει τα σχετικά μονοπάτια των XML αρχείων που θα χρησιμοποιήσουμε σα βάση εικόνων. Το αρχείο *input.txt* δηλαδή περιέχει τα κάτωθι:
logos/Adidas/variant-1.xml.gz
logos/Adidas/variant-2.xml.gz
logos/Coca-Cola/.xml.gz

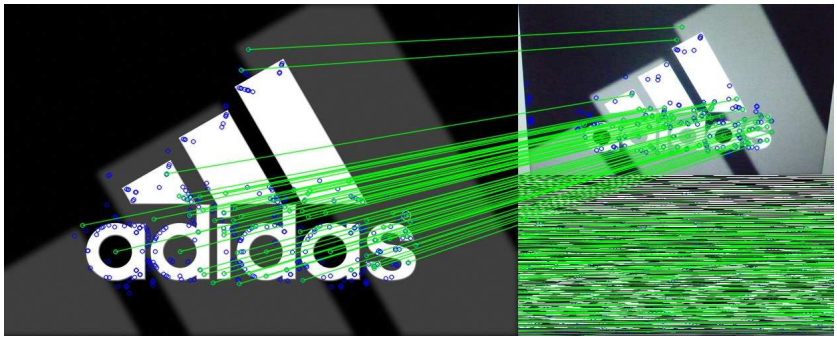
Σαν όρισμα για την εικόνα-ερώτηση θα δώσουμε αρχικά μια εικόνα πανομοιότυπη με την εικόνα 4.2. Εκτελούμε, έτσι, τον πυρήνα με την εξής εντολή:

byakugan -g -p -i input.txt -q adidas.jpg
Τα αποτελέσματα έχουν ως εξής:

```
1 [ithilgore@fitz byakugan_server]$ ./byakugan –p –g –i input.txt –q adidas.jpg
2 [+] Process mode.
3 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/mod2_small.jpg ↩
      real_matches:143
4 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/mod4_small.jpg ↩
      real_matches:60
5 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/logo2.jpg ↩
      real_matches:235
6 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/mod5_small.jpg ↩
      real_matches:11
7 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/mod1_small.jpg ↩
      real_matches:27
8 /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant–1/mod3_small.jpg ↩
```



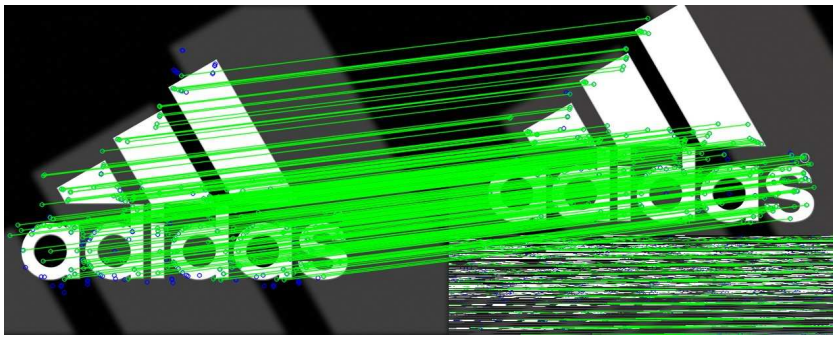
Σχήμα 4.10: Αντιστοίχιση με τη φωτογραφία 4.4, κοινά σημεία: 143



Σχήμα 4.11: Αντιστοίχιση με τη φωτογραφία 4.6, κοινά σημεία: 60

```
    real_matches:12
/home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-2/adidas_logo_black.jpg real_matches:71
/home/ithilgore/byakugan/code/byakugan_server/logos/Coca-Cola/coca-cola.jpg real_matches:5
Best match: /home/ithilgore/byakugan/code/byakugan_server/logos/Adidas/variant-1/logo2.jpg
Processing done.
```

Παρατηρούμε πως η καλύτερη αντιστοίχιση έγινε, όπως ήταν αναμενόμενο, με την πρωτότυπη εικόνα 4.2. Το αξιοσημείωτο είναι το γεγονός ότι στη σύγκριση με την εικόνα 4.8 βρέθηκε ικανοποιητικός αριθμός κοινών σημείων παρ'όλο που πρόκειται για διαφορετική ουσιαστικά εικόνα και όχι για παραλλαγή της αρχικής με αλλοιώσεις που οφείλονται στη γωνία λήψης και το θόρυβο της φωτογραφίας όπως έγινε στις υπόλοιπες εικόνες. Αυτό είναι γενικά θετικό γεγονός καθώς βλέπουμε ότι το σύστημα είναι ανθεκτικό και σε βασικές παραλλαγές της εικόνας προς σύγκριση. (Σημείωση: Τα ¨σκουπίδια¨ που φαίνονται κάτω από τις εικόνες στα δεξιά δεν λαμβάνονται υπ'όψιν

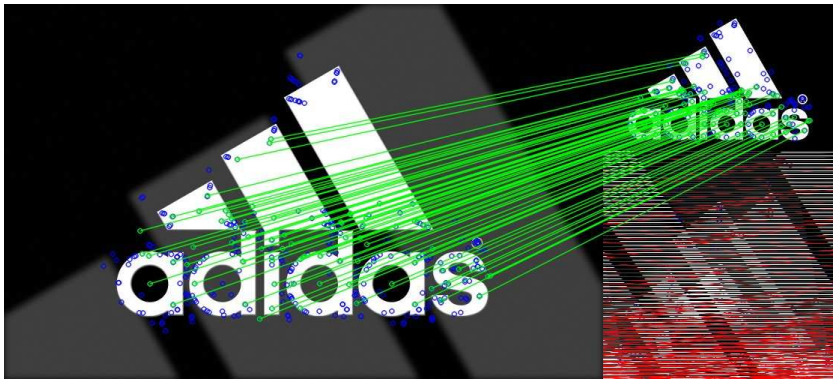


Σχήμα 4.12: Αντιστοίχιση με το επίσημο λογότυπο 4.2, κοινά σημεία: 235 (το καλύτερο)

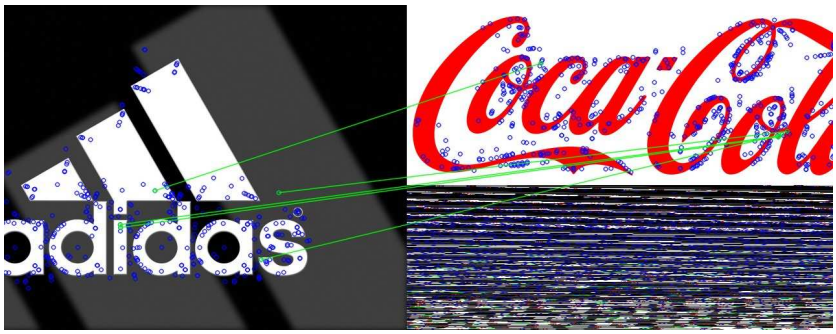
Σχήμα 4.13: Αντιστοίχιση με τη φωτογραφία 4.7, κοινά σημεία: 11 - προσοχή στο γεγονός ότι λόγω του περιορισμένου boungind box όπως καθορίστηκε από το cnrfile της εικόνας, βρίσκονται σαφώς λιγότερα σημεία

Σχήμα 4.14: Αντιστοίχιση με τη φωτογραφία 4.3, κοινά σημεία: 27 - ισχύει ότι και στην προηγούμενη σύγκριση για το περιορισμένο boungind box

Σχήμα 4.15: Αντιστοίχιση με τη φωτογραφία 4.5, κοινά σημεία: 12 - ακόμα πιο μικρο boungind box



Σχήμα 4.16: Αντιστοίχιση με το λογότυπο 4.8, κοινά σημεία: 71 - ικανοποιητικά αρκετά κοινά σημεία παρ'όλο που πρόκειται για διαφορετική ουσιαστικά εικόνα και όχι για παραλλαγή της αρχικής με αλλοιώσεις που οφείλονται στη γωνία λήψης και το θόρυβο της φωτογραφίας όπως έγινε στις προηγούμενες εικόνες - αυτό είναι γενικά θετικό γεγονός καθώς βλέπουμε ότι το σύστημα είναι ανθεκτικό και σε βασικές παραλλαγές της εικόνας



Σχήμα 4.17: Αντιστοίχιση με το λογότυπο 4.9, κοινά σημεία: 5 - προφανώς τα λιγότερα κοινά σημεία αφού πρόκειται για τελείως διαφορετική εικόνα

και δεν έχουν καμία σχέση με την αντιστοίχιση.)

Ο χρόνος εκτέλεσης παραμένει ιδιαίτερα χαμηλός, και αυτό σε ένα μηχάνημα (Intel(R) Pentium(R) M processor 2.00GHz με 1 GB RAM) που είναι σαφώς παλαιότερης εποχής:
real 0m2.532s
user 0m2.117s
sys 0m0.077s

4.3 Ολοκληρωμένη εκτέλεση του συστήματος Byakugan

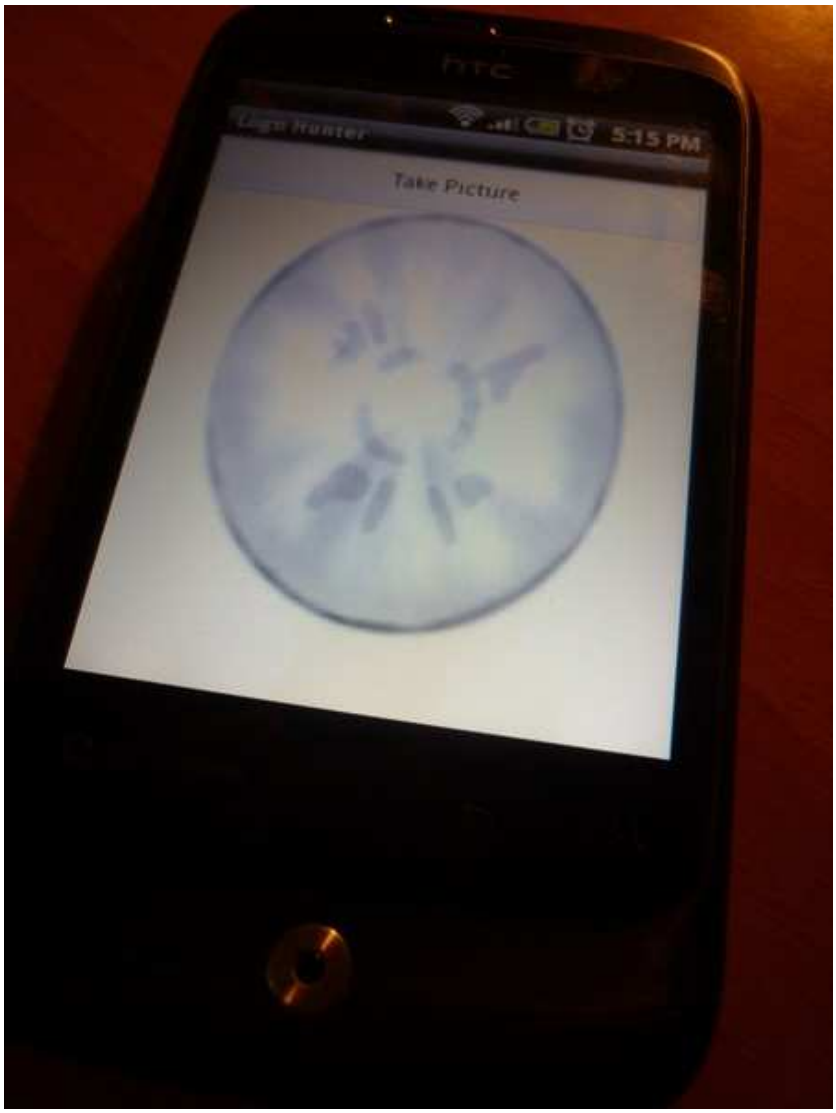
Στο παρόν εδάφιο, θα παρουσιάσουμε μια ολοκληρωμένη επισκόπηση της συνολικής λειτουργίας του συστήματός μας. Θα δούμε τη διαδικασία από την αρχή όπου ο χρήστης θέτει την Android εφαρμογή στο κινητό του σε λειτουργία, μέχρι την τελική λήψη της απάντησης από τον εξυπηρετητή ιστού με τον οποίο επικοινωνεί. Για την παρούσα ανάλυση χρησιμοποιήθηκαν τα εξής:
HTC Wildfire mobile device - Android 2.2.1
Laptop: Intel(R) Pentium(R) M processor 2.00GHz, 1 GB RAM

Στις εικόνες 4.18 και 4.19 βλέπουμε αντίστοιχα την εφαρμογή *Logo Hunter* εγκατεστημένη στην προαναφερθείσα κινητή συσκευή και το αρχικό παράθυρο διεπαφής αφού την εκτελέσουμε.

Πριν προχωρήσουμε παραπέρα όμως, πρέπει να εκτελέσουμε τον εξυπηρετητή ιστού του Byakugan στον υπολογιστή μας και να ρυθμίσουμε την εφαρμογή να επικοινωνεί με την IP address που αυτός κατέχει στο τοπικό δίκτυο.

```
1
2 # ifconfig |  grep -w inet | cut -d" " -f10
3 192.168.1.4
```

Σχήμα 4.18: Η εφαρμογή μας Logo Hunter εγκατεστημένη στη κινητή συσκευή HTC Wildfire που χρησιμοποιήσαμε



Σχήμα 4.19: Το αρχικό παράθυρο διεπαφής της εφαρμογής Logo Hunter

```
4
5
6 [ithilgore@fitz byakugan_server]$ python2 server.py
7 started httpserver...
```

Επομένως, ρυθμίζουμε την εφαρμογή μας να επικοινωνεί με την παραπάνω διεύθυνση. Το port που ακούει ο python web server όπως έχουμε αναφέρει είναι το 8000 από προεπιλογή. Η ρύθμιση γίνεται πατώντας το (hardware) κουμπί "menu" της συσκευής και επιλέγοντας το Settings (εικόνες 4.20, 4.21, 4.22).

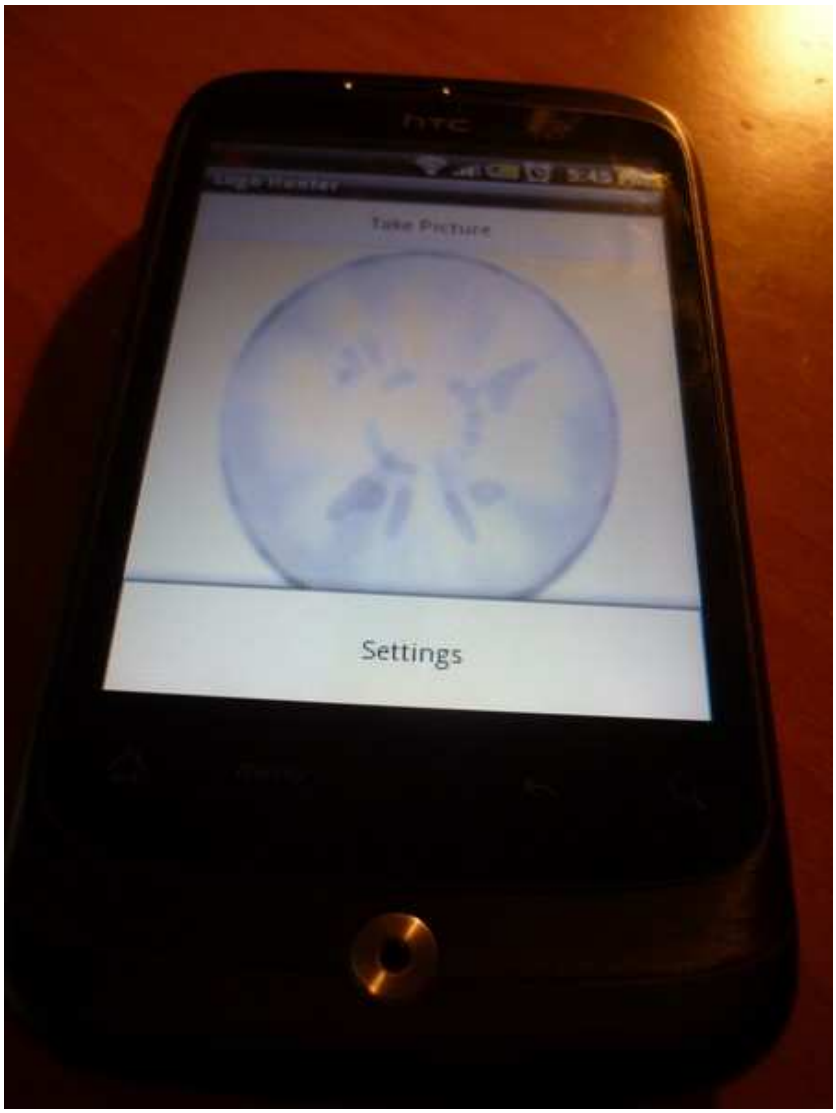
Πλέον είμαστε έτοιμοι να πατήσουμε στο κουμπί "Take Picture" η ενεργοποίηση του οποίου θα μας μεταφέρει στη διεπαφή της κάμερας του κινητού. Από εκεί θα τραβήξουμε φωτογραφία το λογότυπο της Adidas από πλάγια γωνία λήψης για να προστεθεί σκοπίμως έτσι σχετική αλλοίωση στην εικόνα. Στην εικόνα 4.23 βλέπουμε τη φωτογραφία που τραβήξαμε με το κινητό. Όταν πατήσουμε "Done", τότε η εφαρμογή ξεκινάει να στέλνει αυτόματα τη φωτογραφία στον web server, ενώ μπορούμε να δούμε την πρόοδο στο κάτω μέρος της διεπαφής (εικόνα 4.24).

Ο εξυπηρετητής ιστού μας ειδοοποιεί εκτυπώνοντας σχετικό μήνυμα στην έξοδο του όταν λάβει τη φωτογραφία καθώς και με ποιό όνομα την αποθηκεύει προσωρινά στο σύστημα αρχείων. Αμέσως μετά εκτελεί το byakugan core στα παρασκήνια και όταν έχει έτοιμη την απάντηση, την εκτυπώνει στην έξοδο και τη στέλνει δικτυακά πίσω στην εφαρμογή Android.

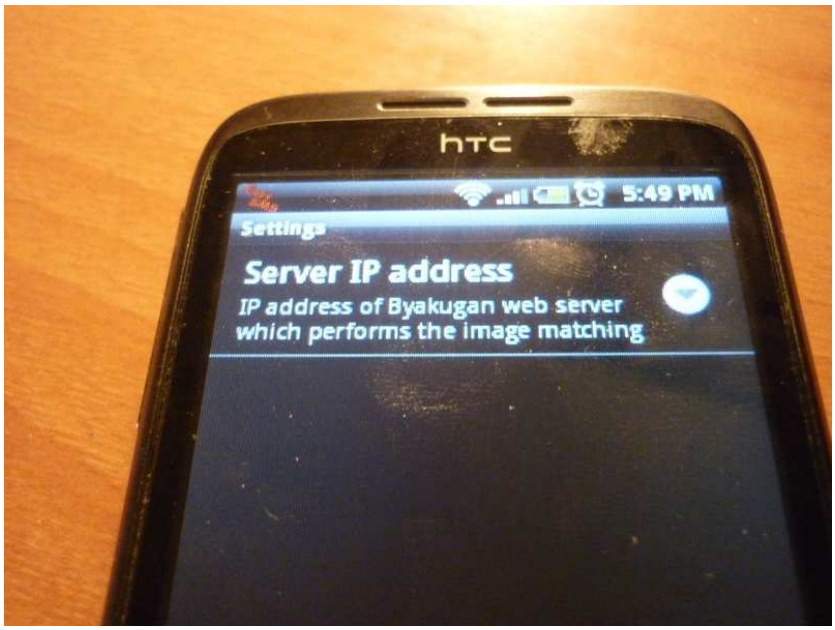
```
1
2 File received. Saved as /tmp/tmpBpq9hv
3 /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant_1/↩
      mod2_small.jpg
4 192.168.1.2 - - [31/May/2012 19:08:48] "POST / HTTP/1.1" 200 -
```

Η εφαρμογή λαμβάνει το αποτέλεσμα και το εμφανίζει σε παράθυρο διαλόγου (εικόνα 4.25). Σκοπίμως, για την ώρα έχουμε αφήσει να εμφανίζεται το πλήρες μονοπάτι της εικόνας που βρέθηκε ως η καλύτερη αντιστοίχιση για να μπορούμε να ελέγξουμε και να διαπιστώσουμε εύκολα την ακρίβεια του συστήματος μας. Αυτό φυσικά μπορεί να γίνει πιο ¨φιλικό προς το χρήστη¨ ανά πάσα στιγμή με λίγες γραμμές κώδικα.

Ας δοκιμάσουμε τώρα να τρέξουμε τοπικά το byakugan core για να δούμε αναλυτικά τα αποτελέσματα που προηγουμένως ¨κρύφτηκαν¨ λόγω της



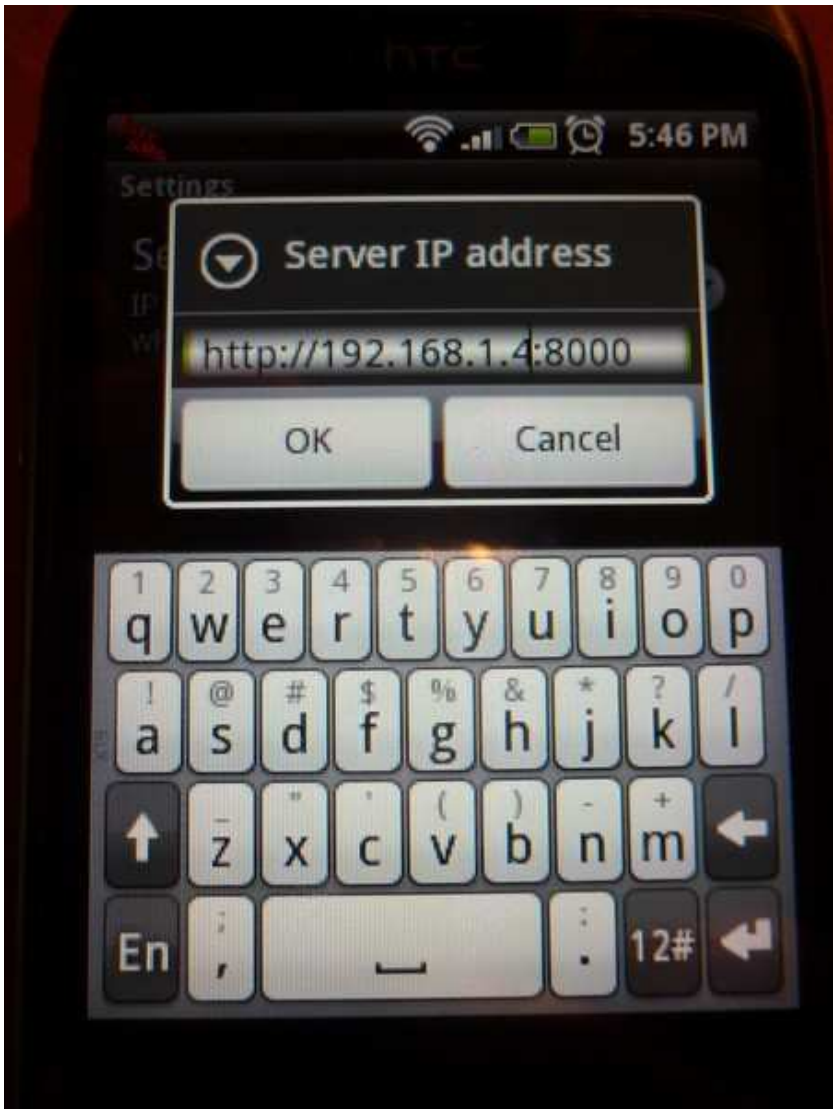
Σχήμα 4.20: Το μενού που εμφανίζεται πατώντας το ομώνυμο κουμπί της συσκευής



Σχήμα 4.21: Το μενού προτιμήσεων/ρυθμίσεων όπου καθορίζουμε τη δικτυακή διεύθυνση του εξυπηρετητή με τον οποίον θα επικοινωνήσουμε

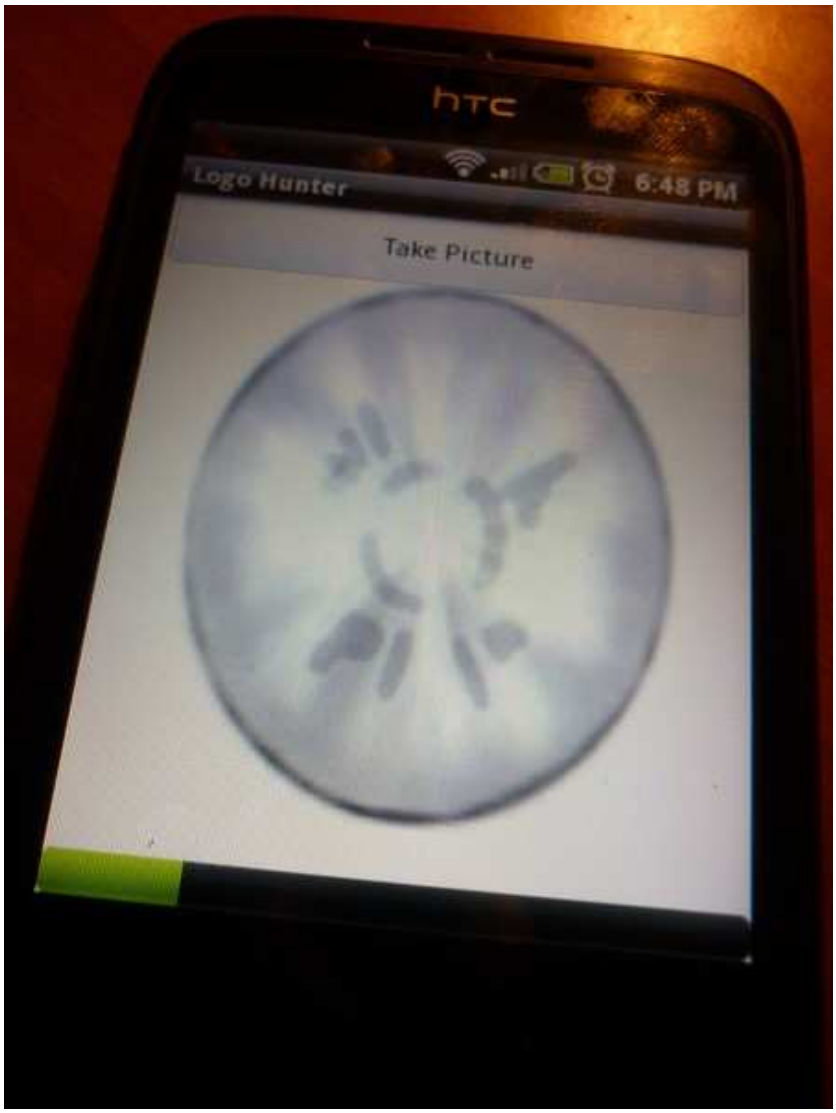
παρασκηνιακής εκτέλεσης του από τον εξυπηρετητή ιστού.

```
1  [ithilgore@fitz byakugan_server]$ ./byakugan -p -i input.txt -q /tmp/tmpBpq9hv
2  [+] Process mode.
3  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
       mod2_small.jpg real_matches:33
4  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
       mod4_small.jpg real_matches:15
5  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/logo2↩
       .jpg real_matches:33
6  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
       mod5_small.jpg real_matches:7
7  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
       mod1_small.jpg real_matches:9
8  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-1/↩
       mod3_small.jpg real_matches:7
9  /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/variant-2/↩
       adidas_logo_black.jpg real_matches:33
10 /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Coca-Cola/coca-cola.↩
       jpg real_matches:6
11 Best match: /home/ithilgore/thesis_papers/byakugan/code/byakugan_server/logos/Adidas/↩
       variant-1/mod2_small.jpg
12 Processing done.
```

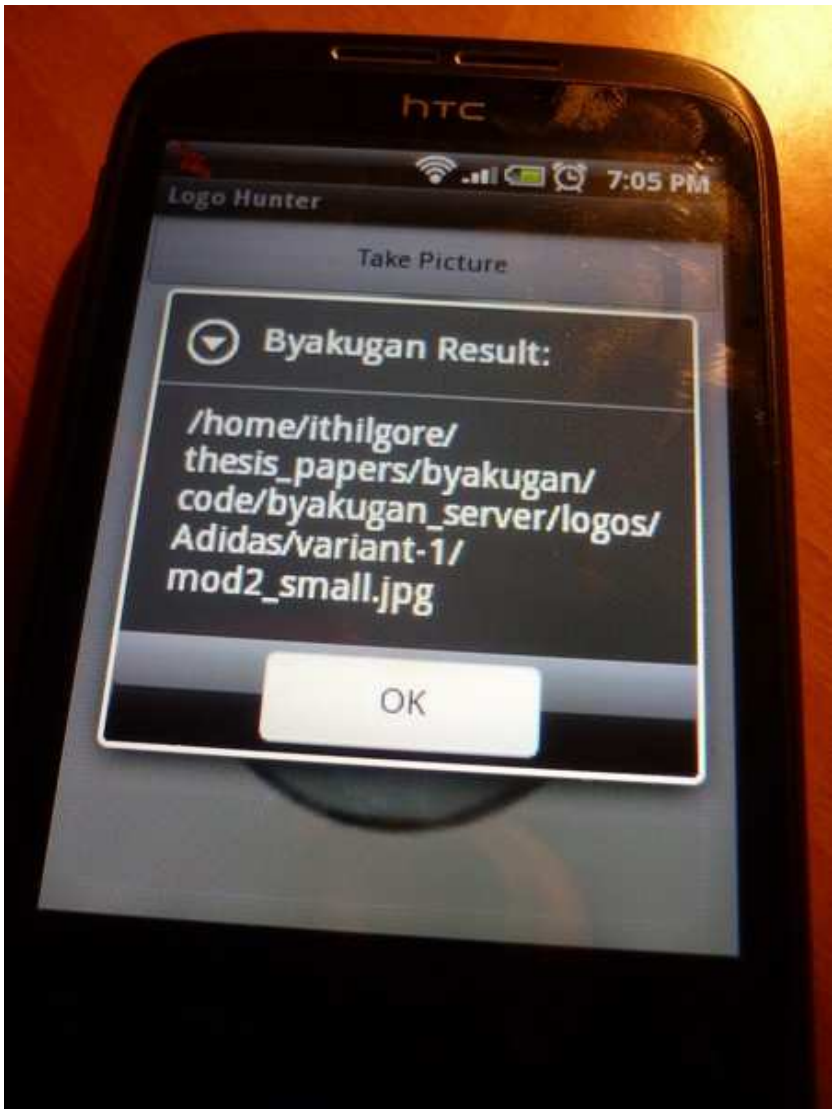


Σχήμα 4.22: Καθορισμός της IP address και port σύμφωνα με τα προηγούμενα

Σχήμα 4.23: Η φωτογραφία που βγάλαμε με την κάμερα του κινητού θα περιέχει αρκετή αλλοίωση, τόσο ως προς τη γωνία λήψης όσο και από το φωτισμό και άλλες παραμορφώσεις θορύβου.



Σχήμα 4.24: Η μπάρα προόδου μας δείχνει πόσος χρόνος απομένει προσεγγιστικά μέχρι τη λήψη της τελικής απάντησης.



Σχήμα 4.25: Το αποτέλεσμα εμφανίζεται σε ξεχωριστό παράθυρο διαλόγου τη στιγμή που παραλαμβάνεται από την εφαρμογή

Παρατηρούμε πως υπήρχε ίσοπαλία΄ μεταξύ των εικόνων 4.4 (mod2_small.jpg), 4.2 (logo2.jpg) και 4.8 (adidas_logo_black.jpg) με 33 κοινά σημεία στην κάθε μία. Αυτό δεν πρέπει να μας κάνει εντύπωση, αφού η εικόνα 4.4 είναι αρκετά ποιοτική φωτογραφία του αρχικού λογότυπου, η 4.2 είναι το αρχικό λογότυπο και η 4.8 είναι επίσης επίσημο λογότυπο που μοιάζει πολύ με το αρχικό. Επίσης έχουμε 15 κοινά σημεία με την 4.6 (mod4_small.jpg) η οποία είναι φωτογραφία του αρχικού λογοτύπου αλλά με γωνία λήψης από πάνω. Για τις υπόλοιπες φωτογραφίες έχουμε μονοψήφιο αριθμό κοινών σημείων και αυτό διότι είτε έχουν πολύ περιορισμένο bounding box (σκοπίμως επιλεγμένο από μας για καθαρά λόγους testing) είτε είναι τελείως άσχετες με το περιεχόμενο (π.χ το λογότυπο της Coca-Cola με τα 6 κοινά σημεία, τα πιο λίγα συνολικά). Εδώ να σημειώσουμε πως η φωτογραφία που τραβήξαμε με το κινητό είναι διαστάσεων 585×777 pixels και μεγέθους μόλις 82 KB λόγω συμπίεσης, επομένως είναι επόμενο να έχουμε ακόμα περισσότερη απώλεια πληροφορίας και λιγότερα επί μέρους κοινά σημεία με τη βάση εικόνων.

Ο συνολικός χρόνος για όλη την εκτέλεση διήρκησε λιγότερο από 32 δευτερόλεπτα, ενώ το μεγαλύτερο κομμάτι του (20 περίπου seconds) το κατανάλωσε η μεταφορά της φωτογραφίας από το κινητό προς τον εξυπηρετητή ιστού μέσω του ασύρματου τοπικού δικτύου. Ο χρόνος μεταφοράς παραμένει ένα σημαντικό bottleneck και αυτό κυρίως λόγω της φύσης των ασυρμάτων δικτύων που επιδέχονται αρκετό θόρυβο και συνεπώς σφάλματα στα πακέτα. Παρ΄όλα αυτά, η παραλληλοποίηση της εφαρμογής μας, που επιτρέπει στο χρήστη να πάρει κι άλλες φωτογραφίες ή να κάνει οτιδήποτε άλλο επιθυμεί, ενώ μεταφέρεται η αρχική εικόνα, εξουδετερώνει εν μέρει το σχετικά μεγάλο χρόνο αναμονής.

4.4 Παίγνια και προεκτάσεις

Η εποχή του gaming layer που κάθεται πάνω από το σοσιαλ νετωορκινγ λαψερ που έχτισαν το Facebook και οι παρεμφερείς υπηρεσίες κοινωνικής δικτύωσης, έχει φτάσει. Ο ιδρυτής του scvngr, της διάσημης social location-based gaming πλατφόρμας, Seth Priebatsch, στην ομιλία του στο TED (gaming layer TED talk) καθώς και ο Jesse Schell, πρώην Creative Director του Disney Imagineering Virtual Reality Studio με την ομιλία του για

την αυξανόμενη διείσδυση των παιγνίων στην πραγματική ζωή (games invade real life) είναι προάγγελοι της ανερχόμενης εποχής αυτής. Η ραγδαία εξελισσόμενη δικτύωση με αισθητήρες και μικρές έξυπνες συσκευές με επίκεντρο ίσως την ισχυρή κινητή συσκευή που σχεδόν όλοι πλέον διαθέτουν, επιτρέπει τη μετατροπή των καθημερινών δραστηριοτήτων σε παίγνια. Η συλλογή πόντων μέσω εκπλήρωσης συγκεκριμένων πράξεων (πχ ανακύκλωση όπου οι κάδοι σκουπιδιών θα διαθέτουν αισθητήρες) και η δυνατότητα εξαργύρωσης τους στην πραγματική ζωή (πχ μέσω επιστροφής φόρου) αποτελούν επερχόμενες δραστηριότητες της εποχής αυτής. Το Byakugan μπορεί ήδη στη μορφή που είναι αλλά ακόμα περισσότερο με περαιτέρω βελτίωση, να αποτελέσει ένα κομμάτι του κόσμου αυτού.

Το σύστημα μας, ώντας μια ολοκληρωμένη πλατφόρμα λογισμικού, μπορεί εύκολα να μετατραπεί σε παίγνιο της λογικής ¨κυνήγι θησαυρού¨ (treasure hunting). Ο χρήστης θα καλείται να περιηγηθεί σε μια τοποθεσία, στα πλαίσια πχ μιας πόλης, ενός πανεπιστημιακού campus ή μιας καφετέριας, να ανιχνεύσει τα κρυμμένα ή μη λογότυπα που έχουν στήσει οι εταιρείες, να τα φωτογραφίσει και να τα στείλει μέσω ενός τοπικού ασύρματου δικτύου ή μέσω 3G κλπ. Η όλη διαδικασία μπορεί να γίνεται πλήρως αυτοματοποιημένα χάρη στην έξυπνη αναγνώριση λογοτύπων του Byakugan. Αυτό έχει σαφώς και επιχειρηματικές προεκτάσεις, καθώς παράλληλα γίνεται διαφήμιση των εν λόγω brands των οποίων τα λογότυπα καλείται να φωτογραφίσει ο χρήστης.

Η μία εκδοχή του παιχνιδιού όπου η διαφημιστική εταιρεία είναι από μια συγκεκριμένη μάρκα απαιτεί διάφορες εκδόσεις του λογοτύπου της εταιρείας και μόνο, οδηγώντας σε μια σχετικά μικρή βάση-εικόνων. Αντίστοιχα, σε μια άλλη εκδοχή του παιχνιδιού όπου ο χρήστης καλείται να φωτογραφίσει όσα περισσότερα λογότυπα (διαφορετικών εταιρειών) μπορεί μέσα στην τοποθεσία που ορίζεται, οι απαιτήσεις είναι κυρίως ως προς τη μεγάλη ποικιλία των εικόνων. Παρ'όλα αυτά, σε κάθε περίπτωση, αφού οι εταιρείες ή οι διαχειριστές που θα στήσουν τα λογότυπα στην τοποθεσία (πχ μέσω billboards σε μια πόλη ή μέσω της ετικέτας της φιάλης του αναψυκτικού/ποτού στην καφετέρια) γνωρίζουν εκ των προτέρων τις διάφορες πιθανές εκδοχές των βασικών λογοτύπων, γίνεται αρκετά εύκολο να συμπεριληφθούν αυτές μέσα στη βάση-εικόνων και το μέγεθος της και επομένως οι ενδεχόμενες συγκρίσεις να

διατηρηθούν σχετικά μικρά.

Οι προγραμματιστικές προεκτάσεις που μπορεί να λάβει το πρόγραμμα μας είναι αρκετές. Αναφέρουμε εδώ μερικές συνοπτικά:

1. Βελτίωση της εφαρμογής Android με δυνατότητα επιλογής να στέλνεται ως απάντηση καλύτερου match ολόκληρη η εικόνα αντί για το όνομα της μόνο.

2. Μεταφορά της εφαρμογής για κινητές συσκευές και σε iOS για iPhone και iPad.

3. Μεταφορά του πυρήνα και σε Windows.

4. Βελτίωση του server με δυνατότητα επέκτασης στο Cloud και την πλήρη παραλληλοποίηση του μέσω των δυνατοτήτων που προσφέρονται από το ¨νέφος¨.

5. Δυναμική εναλλαγή άλλων αλγορίθμων επεξεργασίας εικόνας (όπως SIFT) αν το σύστημα κρίνει ότι χρειάζεται περισσότερη ακρίβεια ενώ μπορεί να θυσιάσει μέρος της ταχύτητας.

6. Δυνατότητα χρήσης άλλων δικτύων πέραν των τοπικών ασυρμάτων για την επικοινωνία με τον εξυπηρετητή (πχ 3G, αποστολή εικόνας μέσω MMS κλπ)

Κεφάλαιο 5

Υλοποίηση

> The person who gets the farthest is generally the one who is willing to do and dare. The sure-thing boat never gets far from shore.
>
> Dale Carnegie

5.1 Εισαγωγή

Στο κεφάλαιο αυτό, παρατίθεται ο πλήρης κώδικας της υλοποίησης του συστήματος μας. Παραλείπονται τα κομμάτια των βιβλιοθηκών (πχ TinyXML, OpenCV, Boost) που χρησιμοποιήθηκαν έτοιμα. Το εξωτερικό λογισμικό το οποίο χρησιμοποιήθηκε για τη μετάφραση του πυρήνα και δεν περιλαμβάνεται μέσα στο πακέτο, είναι:
OpenCV-2.3.1
Boost C++ library 1.49
Για τον web server χρειάζεται:
Python 2.7.2
Για την εφαρμογή Ανδροιδ χρησιμοποιήθηκαν τα:
Android SDK 2.2
Eclipse 3.7.2

5.2 Byakugan Core

5.2.1 byakugan.cc

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdarg.h>
4  #include <getopt.h>
5  #include <unistd.h>
6  #include "Byakugan.h"
7  #include "ByakuganOps.h"
8
9  using namespace cv;
10
11 static void print_usage(void);
12 static char *grab_next_file(FILE *inputfd, int argc, char **argv);
13
14
15 static void
16 print_usage(void)
17 {
18   printf("byakugan \n"
19       "Usage: two modes of operation:\n"
20       "        [+] SCAN mode\n"
21       "         - scan given directories for images and compute the needed\n"
22       "         - keypoints which will be saved in XML format\n"
23       "        ./byakugan -s <directory_1> <directory_2>...\n"
24       "        ./byakugan -s -i <file_with_list_of_directories>\n"
25       "        [+] PROCESS mode\n"
26       "         - compare <query_image> against all images which have been\n"
27       "         - previously been decomposed and saved into XML-encoded format\n"
28       "        ./byakugan -g -p <XML_file_1> <XML_file_2>... -q <query_image>\n"
29       "        ./byakugan -g -p -i <file_with_list_of_XML_files> -q <query_image>\n"
30       "OPTIONS:\n"
31       "        -s: scan mode\n"
32       "        -p: process mode\n"
33       "        -q <query_image>: image to compare against\n"
34       "        -i <input_filename: get list of directories/files from this file\n"
35       "        -g: Show correspondence graph with matching keypoints\n"
36       "        -v: Increase verbosity level (use twice or more for greater effect)\n"
37       "        -d[level]: Set or increase debugging level (Up to 10 is meaningful)\n"
38       "        -h: Print this help summary page.\n"
39       "        -V: Print version number\n"
40       );
41
42   exit(EXIT_FAILURE);
43
44 }
```

```
45
46
47  static char *
48  grab_next_file(FILE *inputfd, int argc, char **argv)
49  {
50    static char file[1024];
51    unsigned int file_index;
52    int ch;
53
54    if (!inputfd) {
55      return ((optind < argc) ? argv[optind++] : NULL);
56    } else {
57        file_index = 0;
58        while((ch = getc(inputfd)) != EOF) {
59          if (ch == ' ' || ch == '\r' || ch == '\n' || ch == '\t' || ch == '\0') {
60            if (file_index == 0)
61              continue;
62            file[file_index] = '\0';
63            return file;
64          } else if (file_index < sizeof(file) / sizeof(char) -1) {
65            file[file_index++] = (char) ch;
66          } else fatal("One of the fileifications from your input file "
67              "is too long (> %d chars)", (int) sizeof(file));
68        }
69        file[file_index] = '\0';
70      }
71    if (!*file)
72      return NULL;
73    return file;
74  }
75
76
77
78
79
80  int
81  main(int argc, char **argv) {
82
83    FILE *inputfd = NULL;
84    char *file = NULL;
85    char *file_copy = NULL;
86    int arg;
87    int option_index;
88    extern char *optarg;
89    extern int optind;
90    struct option long_options[] =
91    {
92      {"scan", no_argument, 0, 's'},
93      {"process", no_argument, 0, 'p'},
94      {"version", no_argument, 0, 'V'},
95      {"verbose", no_argument, 0, 'v'},
```

```
96    {"debug", optional_argument, 0, 'd'},
97    {"help", no_argument, 0, 'h'},
98    {0, 0, 0, 0}
99  };
100
101 if (argc < 2)
102   print_usage();
103
104 /* Argument parsing */
105 optind = 1;
106 while((arg = getopt_long_only(argc, argv, "d:hi:gpq:sv::V",
107         long_options, &option_index)) != EOF) {
108   switch(arg) {
109
110     case 0:
111       break;
112     case 'd':   /* debugging output level */
113       if (optarg && isdigit(optarg[0])) {
114         o->debugging = o->verbose = atoi(optarg);
115       } else {
116         const char *p;
117         o->debugging++;
118         o->verbose++;
119         for (p = optarg != NULL ? optarg : ""; *p == 'd'; p++) {
120           o->debugging++;
121           o->verbose++;
122         }
123         if (*p != '\0')
124           fatal("Invalid argument to -d: \"%s\".", optarg);
125       }
126       break;
127     case 'h':   /* help */
128       print_usage();
129       break;
130     case 'i':
131       if (inputfd) {
132         fatal("Only one input filename allowed");
133       }
134       if (!strcmp(optarg, "-")) {
135         inputfd = stdin;
136       } else {
137         inputfd = fopen(optarg, "r");
138         if (!inputfd) {
139           fatal("Failed to open input file %s for reading", optarg);
140         }
141       }
142       break;
143     case 'g': /* show correspondence img with matching keypoints */
144       o->show_graph = true;
145       break;
146     case 'p': /* process mode */
```

```
147         o->process = true;
148         break;
149       case 'q': /* query image for process mode */
150          o->query = strndup(optarg, strlen(optarg));
151          break;
152       case 's': /* scan mode */
153         o->scan = true;
154         break;
155       case 'V': /* show version */
156         printf("\nbyakugan version 1.0alpha\n");
157         exit(EXIT_SUCCESS);
158         break;
159       case 'v': /* verbosity level */
160         if (optarg && isdigit(optarg[0])) {
161           o->verbose = atoi(optarg);
162         } else {
163           const char *p;
164           o->verbose++;
165           for (p = optarg != NULL ? optarg : ""; *p == 'v'; p++)
166             o->verbose++;
167           if (*p != '\0')
168             fatal("Invalid argument to -v: \"%s\".", optarg);
169         }
170         break;
171       case '?':   /* error */
172         print_usage();
173     }
174   }
175
176   if (o->scan && o->process)
177     fatal("You can specify either only 'process' (-p) mode or 'scan' (-s) mode!\n");
178
179   if (o->query && !o->process)
180     fatal("You have specified a query image without specifying 'process' (-p) mode!\n");
181
182   if (o->process && !o->query)
183     fatal("You have specified 'process' (-p) mode specified and NO query (-q) image!\n")↩
         ;
184
185   while ((file = grab_next_file(inputfd, argc, argv))) {
186
187     file_copy = strndup(file, strlen(file));
188     Byakugan::byakugan()->files.push_back(file_copy);
189
190   }
191
192   if (o->scan && Byakugan::byakugan()->files.empty())
193     fatal("You have specified scan mode and zero directories to scan!\n");
194
195   if (o->process && Byakugan::byakugan()->files.empty())
196     fatal("You have specified process mode and zero files to process!\n");
```

```
197
198   if (o->scan) {
199     Byakugan::byakugan()->scan();
200     exit(EXIT_SUCCESS);
201   }
202
203   Byakugan::byakugan()->process();
204
205
206   exit(EXIT_SUCCESS);
207 }
```

5.2.2 Byakugan.h

```
1  #pragma once
2  #include "Image.h"
3  #include "TrainingSet.h"
4  #include "ByakuganOps.h"
5  #include <boost/filesystem.hpp>
6
7  using namespace std;
8  using namespace cv;
9
10 /* Byakugan is the main class and a singleton */
11 class Byakugan {
12
13   public:
14
15     static Byakugan *byakugan();  /* get instance */
16
17     /*
18      * Process mode:
19      * Loads saved and decomposed images from XML files as created by the
20      * scan() function and then compares them against the query image as
21      * specified by user input (-q).
22      */
23     int process(void);
24
25     /*
26      * Recursively scans all directories as previously specified by user
27      * input and saved into "files" and then searches for images. For each
28      * directory it makes a TrainingSet out of the images and then creates an
29      * XML file with computed keypoints and descriptors for each set.
30      */
31     void scan(void);
32
33     vector <char *>files; /* directories/files to scan/process images */
34
```

```
35   private:
36
37     Byakugan();
38     Byakugan(Byakugan const&) {};
39     Byakugan & operator=(Byakugan const &){};
40     ~Byakugan();
41     static Byakugan *instance; /* instance */
42
43     /*
44      * recursive use by scan function
45      */
46     void scan_internal(fs::path dir, int level);
47
48     SurfFeatureDetector* detector;
49     SurfDescriptorExtractor* extractor;
50     int max_levels; /* maximum levels of recursive directory scanning */
51     Image query;    /* query image */
52     Image gray;
53     vector<TrainingSet*> trainingSets;
54
55 };
```

5.2.3 Byakugan.cc

```
1  #include "Byakugan.h"
2
3  /* singleton */
4  Byakugan *Byakugan::instance = NULL;
5
6
7  Byakugan *Byakugan::
8  byakugan()
9  {
10
11   if (!instance)
12     instance = new Byakugan();
13
14   return instance;
15 }
16
17
18 Byakugan::
19 Byakugan()
20 {
21
22   double hessianThreshold = 400;
23   int octaves = 3;
24   int octaveLayers = 4;
```

```
25    detector = new SurfFeatureDetector(hessianThreshold, octaves, octaveLayers);
26    extractor = new SurfDescriptorExtractor();
27    max_levels = 10;
28
29  }
30
31
32
33  Byakugan::
34  ~Byakugan()
35  {
36    for (int i = 0; i < trainingSets.size(); i++) {
37      delete trainingSets[i];
38    }
39
40  }
41
42
43  /*
44   * Scan mode:
45   * Recursively scans all directories as previously specified by user
46   * input and saved into "files" and then searches for images. For each
47   * directory it makes a TrainingSet out of the images and then creates an
48   * XML file with computed keypoints and descriptors for each set.
49   */
50  void Byakugan::
51  scan(void)
52  {
53    printf("[+] Scan mode.\n");
54
55    for(int i = 0; i < files.size(); i++) {
56      scan_internal(fs::path(files[i]), 0);
57    }
58
59    printf("Scanning done.\n");
60
61  }
62
63
64  /*
65   * recursive use by scan function
66   */
67  void Byakugan::
68  scan_internal(fs::path dir, int level)
69  {
70
71  if (level > max_levels)
72      return;
73
74    /* make a TrainingSet out of the images in this directory */
75    TrainingSet set;
```

```
76    set.setFeatureDetector(detector);
77    set.setDescriptorExtractor(extractor);
78
79    set.loadFromDirectory(dir.string());
80    set.save();
81
82    fs::directory_iterator end_iter;
83    for (fs::directory_iterator dir_itr(dir); dir_itr != end_iter; ++dir_itr) {
84      if (fs::is_directory(dir_itr->path())) {
85          scan_internal(dir_itr->path(), level + 1);
86      }
87    }
88
89  }
90
91
92  /*
93   * Process mode:
94   * Loads saved and decomposed images from XML files as created by the
95   * scan() function and then compares them against the query image as
96   * specified by user input (-q).
97   */
98  int Byakugan::
99  process(void)
100 {
101   Mat input_image;
102
103   printf("[+] Process mode.\n");
104
105   /* read in input image to compare against */
106   input_image = imread(o->query, 0);
107   query.useMat(input_image);
108
109   for(int i = 0; i < files.size(); i++) {
110     TrainingSet* set = new TrainingSet();
111
112     set->setFeatureDetector(cv::FeatureDetector::create("SURF"));
113     set->setDescriptorExtractor(cv::DescriptorExtractor::create("SURF"));
114
115     set->loadFromArchive(files[i]);
116     trainingSets.push_back(set);
117   }
118
119   vector<TrainingSet *>::iterator it;
120   int i = 0;
121   for (it = trainingSets.begin(); it != trainingSets.end(); it++) {
122
123     if (o->verbose)
124       printf("i: %d, images_count: %d\n", i, (*it)->images.size());
125
126     i++;
```

```
127   }
128
129
130   /* vector holding the best matches from each Training Set - each index
131    * and value refers to the index of the image of each TrainingSet that best
132    * matches the query image
133    */
134   vector <bm>matches;
135
136   /* process */
137   for (int i = 0; i < trainingSets.size(); i++) {
138     matches.push_back(trainingSets[i]->match(query));
139   }
140
141   vector<bm>::iterator bm_it;
142
143 #if 0
144   for (bm_it = matches.begin(); bm_it != matches.end(); bm_it++)
145     printf("%d %d \n", bm_it->index, bm_it->matches);
146 #endif
147
148   /* find the index of the image with the highest matching value from all
149    * images */
150   int max = matches[0].matches;
151   int set_index = 0;
152   vector<int>::size_type image_index = matches[0].index;
153
154   for (i = 0, bm_it = matches.begin(); bm_it != matches.end(); bm_it++, i++) {
155     if (bm_it->matches > max) {
156       max = bm_it->matches;
157       image_index = bm_it->index;
158       set_index = i;
159     }
160   }
161
162   if (o->debugging)
163     printf("set:%d image:%d matches:%d path:%s\n",
164       set_index, image_index, max,
165       trainingSets[set_index]->images[image_index].imgfile.c_str());
166
167   printf("Best match: %s\n", trainingSets[set_index]->images[image_index].imgfile.c_str↩
        ());
168
169   printf("Processing done.\n");
170 }
```

5.2.4 ByakuganOps.h

```
#pragma once

/* User-provided options are saved into this singleton class */
class ByakuganOps {
  public:

    static ByakuganOps* get_instance();

    int verbose;  /* verbosity level */
    int debugging;/* debugging level */
    bool scan;    /* scan mode */
    bool process; /* process mode */
    char *query;  /* query image filename */
    bool show_graph; /* show correspondence img with matching keypoints */

  private:
    ByakuganOps();
    ByakuganOps(ByakuganOps const&) {};
    ByakuganOps & operator=(ByakuganOps const &){};
    ~ByakuganOps();
    static ByakuganOps *ops_instance; /* instance */

};

extern ByakuganOps* o;
```

5.2.5 ByakuganOps.cc

```
#include "ByakuganOps.h"
#include <stddef.h>

/* singleton */
ByakuganOps *ByakuganOps::ops_instance = NULL;

ByakuganOps::
ByakuganOps()
{
  verbose = 0;
  debugging = 0;
  scan = false;
  process = false;
  query = NULL;
  show_graph = false;
}

ByakuganOps *ByakuganOps::
get_instance()
```

```
20 {
21   if (!ops_instance)
22     ops_instance = new ByakuganOps();
23
24   return ops_instance;
25
26 }
27
28 ByakuganOps* o = ByakuganOps::get_instance();
```

5.2.6 Image.h

```
1  #pragma once
2  #include <opencv2/opencv.hpp>
3  #include <boost/filesystem.hpp>
4
5  using namespace cv;
6  using namespace std;
7  namespace fs = boost::filesystem;
8
9  class Image {
10 public:
11
12   // Constructors
13   Image(){}
14   Image(Mat m)              { m.copyTo(cvImg);            }
15   Image(Size size, int type)        { cvImg.create(size, type);       }
16   Image(Size size)            { cvImg.create(size, CV_8UC3);      }
17   Image(int width, int height, int type)  { cvImg.create(height, width, type);    }
18   Image(int width, int height)      { cvImg.create(height, width, CV_8UC3); }
19
20   // Operator overloading
21   Image& operator=(const Image &other);
22   Image clone() { return *(new Image(cvImg.clone())); }
23
24
25   // Convenience methods
26   Size size()       { return cvImg.size();    }
27   int width()       { return cvImg.size().width;  }
28   int height()      { return cvImg.size().height; }
29   int channels()      { return cvImg.channels();  };
30   void convert(int code)  { cvtColor(cvImg, cvImg, code);     }
31   void add(Image &img)  { cv::add(img.cvImg, cvImg, cvImg);   }
32   void dilate(int amt)  { cv::dilate(cvImg, cvImg, Mat(), Point(-1,-1), amt); }
33   void erode(int amt)   { cv::erode(cvImg, cvImg, Mat(), Point(-1,-1), amt);  }
34   void blur(int amt)    { cv::blur(cvImg, cvImg, Size(amt,amt));    }
35   void invert()     { cv::bitwise_not(cvImg, cvImg);  }
36   void equalize()     { equalizeHist(cvImg, cvImg);   }
```

```
37  void show(string wn)  { cv::imshow(wn, cvImg);    }
38  bool empty()      { return cvImg.empty();   }
39
40
41  // Find edges on an image
42  void canny(double thresh1=5, double thresh2=50, int aperatureSize=3, bool moreAccurate↩
        =false);
43  void soebel(bool xfirst=true, int ksize=3);
44
45  // Apply a threshold to an image
46  void threshold(int t, bool inverse=false);
47  void adaptiveThreshold(bool inverse=false, bool gaussian=true);
48
49  // Point this image to the data in m
50  void useMat( cv::Mat m );
51
52  // Open an image from disk
53  bool open(string filename, bool forceGrayscale=false);
54
55  // Write some text into the image
56  void text(const string txt, int x, int y, float scale=1);
57
58  // Get a section from another image
59  void copySectionFrom(Image src, Rect src_rect);
60  void copySectionFrom(Image src, Rect src_rect, Rect dst_rect);
61
62  // Copy this image into another at a particular location
63  void copyInto(Image background, int x, int y, Image mask);
64  void copyInto(Image background, int x, int y);
65
66  // Show this image using HighGUI
67  void show(string windowname, int x, int y);
68
69  // Get an image for each channel in this image
70  vector<Image> split();
71
72  // Get a matrix of the image as a list of pixels (ie: 1 column, w*h rows)
73  Mat asList();
74
75  // Set the depth of values  eg. CV_8U, CV_32F
76  void setDepth(int newType);
77
78  // Get a particular number of kmeans clusters of colors
79  float* getClusters(int clusterCount);
80
81  // Get some copies of this image
82  vector<Image> copies(int num);
83
84  // Get a grayscale version of this image.
85  Image getGrayscale();
86
```

```
87    // Get a reference to the cv::Mat
88    Mat* getCvImg() { return &cvImg; }
89
90  //protected:
91    cv::Mat cvImg;
92    Rect roi;
93    fs::path imgfile;     // path to the image file
94  };
```

5.2.7 Image.cc

```
1   #include "Image.h"
2
3
4   // ----------------------------------------
5   void Image::useMat( cv::Mat m )
6   {
7     cvImg = m;
8   }
9
10
11  // ----------------------------------------
12  Image& Image::operator=(const Image &other)
13  {
14    if (this == &other)      // Same object?
15      return *this;        // Yes, so skip assignment, and just return *this.
16    other.cvImg.copyTo( cvImg );
17    return *this;
18  }
19
20  // ----------------------------------------
21  bool Image::open(string filename, bool forceGrayscale)
22  {
23    int gray = (forceGrayscale) ? 0 : 1;
24    cvImg = cv::imread(filename, gray);
25    return !cvImg.empty();
26  }
27
28
29  // ----------------------------------------------------
30  void Image::threshold(int t, bool inverse)
31  {
32    int thresholdType = inverse ? THRESH_BINARY_INV : THRESH_BINARY;
33    cv::threshold(cvImg, cvImg, t, 255, thresholdType);
34  }
35
36  // ----------------------------------------------------
37  void Image::adaptiveThreshold(bool inverse, bool gaussian)
```

```
38 {
39   int adaptiveMethod = gaussian ? ADAPTIVE_THRESH_GAUSSIAN_C : ADAPTIVE_THRESH_MEAN_C;
40   int thresholdType = inverse ? THRESH_BINARY_INV : THRESH_BINARY;
41   cv::adaptiveThreshold(cvImg, cvImg, 255, adaptiveMethod, thresholdType, 5, 5);
42 }
43
44 // ------------------------------------------------------
45 void Image::soebel(bool xfirst, int ksize)
46 {
47   Sobel(cvImg, cvImg, CV_8U, (xfirst)?1:0, (xfirst)?0:1, ksize);
48 }
49
50 // ------------------------------------------------------
51 void Image::canny(double thresh1, double thresh2, int aperatureSize, bool moreAccurate)
52 {
53   Canny(cvImg, cvImg, thresh1, thresh2, aperatureSize, moreAccurate);
54 }
55
56
57 // ------------------------------------------------------
58 void Image::text(const string txt, int x, int y, float scale)
59 {
60   putText(cvImg, txt, Point(x,y), FONT_HERSHEY_PLAIN, 1, Scalar(255,255,255));
61 }
62
63 // ------------------------------------------------------
64 void Image::copySectionFrom(Image src, Rect src_rect)
65 {
66   copySectionFrom(src, src_rect, src_rect);
67 }
68
69 // ------------------------------------------------------
70 Image Image::getGrayscale()
71 {
72   int code;
73   switch(channels())
74   {
75     case 1: return this->clone();
76     case 3: code=CV_RGB2GRAY; break;
77     case 4: code=CV_RGBA2GRAY; break;
78   }
79
80   Mat gray;
81   cvtColor(cvImg, gray, code);
82   Image* grayscale = new Image(gray);
83   return *grayscale;
84 }
85
86 // ------------------------------------------------------
87 void Image::copySectionFrom(Image src, Rect src_rect, Rect dst_rect)
88 {
```

```
89    Mat src_roi = src.cvImg(src_rect);
90    Mat dst_roi = cvImg(dst_rect);
91    src_roi.copyTo(dst_roi);
92  }
93
94  // ------------------------------------------------------
95  void Image::copyInto(Image background, int x, int y, Image mask)
96  {
97    Rect bg = Rect(0, 0, background.width(), background.height());
98    Rect fg = Rect(x, y, width(), height());
99    Rect intersect = fg & bg;
100
101   Mat foreground_roi = cvImg( Rect(0, 0, intersect.width, intersect.height) );
102   Mat mask_roi = mask.cvImg( Rect(0, 0, intersect.width, intersect.height) );
103   Mat background_roi = background.cvImg(Rect(x, y, intersect.width, intersect.height));
104
105
106   foreground_roi.copyTo(background_roi, mask_roi);
107 }
108
109
110 // ------------------------------------------------------
111 void Image::copyInto(Image background, int x, int y)
112 {
113   copyInto(background, x, y, Image(Mat::ones(size(), CV_8UC1)));
114 }
115
116
117 // ------------------------------------------------------
118 void Image::show(string windowname, int x, int y)
119 {
120   namedWindow(windowname);
121   cvMoveWindow(windowname.c_str(), x, y);
122   imshow(windowname, cvImg);
123 }
124
125
126 // ------------------------------------------------------
127 vector<Image> Image::split()
128 {
129   vector<Mat> cvPlanes;
130   vector<Image> planes;
131   cv::split(cvImg, cvPlanes);
132   for(int i=0; i<cvPlanes.size(); i++)
133   {
134     Image img(cvPlanes[i]);
135     planes.push_back( img );
136   }
137   return planes;
138 }
139
```

```
140 // ------------------------------------------------------
141 // TO DO: There must be a better way to do this.  Maybe post on opencv list
142 // Maybe we can elongate it to make an extremely wide image and then rotate it.
143 // NOTE: Not sure this works with >1 channel images
144 Mat Image::asList()
145 {
146   int rows = cvImg.total();
147   int cols = 1;
148   Mat list(rows, cols, cvImg.type());
149   uchar* src;
150   uchar* dest = list.ptr(0);
151   for(int i=0; i<cvImg.size().height; i++) {
152     src = cvImg.ptr(i);
153     memcpy(dest, src, cvImg.step);
154     dest += cvImg.step;
155   }
156   return list;
157 }
158
159
160 // ------------------------------------------------------
161 void Image::setDepth(int newType)
162 {
163   cvImg.convertTo(cvImg, newType);
164 }
165
166
167 // ------------------------------------------------------
168 float* Image::getClusters(int clusterCount)
169 {
170   Mat list = asList();
171   list.convertTo(list, CV_32F);
172
173   Mat labels(list.size(), CV_8UC1);
174   Mat centers(clusterCount, 1, list.type());
175   TermCriteria termcrit(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, 10, 1.0);
176   kmeans(list, clusterCount, labels, termcrit, 3, KMEANS_PP_CENTERS, centers);
177   float* c = new float[clusterCount];
178   memcpy(c, centers.ptr<float>(0), sizeof(float)*clusterCount);
179   return c;
180 }
181
182 // ------------------------------------------------------
183 vector<Image> Image::copies(int num)
184 {
185   vector<Image> clones;
186   for(int i=0; i<num; i++) {
187     Image aCopy( cvImg );
188     clones.push_back( aCopy );
189   }
190   return clones;
```

```
191 }
```

5.2.8 TrainingImage.h

```
1  #pragma once
2  #define MARGIN 50
3
4  #include "Matcher.h"
5  #include "Image.h"
6  #include "ByakuganOps.h"
7  #include <boost/filesystem.hpp>
8  #include <boost/algorithm/string.hpp>
9  #include <fstream>
10 #include <iostream>
11
12 using namespace std;
13 using namespace cv;
14 namespace fs = boost::filesystem;
15
16 class Matcher;
17
18
19 class TrainingImage : public Image {
20 public:
21
22   TrainingImage();
23
24   /*
25    * Computes the ROI (region of interest) by intelligently gathering the
26    * user-generated data from the cnr files.
27    * It gets the "corners" residing inside the .cnr files and comprehends
28    * which of the coordinates correspond to which corner from
29    * top-left to bottom right.
30    * Returns a rectangle which defines the ROI.
31    */
32   Rect create_outline();
33
34   /*
35    * Loads in the "corners" from the cnr file, or if it doesn't exist, pops open
36    * a window that allows the user to mark the corners
37    */
38   void generate_cnrfile();
39
40   /*
41    * If an image has a name like myimage.jpg?fooblah=boo, rename it myimage.jpg
42    */
43   void remove_query_string();
44
```

```
45    /*
46     * We need to keep track of the image filename, so we override the open
47     * method from Image
48     */
49    void open(string _imgfile);
50
51    /*
52     * OpenCV mouse listener
53     */
54    static void on_mouse( int event, int x, int y, int flags, void* param );
55
56    /*
57     * loads the TrainingImage data into the Matcher and then performs the
58     * query against "image"
59     */
60    int match(Image image);
61
62    //protected:
63    fs::path cnrfile;     // path to the corners file
64    vector<KeyPoint> keypoints;
65    Mat descriptors;
66
67
68    Mat image_w_margin;
69    vector<Point> corners;
70  };
```

5.2.9 TrainingImage.cc

```
1   #include "TrainingImage.h"
2   #include "Matcher.h"
3
4   TrainingImage::
5   TrainingImage() {
6
7
8   }
9
10
11  /*
12   * Computes the ROI (region of interest) by intelligently gathering the
13   * user-generated data from the cnr files.
14   * It gets the "corners" residing inside the .cnr files and comprehends
15   * which of the coordinates correspond to which corner from
16   * top-left to bottom right.
17   * Returns a rectangle which defines the ROI.
18   */
19  Rect TrainingImage::
```

```
20 create_outline()
21 {
22
23   vector <Point>::iterator it;
24   int small_x, small_y, pos_x, pos_y, big_x, big_y;
25   int i;
26   Point top_left;
27   Point bottom_right;
28   int width, height;
29   Rect ret;
30
31
32   /* find smallest x coordinate */
33   small_x = corners[0].x;
34   for (i = 0, pos_x = 0, it = corners.begin(); it != corners.end(); it++, i++) {
35     if (it->x < small_x) {
36       small_x = it->x;
37       pos_x = i;
38     }
39   }
40
41   /* find smallest y coordinate */
42   small_y = corners[0].y;
43   for (i = 0, pos_y =0, it = corners.begin(); it != corners.end(); it++, i++) {
44     if (it->y < small_y) {
45       small_y = it->y;
46       pos_y = i;
47     }
48   }
49
50   /* top left corner will be a point which honors the constraint of having
51    * the smallest x and smallest y coordinates:
52    * in the example below the intersecting lines of the 2 different points
53    * create the top left corner of our outline to be. C has the smallest y
54    * while B has the smallest x.
55    *
56    * 0
57    * ------------------------------------------> X
58    * |
59    * |     x-----------------B
60    * |     |   A
61    * |     |
62    * |     |
63    * |     |
64    * |     |               D
65    * |     C
66    * |
67    * v
68    * Y
69    */
70   top_left.x = small_x;
```

```
71    top_left.y = small_y;
72
73    if (o->debugging > 5) {
74      printf("small_x %d, pos: %d \n", small_x, pos_x);
75      printf("small_y %d, pos: %d \n", small_y, pos_y);
76    }
77
78    /* Now to form the rest of the rectangle, we need the biggest x and y
79     * available, because in cases like the above, if point D defined the
80     * bottom right corner of our rectangle we would be probably losing mportant
81     * information of the object
82     */
83
84    /* find biggest x coordinate */
85    big_x = corners[0].x;
86    for (it = corners.begin(); it != corners.end(); it++) {
87      if (it->x > big_x)
88        big_x = it->x;
89    }
90
91    /* find biggest y coordinate */
92    big_y = corners[0].y;
93    for (it = corners.begin(); it != corners.end(); it++) {
94      if (it->y > big_y)
95        big_y = it->y;
96    }
97
98    bottom_right.x = big_x;
99    bottom_right.y = big_y;
100
101   /* calculate width and height of our rectangle */
102   width = bottom_right.x - corners[pos_x].x;
103   height = bottom_right.y - corners[pos_y].y;
104
105   if (o->debugging > 5) {
106     printf("big_x %d, big_y %d\n", big_x, big_y);
107     printf("width: %d  height: %d\n", width, height);
108   }
109
110   ret = Rect(top_left.x, top_left.y, width, height);
111   return ret;
112
113 }
114
115
116
117 /*
118  * Loads in the "corners" from the cnr file, or if it doesn't exist, pops open
119  * a window that allows the user to mark the corners
120  */
121 void TrainingImage::
```

```
122 generate_cnrfile()
123 {
124   corners.clear();
125   cnrfile = imgfile;
126   cnrfile.replace_extension(".cnr");
127 
128   ifstream myfile( cnrfile.string().c_str() );
129   if (myfile.is_open()) {
130     string line;
131     while (myfile.good()) {
132 
133       getline(myfile, line);
134 
135       if (!line.empty()) {
136         Point p;
137         stringstream ss(line);
138         ss >> p.x >> p.y;
139         corners.push_back( p );
140       }
141     }
142 
143     vector<Point>::iterator it;
144 
145     if (o->debugging) {
146       printf("------corners------\n");
147       for (it = corners.begin(); it != corners.end(); it++) {
148         printf("x: %d  y: %d\n", it->x, it->y);
149       }
150     }
151 
152     myfile.close();
153     return;
154   }
155 
156   printf("No corners file found. Lets create one.\n");
157 
158 
159   /* We need to construct an image with a margin around it
160    * This is because the bounds of the "plane" on which the object exists might
161    * extend outside of the image.
162    */
163   image_w_margin = Mat(Size(cvImg.size().width+(MARGIN*2),
164         cvImg.size().height+(MARGIN*2)), cvImg.type());
165   image_w_margin = Scalar(255,255,255);
166 
167   /* Copy the image from the file into the one with the margin */
168   Mat roi = image_w_margin(Rect(50, 50, cvImg.size().width, cvImg.size().height));
169   cvImg.copyTo(roi);
170 
171   /* Show the image */
172   imshow("image", image_w_margin);
```

```
173   setMouseCallback("image", on_mouse, this);
174
175   bool done=false;
176   do {
177
178     int key = waitKey(100);
179
180     if (key == 27 || corners.size() > 3)
181       done = true;
182     if (key == 'c')
183       generate_cnrfile();
184
185     imshow("image", image_w_margin);
186   } while (!done);
187
188   if (corners.size() == 4) {
189
190     ofstream myfile;
191     myfile.open(cnrfile.string().c_str(), ios::trunc);
192     if (myfile.is_open()) {
193       for (int i = 0; i < corners.size(); i++) {
194         myfile << corners[i].x << " " << corners[i].y << endl;
195       }
196       myfile.close();
197     }
198   }
199
200 }
201
202
203 /*
204  * If an image has a name like myimage.jpg?fooblah=boo, rename it myimage.jpg
205  */
206 void TrainingImage::
207 remove_query_string()
208 {
209   // If there is a query string after the extension, take it off!
210   string old_name = imgfile.string();
211   size_t found = old_name.find("&");
212   if (found != string::npos) {
213     string new_name = old_name.substr(0, found);
214     fs::rename(old_name, new_name);
215     imgfile = fs::path(new_name);
216   }
217 }
218
219
220
221
222 /*
223  * We need to keep track of the image filename, so we override the open method from ↩
```

```
     Image
 */
void TrainingImage::
open(string _imgfile)
{
  imgfile = fs::path(_imgfile);
  Image::open(_imgfile);
}


/*
 * OpenCV mouse listener
 */
void TrainingImage::
on_mouse(int event, int x, int y, int flags, void* param)
{
  TrainingImage* img = (TrainingImage*)param;
  if (event == CV_EVENT_LBUTTONUP) {
    circle(img->image_w_margin, Point(x,y), 2, Scalar(0,0,255), 2);
    img->corners.push_back(Point(x-MARGIN, y-MARGIN));
  }
}


/*
 * loads the TrainingImage data into the "matcher" and then performs the
 * query against "image"
 */
int TrainingImage::
match(Image image)
{


  Matcher::matcher()->loadTrainingImageData(this);
  return Matcher::matcher()->doQuery(image);

}
```

5.2.10 TrainingSet.h

```
#include <boost/filesystem.hpp>
#include <boost/algorithm/string.hpp>
#include "TrainingImage.h"
#include "ByakuganOps.h"
#include "utils.h"


using namespace std;
```

```
9  using namespace cv;
10 namespace fs = boost::filesystem;
11
12
13 class TrainingSet  {
14 public:
15
16   TrainingSet(){}
17
18   /*
19    * Loads TrainingImages from a previously saved archive in an XML file as
20    * created by the Byakugan scan() function
21    */
22   void loadFromArchive(string path);
23
24   /*
25    * Used by Byakugan's scan() function, it loads any images from
26    * the directory saved into "_path" and makes TrainingImages
27    * out of them
28    */
29   bool loadFromDirectory(string _path);
30
31   /*
32    * Saves the TrainingSet into a propery formatted XML file.
33    * The attributes which are saved are:
34    * the filename, the full path, the computed keypoints and descriptors for
35    * each image
36    */
37   void save();
38
39   /*
40    * Match all images of this TrainingSet against the query image which is
41    * provided in "img". Returns struct which holds the index and the value
42    * (matches_count) for the best matching image
43    */
44   bm match(Image img);
45
46   /*
47    * Sets the Feature Detector
48    */
49   void setFeatureDetector(FeatureDetector* _detector);
50
51   /*
52    * Sets Descriptor Extractor
53    */
54   void setDescriptorExtractor(DescriptorExtractor* _extractor);
55
56   vector<TrainingImage> images;
57
58 protected:
59
```

```
60    static bool is_image(fs::path p)
61    {
62      if(!is_regular_file(p)) {
63        return false;
64      }
65      string ext = extension(p);
66      boost::to_lower(ext);
67      const char *image_exts[] = {".jpg", ".jpeg", ".png", ".tiff", ".tif"};
68      for (int i = 0; i < 5; i++) {
69        if (boost::starts_with(ext, image_exts[i])) {
70          return true;
71        }
72      }
73      return false;
74    }
75
76    /* directory path to load images from */
77    fs::path dir;
78
79    /* Use generic container for detector so we can use different types */
80    Ptr<FeatureDetector> detector;
81    /* Same for extractor */
82    Ptr<DescriptorExtractor> extractor;
83  };
```

5.2.11 TrainingSet.cc

```
1   #include "TrainingSet.h"
2   #include "tinyxml.h"
3   #include "tinystr.h"
4   #include <fstream>
5   #include <iostream>
6   #include <boost/iostreams/filtering_streambuf.hpp>
7   #include <boost/iostreams/copy.hpp>
8   #include <boost/iostreams/filter/gzip.hpp>
9   #include <boost/iostreams/device/file_descriptor.hpp>
10  #include <boost/iostreams/filtering_stream.hpp>
11  #include <boost/iostreams/device/file.hpp>
12
13  using namespace std;
14  using namespace boost::iostreams;
15
16  /*
17   * Loads TrainingImages from a previously saved archive in an XML file as
18   * created by the Byakugan scan() function
19   */
20  void TrainingSet::
21  loadFromArchive(string path)
```

```
22 {
23   const char *name;
24   const char *value;
25   int total;
26   string new_name;
27   bool compressed = false;
28
29   TiXmlNode *node;
30   TiXmlElement *elem, *filename, *keypoints, *filepath;
31
32   new_name = path;
33
34   /* first check if the files are compressed */
35   size_t found = path.find(".gz");
36   if (found != string::npos) {
37
38     compressed = true;
39
40     /* First uncompress the .gz file */
41     ifstream file(path.c_str());
42     filtering_streambuf<input> in;
43     in.push(gzip_decompressor());
44     in.push(file);
45
46     //TODO: find a more efficient way to do this instead of having to write
47     //it to the disk each time
48
49     /* remove .gz extension */
50     new_name = path.substr(0, path.size()-3);
51     ofstream out(new_name.c_str());
52
53     boost::iostreams::copy(in, out);
54
55   }
56
57
58   /* now open the uncompressed .xml file */
59
60   TiXmlDocument doc(new_name.c_str());
61
62   TiXmlHandle hDoc(&doc);
63   TiXmlHandle root(0);
64
65   if (!doc.LoadFile()) {
66     printf("cannot load archive file %s \n", path.c_str());
67     return;
68   }
69
70   images.clear();
71   if (o->verbose)
72     printf("Start loading archive...\n");
```

```
73
74  node = doc.FirstChild("total");
75  elem = node->ToElement();
76
77  /* get <total> number of files */
78  name = elem->Value();
79  value = elem->GetText();
80  total = Strtol(value);
81
82  /* navigate to <objects> */
83  elem = elem->NextSiblingElement();
84
85
86  /* Start of file:
87     <?xml version="1.0" ?>
88     <total>7</total>
89     <objects>
90     <filename>image_0
91     <path>
92     <keypoints>
93     */
94
95  /* keep <filename> reference for later */
96  node = elem->FirstChildElement();
97  filename = node->ToElement();
98
99  string full_p;
100
101 /* loop through images (<filename>) */
102 for (int i = 0; i < total; i++) {
103
104   TrainingImage img;
105
106   if (i != 0) {
107     /* traverse through the next <filename
108      * and update the current filename node */
109     node = filename->NextSiblingElement();
110     filename = node->ToElement();
111   }
112
113   /* get the full file path */
114   node = filename->FirstChildElement();
115   filepath = node->ToElement();
116   value = filepath->GetText();
117
118   if (o->debugging)
119     printf("path: %s\n", value);
120
121   full_p = value;
122   /* read in the image */
123   img.open(full_p);
```

```
124
125     /* traverse through <roi> and get values */
126     node = filepath->NextSiblingElement();
127     elem = node->ToElement();
128     elem->QueryIntAttribute("x", &img.roi.x);
129     elem->QueryIntAttribute("y", &img.roi.y);
130     elem->QueryIntAttribute("width", &img.roi.width);
131     elem->QueryIntAttribute("height", &img.roi.height);
132
133     if (o->debugging)
134       printf("--------roi: %d %d %d %d \n",
135           img.roi.x, img.roi.y, img.roi.width, img.roi.height);
136
137
138     /* keep <keypoints> for future reference */
139     node = elem->NextSiblingElement();
140     keypoints = node->ToElement();
141
142     /* navigate to first <keypoint> */
143     node = keypoints->FirstChildElement();
144     elem = node->ToElement();
145
146     KeyPoint *k;
147     double x, y, size, angle, response;
148
149     /* loop through each <keypoint> for this image */
150     for (elem; elem; elem=elem->NextSiblingElement()) {
151
152       k = new KeyPoint();
153
154       elem->QueryDoubleAttribute("pt.x", &x);
155       elem->QueryDoubleAttribute("pt.y", &y);
156       elem->QueryDoubleAttribute("size", &size);
157       elem->QueryDoubleAttribute("angle", &angle);
158       elem->QueryDoubleAttribute("response", &response);
159       elem->QueryIntAttribute("octave", &k->octave);
160       elem->QueryIntAttribute("class_id", &k->class_id);
161
162       k->pt.x = x;
163       k->pt.y = y;
164       k->size = size;
165       k->angle = angle;
166       k->response = response;
167
168       //printf("val: %.16e %.16e %.16e %.16e %.16e\n",
169       //      k->pt.x, k->pt.y, k->size, k->angle, k->response);
170
171       img.keypoints.push_back(*k);
172     }
173
174     /* node will now point to <rows> which is child of <descriptors> */
```

```
175     node = keypoints->NextSiblingElement()->FirstChildElement();
176     elem = node->ToElement();
177 
178     int rows, cols;
179 
180     rows = Strtol(elem->GetText());
181     /* navigate to <cols> */
182     elem = elem->NextSiblingElement();
183     cols = Strtol(elem->GetText());
184 
185     if (o->debugging)
186       printf("xml descriptor: rows:%d cols:%d \n", rows, cols);
187 
188     /* see opencv/modules/features2d/src/descriptors.cpp:190 */
189     img.descriptors.create(rows, cols, CV_32FC1);
190 
191     /* now point to first descriptor */
192     elem = elem->NextSiblingElement()->NextSiblingElement();
193 
194     MatIterator_<float> it = img.descriptors.begin<float>();
195     MatIterator_<float> it_end = img.descriptors.end<float>();
196     float d_val;
197 
198     /* WARNING: CV_32FC1 means 32bit floating point values
199      * so ensure float is always 32bit */
200 
201     /* Get each <descriptor> value, convert it to float and write it
202      * to the Mat img.descriptors sequentially */
203     for (elem; elem; elem=elem->NextSiblingElement()) {
204 
205       value = elem->GetText();
206       sscanf(value, "%f", &d_val);
207 
208       //printf("%.16e \n", d_val);
209 
210       *it = d_val;
211       it++;
212     }
213 
214 #if 0
215     it = img.descriptors.begin<float>();
216     printf("================ MAT ================== \n");
217     for (; it != it_end; it++) {
218       printf("%.16e\n", *it);
219     }
220     printf("============ END ============= \n");
221 #endif
222 
223     images.push_back(img);
224 
225   }
```

```
226
227   /* if the file was compressed, remove the uncompressed file now */
228   if (compressed)
229     if (remove(new_name.c_str()))
230       printf("Error deleting %s\n", new_name.c_str());
231 }
232
233
234
235 /*
236  * Used by Byakugan's scan() function, it loads any images from
237  * the directory saved into "_path" and makes TrainingImages
238  * out of them
239  */
240 bool TrainingSet::
241 loadFromDirectory(string _path)
242 {
243
244   images.clear();
245
246   dir = fs::path(_path);
247   fs::path full_p;
248   Mat image_roi;
249   Image gray;
250
251   if (!fs::exists(dir) || !fs::is_directory(dir)) {
252     printf("%s is not a directory.\n", dir.c_str());
253     return false;
254   }
255
256   /* Loop through the contents of the directory and assemble
257    * SURF points of any images found */
258   if (o->debugging)
259     printf("Opening %s \n", dir.c_str());
260
261   fs::directory_iterator end_iter;
262   for (fs::directory_iterator dir_itr(dir); dir_itr != end_iter; ++dir_itr) {
263
264     if (o->debugging)
265       printf("encountered a file: %s \n", dir_itr->path().c_str());
266
267     if (is_image(dir_itr->path())) {
268
269       if (o->debugging)
270         printf("found an image %s \n", dir_itr->path().c_str());
271
272       full_p = boost::filesystem3::complete(dir_itr->path().c_str());
273       if (o->debugging)
274         printf("full path: %s\n", full_p.c_str());
275
276       TrainingImage img;
```

```
277       img.open(full_p.string());
278       img.generate_cnrfile();
279
280       if (fs::exists(img.cnrfile)) {
281
282         vector<Point>::iterator it;
283
284         /* use the region of interest as defined by the cnr files */
285         img.roi = img.create_outline();
286         //printf("roi: x: %d y: %d  width: %d height: %d \n",
287         //    roi.x, roi.y, roi.width, roi.height);
288
289         /* convert to grayscale */
290         gray = img.getGrayscale();
291         image_roi = gray.cvImg(img.roi);
292
293         if (o->debugging)
294           printf("Identifying keypoints \n");
295         detector->detect(image_roi, img.keypoints);
296
297         if (o->debugging)
298           printf("Extracting descriptors \n");
299         extractor->compute(image_roi, img.keypoints, img.descriptors);
300
301         if (o->debugging > 5)
302           printf("descriptors cols: %d  rows: %d\n",
303               img.descriptors.cols, img.descriptors.rows);
304
305         images.push_back(img);
306       }
307     }
308   }
309
310   return true;
311 }
312
313
314
315 /*
316  * Saves the TrainingSet into a propery formatted XML file.
317  * The attributes which are saved are:
318  * the filename, the full path, the computed keypoints and descriptors for
319  * each image
320  */
321 void TrainingSet::
322 save()
323 {
324   TiXmlDocument doc;
325   char total[16], filenamebuf[16], rows_buf[16], cols_buf[16], type_buf[2];
326   int rows, cols, width, height;
327   fs::path compressed_file(dir.string()+".xml.gz");
```

```
328
329   /* If we found any images, make a datafile. */
330   if (images.size() <= 0)
331     return;
332
333   /* Construct the path to where the data will be stored */
334   fs::path datafile(dir.string()+".xml");
335
336   if (o->debugging)
337     printf("saving data to %s \n", datafile.c_str());
338
339
340   /* write "<?xml version="1.0" ?>" to file beginning */
341   TiXmlDeclaration *decl = new TiXmlDeclaration("1.0", "", "");
342   doc.LinkEndChild(decl);
343
344   /* write <total>images.size()</total> element */
345   TiXmlElement *element = new TiXmlElement("total");
346   Snprintf(total, sizeof(total), "%d", images.size());
347   TiXmlText *text = new TiXmlText(total);
348   element->LinkEndChild(text);
349   doc.LinkEndChild(element);
350
351   /* write <objects> to root */
352   TiXmlElement *objects = new TiXmlElement("objects");
353   doc.LinkEndChild(objects);
354
355   /* <keypoints> */
356   TiXmlElement *keypoints;
357
358   /* <path> */
359   TiXmlElement *path;
360
361   /* <roi> attributes: x, y, width, height */
362   TiXmlElement *roi;
363
364   for (int i = 0; i < images.size(); i++) {
365
366     TiXmlElement *filename = new TiXmlElement("filename");
367     Snprintf(filenamebuf, sizeof(filenamebuf), "image_%d", i);
368     TiXmlText *text = new TiXmlText(filenamebuf);
369     filename->LinkEndChild(text);
370     objects->LinkEndChild(filename);
371
372     path = new TiXmlElement("path");
373     text = new TiXmlText(images[i].imgfile.c_str());
374     path->LinkEndChild(text);
375     filename->LinkEndChild(path);
376
377     roi = new TiXmlElement("roi");
378     roi->SetAttribute("x", images[i].roi.x);
```

```
379     roi->SetAttribute("y", images[i].roi.y);
380     roi->SetAttribute("width", images[i].roi.width);
381     roi->SetAttribute("height", images[i].roi.height);
382     filename->LinkEndChild(roi);
383 
384     keypoints = new TiXmlElement("keypoints");
385     filename->LinkEndChild(keypoints);
386 
387     KeyPoint *k;
388     TiXmlElement *keypoint;
389 
390     for (int j = 0; j < images[i].keypoints.size(); j++) {
391 
392       k = &(images[i].keypoints[j]);
393 
394       keypoint = new TiXmlElement("keypoint");
395       keypoints->LinkEndChild(keypoint);
396 
397       /*
398        * CV_PROP_RW Point2f pt; //!< coordinates of the keypoints
399        * CV_PROP_RW float size; //!< diameter of the meaningful keypoint neighborhood
400        * CV_PROP_RW float angle; //!< computed orientation of the keypoint (-1 if not ↩
           applicable)
401        * CV_PROP_RW float response; //!< the response by which the most strong keypoints↩
            have been selected.
402        * CV_PROP_RW int octave; //!< octave (pyramid layer) from which the keypoint has ↩
           been extracted
403        * CV_PROP_RW int class_id; //!< object class
404        */
405       keypoint->SetDoubleAttribute("pt.x", k->pt.x);
406       keypoint->SetDoubleAttribute("pt.y", k->pt.y);
407       keypoint->SetDoubleAttribute("size", k->size);
408       keypoint->SetDoubleAttribute("angle", k->angle);
409       keypoint->SetDoubleAttribute("response", k->response);
410       keypoint->SetAttribute("octave", k->octave);
411       keypoint->SetAttribute("class_id", k->class_id);
412 
413     }
414 
415     /* descriptors */
416     TiXmlElement *descriptors = new TiXmlElement("descriptors");
417 
418     Mat *d = &images[i].descriptors;
419 
420     rows = images[i].descriptors.rows;
421     cols = images[i].descriptors.cols;
422 
423     /* rows */
424     TiXmlElement *xml_rows = new TiXmlElement("rows");
425     Snprintf(rows_buf, sizeof(rows_buf), "%d", rows);
426     TiXmlText *r = new TiXmlText(rows_buf);
```

```
427     xml_rows->LinkEndChild(r);
428     descriptors->LinkEndChild(xml_rows);
429
430     /* cols */
431     TiXmlElement *xml_cols = new TiXmlElement("cols");
432     Snprintf(cols_buf, sizeof(cols_buf), "%d", cols);
433     TiXmlText *c = new TiXmlText(cols_buf);
434     xml_cols->LinkEndChild(c);
435     descriptors->LinkEndChild(xml_cols);
436
437     /* data type (dt) */
438     TiXmlElement *dt = new TiXmlElement("dt");
439     TiXmlText *t = new TiXmlText("f"); /* hard-coded data type: f for float */
440     dt->LinkEndChild(t);
441     descriptors->LinkEndChild(dt);
442
443     /* raw data */
444
445     /* we assume Mat is CONT - see
446      * opencv/modules/core/src/persistence.cpp:3440 */
447     int count = rows * cols;
448
449     const char *data = (const char *)(d->data);
450     const char *ptr;
451     char buf[256] = "";
452     int buf_len;
453     char *d_value; /* descriptor value */
454
455     /* see persistence.cpp:2991 -> cvWriteRawData() */
456
457     for (int j = 0; j < count; j++) {
458
459       ptr = FloatToString(buf, *(float *)data);
460       data += sizeof(float);
461       buf_len = (int)strlen(ptr);
462
463       TiXmlElement *xml_desc = new TiXmlElement("descriptor");
464       d_value = strndup(ptr, buf_len);
465       TiXmlText *t = new TiXmlText(d_value);
466       xml_desc->LinkEndChild(t);
467       descriptors->LinkEndChild(xml_desc);
468
469     }
470
471     filename->LinkEndChild(descriptors);
472
473   }
474
475   doc.SaveFile(datafile.string().c_str());
476
477   /* Compress file with gzip */
```

```
478   ifstream file(datafile.string().c_str());
479   char a[256];
480
481   filtering_ostream out;
482   out.push(gzip_compressor());
483   out.push(file_sink(compressed_file.string().c_str()));
484
485   while (!file.eof()) {
486     file.getline(a, sizeof(a));
487     out << a << endl;
488   }
489
490   /* Remove the plain xml file */
491   if (remove(datafile.string().c_str()))
492     printf("Error deleting %s\n", datafile.string().c_str());
493
494 }
495
496
497 /*
498  * Match all images of this TrainingSet against the query image which is
499  * provided in "img". Returns struct which holds the index and the value
500  * (matches_count) for the best matching image
501  */
502 bm TrainingSet::
503 match(Image img)
504 {
505   /* best matching index - the index of the image inside the Training Set
506    * that best matches the query image
507    */
508   vector<int>::size_type bmi;
509
510   /* struct holding index and matches_count of best matching image */
511   bm b;
512
513   /* vector holding the matches count for each image */
514   vector <int> matches;
515
516   for (int i = 0; i < images.size(); i++) {
517
518     /* perform the matching function and push the result - the matches
519      * count - inside the vector for further processing later */
520     matches.push_back(images[i].match(img));
521   }
522
523   vector<int>::iterator result = std::max_element(matches.begin(), matches.end());
524
525   b.matches = *result;
526   b.index = result - matches.begin();
527
528   if (o->debugging)
```

```
529     printf("bmi: %d %s\n", b.index, images[b.index].imgfile.c_str());
530
531   return b;
532 }
533
534 /*
535  * Sets the Feature Detector
536  */
537 void TrainingSet::
538 setFeatureDetector(FeatureDetector* _detector)
539 {
540   detector = _detector;
541   detector.addref();
542 }
543
544 /*
545  * Sets Descriptor Extractor
546  */
547 void TrainingSet::
548 setDescriptorExtractor(DescriptorExtractor* _extractor)
549 {
550   extractor = _extractor;
551   extractor.addref();
552 }
```

5.2.12 Matcher.h

```
1  #pragma once
2  #include <opencv2/opencv.hpp>
3  #include "Image.h"
4  #include "TrainingImage.h"
5  #include "utils.h"
6
7  class TrainingImage;
8
9
10 class Matcher {
11 public:
12
13   static Matcher* matcher();
14
15
16   bool train(Image& train, string datapath);
17   int doQuery(Image& query);
18
19   bool saveTrainingData(string basepath);
20   bool loadTrainingData(string basepath);
21   bool loadTrainingImageData(TrainingImage *img);
```

```
22
23    void drawTrainKeypointsIntoImage(Image& img);
24
25  private:
26
27    Matcher();
28    Matcher(Matcher &) {};
29    Matcher & operator=(Matcher const &){};
30    ~Matcher();
31    static Matcher *matcher_instance; /* instance */
32
33    bool init(string _detector_alg="SURF", string _extractor_alg="SURF",
34        string _matcher_alg="FlannBased", float _ransacReprojThreshold=3.0);
35
36
37    Image trainImg;
38
39    float ransacReprojThreshold;
40
41    bool bTrainAnalyzed;
42    bool bInited;
43    bool bDoCrossCheckFilter;
44
45    Mat train_descriptors, H12;
46    vector<Point2f> train_pts;
47    vector<KeyPoint> train_kpts;
48    vector<DMatch> matches;
49    vector<int> queryIdxs, trainIdxs;
50
51    string detector_alg;    // Fast, Star, Sift, Surf, Mser, Gftt, Harris, GridAdapted, ↩
          PyramidAdapted
52    string extractor_alg;   // Sift, Surf, Brief, OpponentColor
53    string matcher_alg;     // BruteForce, FlannBased
54
55    cv::Ptr<cv::FeatureDetector> detector;
56    cv::Ptr<cv::DescriptorExtractor> extractor;
57    cv::Ptr<cv::DescriptorMatcher> desc_matcher;
58
59    cv::Mat query_descriptors;
60    vector<cv::KeyPoint> query_kpts;
61
62    bool computed_query; /* true after keypoints and descriptors for query image are ↩
          computed for first time */
63
64    Mat correspondenceImg;
65  };
66
67  extern Matcher *matcher;
```

5.2.13 Matcher.cc

```
1  #include "Matcher.h"
2  #include "TrainingImage.h"
3  #include "ByakuganOps.h"
4
5  /* singleton */
6  Matcher *Matcher::matcher_instance = NULL;
7
8  Matcher::
9  Matcher()
10 {
11
12   init();
13
14 }
15
16 Matcher *Matcher::
17 matcher()
18 {
19   if (!matcher_instance)
20     matcher_instance = new Matcher();
21
22   return matcher_instance;
23 }
24
25
26
27
28 bool Matcher::
29 init(string _detector_alg, string _extractor_alg, string _matcher_alg, float ↩
       _ransacReprojThreshold)
30 {
31   ransacReprojThreshold = _ransacReprojThreshold;
32
33   detector_alg= _detector_alg;
34   extractor_alg= _extractor_alg;
35   matcher_alg= _matcher_alg;
36
37   detector = cv::FeatureDetector::create(detector_alg);
38   extractor = cv::DescriptorExtractor::create(extractor_alg);
39   desc_matcher = cv::DescriptorMatcher::create(_matcher_alg);
40
41   if (detector.empty()) {
42     printf("%s is not a known detector algorithm \n", _detector_alg.c_str());
43     return false;
44   }
45
46   if (extractor.empty()) {
```

```
47      printf("%s is not a known extractor algorithm \n", _extractor_alg.c_str());
48      return false;
49    }
50
51    if (desc_matcher.empty()) {
52      printf("%s is not a known matching algorithm \n", _matcher_alg.c_str());
53      return false;
54    }
55
56    bInited = true;
57    computed_query = false;
58
59    return true;
60  }
61
62
63  bool Matcher::
64  train(Image& _trainImg, string datapath)
65  {
66    assert(_trainImg.channels()==1);
67
68    trainImg = _trainImg;
69
70    if (loadTrainingData(datapath)) {
71      bTrainAnalyzed = true;
72    } else {
73      //loadTrainingImageData(trainImg);
74      bTrainAnalyzed = true;
75    }
76
77    return bTrainAnalyzed;
78  }
79
80
81  void Matcher::
82  drawTrainKeypointsIntoImage(Image& img)
83  {
84    if (bTrainAnalyzed && train_kpts.size() > 0) {
85      drawKeypoints(img.cvImg, train_kpts, img.cvImg,
86          Scalar(255, 0, 0), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
87    }
88  }
89
90
91
92  int Matcher::
93  doQuery(Image& queryImg)
94  {
95    if (!bInited) {
96      printf("WARNING: Not inited. Using default values. Please call init before match.\n"↩
           );
```

```
97    init();
98  }
99
100 if (!bTrainAnalyzed) {
101   printf("WARNING: Train image was not set.\n");
102   return -1;
103 }
104
105 if (o->debugging)
106   printf("train img cols %d rows %d \n", trainImg.cvImg.cols, trainImg.cvImg.rows);
107
108 bDoCrossCheckFilter = true;
109
110 if (computed_query == false) {
111   detector->detect(queryImg.cvImg, query_kpts);
112   extractor->compute(queryImg.cvImg, query_kpts, query_descriptors);
113
114   computed_query = true;
115 }
116
117 if (o->debugging) {
118   printf("Keypoints query image: %d\n", query_kpts.size());
119   printf("Keypoints training image: %d\n", train_kpts.size());
120   printf("query descs cols: %d  rows: %d\n", query_descriptors.cols, query_descriptors↩
        .rows);
121   printf("train descs cols: %d  rows: %d\n", train_descriptors.cols, train_descriptors↩
        .rows);
122 }
123
124
125 /* Match the keypoints */
126 if (bDoCrossCheckFilter) {
127   if (o->debugging)
128     printf("cross matching\n");
129   crossCheckMatching(desc_matcher, query_descriptors, train_descriptors, matches, 1 );
130 } else {
131   if (o->debugging)
132     printf("simple matching\n");
133   desc_matcher->match(query_descriptors, train_descriptors, matches );
134 }
135
136 printf("%s ", trainImg.imgfile.string().c_str());
137
138 if (o->verbose)
139   printf("matches: %d\n", matches.size());
140
141 vector<int> queryIdxs(matches.size());
142 vector<int> trainIdxs(matches.size());
143 for (size_t i = 0; i < matches.size(); i++) {
144   queryIdxs[i] = matches[i].queryIdx;
145   trainIdxs[i] = matches[i].trainIdx;
```

```
146   }
147
148
149   /* Calculate the homography */
150   if (ransacReprojThreshold >= 0)
151   {
152     vector<Point2f> points1; KeyPoint::convert(query_kpts, points1, queryIdxs);
153     vector<Point2f> points2; KeyPoint::convert(train_kpts, points2, trainIdxs);
154
155     if (o->debugging > 5) {
156
157       printf("---------- TRAIN KEYPOINTS ----------\n");
158       for (int i = 0; i < train_kpts.size(); i++)
159         printf("x: %f  y: %f  size: %f \n",
160             train_kpts[i].pt.x, train_kpts[i].pt.y, train_kpts[i].size);
161
162       printf("---------- TRAIN POINTS ----------\n");
163       for (int i = 0; i < trainIdxs.size(); i++)
164         printf("%d \n", trainIdxs[i]);
165
166       printf("---------- QUERY KEYPOINTS ----------\n");
167       for (int i = 0; i < query_kpts.size(); i++)
168         printf("x: %f  y: %f  size: %f \n",
169             query_kpts[i].pt.x, query_kpts[i].pt.y, query_kpts[i].size);
170
171       printf("---------- QUERY POINTS ----------\n");
172       for (int i = 0; i < queryIdxs.size(); i++)
173         printf("%d \n", queryIdxs[i]);
174
175       printf("---------- POINTS 1 ----------\n");
176       for (int i = 0; i < points1.size(); i++)
177         printf("%f %f \n", points1[i].x, points1[i].y);
178
179       printf("---------- POINTS 2 ----------\n");
180       for (int i = 0; i < points2.size(); i++)
181         printf("%f %f \n", points2[i].x, points2[i].y);
182
183     }
184
185     H12 = findHomography(Mat(points1), Mat(points2), CV_RANSAC, 3.0);
186   }
187
188   if (o->debugging > 5) {
189
190     printf("----------MATCHES----------\n");
191     for (int i = 0; i < matches.size(); i++) {
192       printf("queryIdx %d trainIdx %d imgIdx %d dist %f \n",
193           matches[i].queryIdx, matches[i].trainIdx, matches[i].imgIdx, matches[i].↩
                distance);
194     }
195     printf("----------END MATCHES----------\n");
```

```
196
197     printf("H12: cols %d rows: %d \n", H12.rows, H12.cols);
198
199     for (int i = 0; i < H12.rows; i++)
200       for (int j = 0; j < H12.cols; j++)
201         printf("%d %d : % f \n", i, j, H12.at<float>(i, j));
202   }
203
204   /* Make a mask of the points in the query image that survive the perspective transform
205    * of the homography
206    */
207   vector<char> matchesMask(matches.size(), 0);
208
209   /* real matches */
210   int matches_count = 0;
211
212   if (!H12.empty()) {
213
214     vector<Point2f> points1; KeyPoint::convert(query_kpts, points1, queryIdxs);
215     vector<Point2f> points2; KeyPoint::convert(train_kpts, points2, trainIdxs);
216     Mat query_points_transformed;
217     perspectiveTransform(Mat(points1), query_points_transformed, H12);
218
219     double maxInlierDist = ransacReprojThreshold < 0 ? 3 : ransacReprojThreshold;
220     for (size_t i1 = 0; i1 < points2.size(); i1++) {
221       if(norm(points2[i1] - query_points_transformed.at<Point2f>((int)i1,0)) <= ↩
              maxInlierDist) { // inlier
222         matchesMask[i1] = 1;
223         matches_count++;
224       }
225     }
226   }
227
228   if (o->debugging > 5) {
229     printf("-------------MASK ------------\n");
230     for (int i = 0; i < matchesMask.size(); i++)
231       printf("%d ", matchesMask[i]);
232     printf("\n------------END MASK ------------\n");
233   }
234
235
236   printf("real_matches:%d\n", matches_count);
237
238   if (o->show_graph) {
239
240     Mat correspondenceImg;
241
242     if (o->debugging)
243       printf("%d %d %d %d \n", trainImg.roi.x, trainImg.roi.y, trainImg.roi.width, ↩
              trainImg.roi.height);
244
```

```
245     Rect l_roi = trainImg.roi;
246     Mat image_roi = trainImg.cvImg(l_roi);
247
248     drawMatches(queryImg.cvImg, query_kpts, image_roi, train_kpts, matches,
249         correspondenceImg, CV_RGB(0, 255, 0), CV_RGB(0, 0, 255), matchesMask);
250     imshow("correspondence", correspondenceImg);
251
252     cvWaitKey();
253   }
254
255   if (o->debugging)
256     printf("------------ END QUERY ------------");
257
258   return matches_count;
259 }
260
261
262
263 bool Matcher::
264 loadTrainingImageData(TrainingImage *img)
265 {
266   bTrainAnalyzed = true;
267
268   // kanonika den theloume na kanoyme copy ena ena ta stoixeia apo to
269   // Training img sta data alla na exoyme ena pointer se auta
270
271   trainImg = img->cvImg;
272   trainImg.roi = img->roi;
273   trainImg.imgfile = img->imgfile;
274   train_kpts = img->keypoints;
275   train_descriptors = img->descriptors;
276
277 }
278
279
280
281 bool Matcher::
282 loadTrainingData(string base)
283 {
284   if (!bInited) {
285     printf("WARNING: Train not inited. Using default values. Please call init before ↩
          load.\n");
286     init();
287   }
288
289   char filename[255];
290   char nodename[255];
291
292   // Open Keypoints file
293   sprintf(filename, "%s_keypoints.xml", base.c_str());
294   FileStorage keypoints_fs;
```

```
295   keypoints_fs.open(filename, FileStorage::READ);
296   if(!keypoints_fs.isOpened())
297     return false;
298
299   // Load keypoints for set keypoint detection method
300   sprintf(nodename, "%s_keypoints", detector_alg.c_str());
301   if (keypoints_fs[nodename].empty())
302     return false;
303   cv::read(keypoints_fs[nodename], train_kpts);
304   keypoints_fs.release();
305
306
307   // Open descriptor file
308   sprintf(filename, "%s_descriptors.xml", base.c_str());
309   FileStorage descriptors_fs;
310   descriptors_fs.open(filename, FileStorage::READ);
311   if (!descriptors_fs.isOpened())
312     return false;
313
314   // Load descriptors for given extractor method
315   sprintf(nodename, "%s_descriptors", extractor_alg.c_str());
316   if (descriptors_fs[nodename].empty())
317     return false;
318   descriptors_fs[nodename] >> train_descriptors;
319   descriptors_fs.release();
320
321
322   bTrainAnalyzed=true;
323   return true;
324 }
325
326
327 bool Matcher::
328 saveTrainingData(string base)
329 {
330   char filename[255];
331   char nodename[255];
332
333   /* Save Keypoints */
334   sprintf(filename, "%s_keypoints.xml", base.c_str());
335   FileStorage keypoints_fs;
336   keypoints_fs.open(filename, FileStorage::APPEND);
337
338   if (!keypoints_fs.isOpened()) {
339     printf("Could not open keypoints file for writing.\n");
340     return false;
341   }
342
343   sprintf(nodename, "%s_keypoints", detector_alg.c_str());
344   cv::write(keypoints_fs, nodename, train_kpts);
345   keypoints_fs.release();
```

```
346
347   /* Save Descriptors */
348   char descriptors_filename[255];
349   sprintf(descriptors_filename, "%s_descriptors.xml", base.c_str());
350   FileStorage descriptors_fs;
351   descriptors_fs.open(descriptors_filename, FileStorage::APPEND);
352
353   if (!descriptors_fs.isOpened()) {
354     printf("Could not open descriptors file for writing.\n");
355     return false;
356   }
357
358   sprintf(nodename, "%s_descriptors", extractor_alg.c_str());
359   descriptors_fs << string(nodename) << train_descriptors;
360   descriptors_fs.release();
361
362   return true;
363 }
```

5.2.14 utils.h

```
1  #include <opencv2/opencv.hpp>
2  #include <string.h>
3  #include <stdarg.h>
4  #include <stdio.h>
5  #include <errno.h>
6
7  #pragma once
8
9  using namespace cv;
10
11 //Takes a descriptor and turns it into an xy point
12 void keypoints2points(const vector<KeyPoint>& in, vector<Point2f>& out);
13
14 //Takes an xy point and appends that to a keypoint structure
15 void points2keypoints(const vector<Point2f>& in, vector<KeyPoint>& out);
16
17 // filters keypoints
18 void crossCheckMatching(cv::Ptr<DescriptorMatcher>& descriptorMatcher,
19             const Mat& descriptors1, const Mat& descriptors2,
20             vector<DMatch>& filteredMatches12, int knn );
21
22 // Uses a homography to warp points
23 vector<Point> warpCorners(double* h, vector<Point> src );
24
25
26 int Snprintf(char *s, size_t n, const char *fmt, ...);
27 int Vsnprintf(char *s, size_t n, const char *fmt, va_list ap);
```

```
28 int Strncpy(char *dest, const char *src, size_t n);
29 void error(const char *fmt, ...) __attribute__ ((format (printf, 1, 2)));
30 void fatal(char *fmt, ...) __attribute__ ((format (printf, 1, 2)));
31
32 typedef union suf
33 {
34   int i;
35   unsigned u;
36   float f;
37 } suf;
38
39 /* best match */
40 typedef struct bm
41 {
42   vector<int>::size_type index;
43   int matches;
44 } bm;
45
46
47 char *FloatToString(char* buf, float value);
48 inline bool cv_isdigit(char c);
49 int Round(float value );
50 unsigned long int Strtoul(const char *nptr, int fatality = 1);
51 unsigned long int Strtol(const char *nptr, int fatality = 1);
```

5.2.15 utils.cc

```
1  #include "utils.h"
2
3
4  //Takes a descriptor and turns it into an xy point
5  void keypoints2points(const vector<KeyPoint>& in, vector<Point2f>& out)
6  {
7    out.clear();
8    out.reserve(in.size());
9    for (size_t i = 0; i < in.size(); ++i)
10   {
11     out.push_back(in[i].pt);
12   }
13 }
14
15 //Takes an xy point and appends that to a keypoint structure
16 void points2keypoints(const vector<Point2f>& in, vector<KeyPoint>& out)
17 {
18   out.clear();
19   out.reserve(in.size());
20   for (size_t i = 0; i < in.size(); ++i) {
21     out.push_back(KeyPoint(in[i], 1));
```

```
22    }
23  }
24
25
26  void crossCheckMatching( cv::Ptr<DescriptorMatcher>& descriptorMatcher,
27              const Mat& descriptors1, const Mat& descriptors2,
28              vector<DMatch>& filteredMatches12, int knn )
29  {
30    filteredMatches12.clear();
31    vector<vector<DMatch> > matches12, matches21;
32    descriptorMatcher->knnMatch(descriptors1, descriptors2, matches12, knn);
33    descriptorMatcher->knnMatch(descriptors2, descriptors1, matches21, knn);
34
35    for (size_t m = 0; m < matches12.size(); m++) {
36
37      bool findCrossCheck = false;
38      for (size_t fk = 0; fk < matches12[m].size(); fk++) {
39        DMatch forward = matches12[m][fk];
40
41        for (size_t bk = 0; bk < matches21[forward.trainIdx].size(); bk++) {
42          DMatch backward = matches21[forward.trainIdx][bk];
43          if (backward.trainIdx == forward.queryIdx) {
44            filteredMatches12.push_back(forward);
45            findCrossCheck = true;
46            break;
47          }
48        }
49        if (findCrossCheck)
50          break;
51      }
52    }
53  }
54
55
56  //-----------------------------------------------------------
57  vector<Point> warpCorners( double* h, vector<Point> src )
58  {
59    vector<Point> dst(4);
60    // from find_object.cpp  There is probably a "better" way to do this.
61    for(int i = 0; i < 4; i++ )
62    {
63      double x = src[i].x, y = src[i].y;
64      double Z = 1./(h[6]*x + h[7]*y + h[8]);
65      double X = (h[0]*x + h[1]*y + h[2])*Z;
66      double Y = (h[3]*x + h[4]*y + h[5])*Z;
67      dst[i] = Point(cvRound(X), cvRound(Y));
68    }
69    return dst;
70  }
71
72  void error(const char *fmt, ...) {
```

```
73    va_list  ap;
74
75    va_start(ap, fmt);
76    fflush(stdout);
77    vfprintf(stderr, fmt, ap);
78    va_end(ap);
79
80    return;
81  }
82
83
84  void fatal(char *fmt, ...)
85  {
86    va_list ap;
87
88    va_start(ap, fmt);
89    fflush(stdout);
90    vfprintf(stderr, fmt, ap);
91    fprintf(stderr, "\nQUITTING!\n");
92    va_end(ap);
93
94    exit(1);
95  }
96
97  int Strncpy(char *dest, const char *src, size_t n) {
98    strncpy(dest, src, n);
99    if (dest[n-1] == '\0')
100     return 0;
101   dest[n-1] = '\0';
102   return -1;
103 }
104
105 int Vsnprintf(char *s, size_t n, const char *fmt, va_list ap)
106 {
107   int ret;
108
109   ret = vsnprintf(s, n, fmt, ap);
110
111   if (ret < 0 || (unsigned) ret >= n)
112     s[n - 1] = '\0';
113
114   return ret;
115 }
116
117 int Snprintf(char *s, size_t n, const char *fmt, ...)
118 {
119   va_list ap;
120   int ret;
121
122   va_start(ap, fmt);
123   ret = Vsnprintf(s, n, fmt, ap);
```

```
124   va_end(ap);
125
126   return ret;
127 }
128
129
130
131 int Round( float value )
132 {
133 #if defined HAVE_LRINT || defined CV_ICC || defined __GNUC__
134   return (int)lrint(value);
135 #else
136   // while this is not IEEE754-compliant rounding, it's usually a good enough ↩
        approximation
137   return (int)(value + (value &gt;= 0 ? 0.5f : -0.5f));
138 #endif
139 }
140
141 inline bool cv_isdigit(char c)
142 {
143     return '0' <= c && c <= '9';
144 }
145
146 char *
147 FloatToString(char* buf, float value)
148 {
149     suf val;
150     unsigned ieee754;
151     val.f = value;
152     ieee754 = val.u;
153
154     if( (ieee754 & 0x7f800000) != 0x7f800000 )
155     {
156         int ivalue = Round(value);
157         if( ivalue == value )
158             sprintf( buf, "%d.", ivalue );
159         else
160         {
161             static const char* fmt = "%.8e";
162             char* ptr = buf;
163             sprintf( buf, fmt, value );
164             if( *ptr == '+' || *ptr == '-' )
165                 ptr++;
166             for( ; cv_isdigit(*ptr); ptr++ )
167                 ;
168             if( *ptr == ',' )
169                 *ptr = '.';
170         }
171     }
172     else
173     {
```

```
174         if( (ieee754 & 0x7fffffff) != 0x7f800000 )
175             strcpy( buf, ".Nan" );
176         else
177             strcpy( buf, (int)ieee754 < 0 ? "-.Inf" : ".Inf" );
178     }
179
180     return buf;
181 }
182
183 /* strtoul with error checking:
184  * fatality should be 0 if error will be handled by caller, or else Strtoul
185  * will exit
186  */
187 unsigned long int
188 Strtoul(const char *nptr, int fatality)
189 {
190   unsigned long value;
191   char *endp = NULL;
192
193   errno = 0;
194   value = strtoul(nptr, &endp, 0);
195   if (errno != 0 || *endp != '\0') {
196     if (fatality)
197       fatal("Invalid value for number: %s", nptr);
198     else {
199       error("Invalid value for number: %s", nptr);
200     }
201   }
202
203   return value;
204 }
205
206 /* strtol with error checking:
207  * fatality should be 0 if error will be handled by caller, or else Strtoul
208  * will exit
209  */
210 unsigned long int
211 Strtol(const char *nptr, int fatality)
212 {
213   unsigned long value;
214   char *endp = NULL;
215
216   errno = 0;
217   value = strtol(nptr, &endp, 0);
218   if (errno != 0 || *endp != '\0') {
219     if (fatality)
220       fatal("Invalid value for number: %s", nptr);
221     else {
222       error("Invalid value for number: %s", nptr);
223     }
224   }
```

```
225
226   return value;
227 }
```

5.2.16 Makefile

```
1  # GNU Make project makefile autogenerated by Premake
2  CC = gcc
3  CXX = g++
4
5  TARGETDIR  = .
6  TARGET     = $(TARGETDIR)/byakugan
7  DEFINES   += -DDEBUG
8  INCLUDES  += -I. -I./tinyxml -I/opt/local/include -I/usr/local/include -I/usr/include
9  CPPFLAGS  += $(DEFINES) $(INCLUDES)
10 CFLAGS    += $(CPPFLAGS) -g -Wno-write-strings
11 CXXFLAGS  += $(CFLAGS)
12 LDFLAGS   += -L/opt/local/lib -L/usr/local/lib -L/usr/lib
13 LIBS      += -lopencv_core -lopencv_highgui -lopencv_imgproc -lopencv_objdetect -↩
       lopencv_ml \
14              -lopencv_video -lopencv_features2d -lopencv_calib3d -lopencv_contrib \
15              -lopencv_legacy -lopencv_flann -lboost_program_options -lboost_system -↩
                    lboost_filesystem \
16              -lboost_iostreams
17
18
19 OBJS := TrainingImage.o utils.o Byakugan.o ByakuganOps.o byakugan.o TrainingSet.o Image.↩
       o Matcher.o \
20   tinyxml.o tinyxmlerror.o tinyxmlparser.o tinystr.o \
21
22 export SRCS = byakugan.cc Byakugan.cc ByakuganOps.cc utils.cc TrainingSet.cc Matcher.cc ↩
       \
23   Image.cc TrainingImage.cc ./tinyxml/tinystr.cpp ./tinyxml/tinyxmlerror.cpp \
24   ./tinyxml/tinyxmlparser.cpp ./tinyxml/tinyxml.cpp
25
26 SHELLTYPE := msdos
27 ifeq (,$(ComSpec)$(COMSPEC))
28   SHELLTYPE := posix
29 endif
30 ifeq (/bin,$(findstring /bin,$(SHELL)))
31   SHELLTYPE := posix
32 endif
33
34 .PHONY: clean
35
36
37 all: $(TARGETDIR) $(TARGET)
38   @:
```

```
39
40 $(TARGET): $(OBJS)
41   @echo Linking byakugan
42   rm -f $@
43   $(CXX) $(LDFLAGS) -o $@ $(OBJS) $(LIBS)
44
45
46 $(TARGETDIR):
47   @echo Creating $(TARGETDIR)
48 ifeq (posix,$(SHELLTYPE))
49   $(SILENT) mkdir -p $(TARGETDIR)
50 else
51   $(SILENT) mkdir $(subst /,\\,$(TARGETDIR))
52 endif
53
54
55 clean:
56   @echo Cleaning byakugan
57   rm -f dependencies.mk makefile.dep
58   rm -f $(OBJS)
59 ifeq (posix,$(SHELLTYPE))
60   $(SILENT) rm -f  $(TARGET)
61 else
62   $(SILENT) if exist $(subst /,\\,$(TARGET)) del $(subst /,\\,$(TARGET))
63 endif
64
65
66 .cc.o :
67   $(CXX) -c $(CXXFLAGS) $< -o $@
68
69 tinystr.o: ./tinyxml/tinystr.cpp
70   @echo $(notdir $<)
71   $(CXX) $(CXXFLAGS) -o "$@" -c "$<"
72 tinyxmlerror.o: ./tinyxml/tinyxmlerror.cpp
73   @echo $(notdir $<)
74   $(CXX) $(CXXFLAGS) -o "$@" -c "$<"
75 tinyxmlparser.o: ./tinyxml/tinyxmlparser.cpp
76   @echo $(notdir $<)
77   $(CXX) $(CXXFLAGS) -o "$@" -c "$<"
78 tinyxml.o: ./tinyxml/tinyxml.cpp
79   @echo $(notdir $<)
80   $(CXX) $(CXXFLAGS) -o "$@" -c "$<"
81
82
83 makefile.dep:
84   $(CXX) -MM $(CXXFLAGS) $(SRCS) > $@
85 include makefile.dep
```

5.3 Byakugan Server

5.3.1 server.py

```
#!/usr/bin/env python


import sys
from subprocess import PIPE, Popen
from threading  import Thread
import shlex

import cgi
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
from SocketServer import ThreadingMixIn
import threading
import time
import tempfile

import re

class MyHandler(BaseHTTPRequestHandler):

  def do_GET(self):
    print "Somebody made a POST request."
    return

  def do_POST(self):
    global rootnode
    try:
      ctype, pdict = cgi.parse_header(self.headers.getheader('content-type'))
      #print ctype
      #print pdict
      if ctype == 'multipart/form-data':
        query = cgi.parse_multipart(self.rfile, pdict)
      elif ctype == 'application/x-www-form-urlencoded':
        length = int(self.headers.getheader('content-length'))
        query = cgi.parse_qs(self.rfile.read(length), keep_blank_values = 1)

      message = threading.currentThread().getName()
      #self.wfile.write(message)
      #self.wfile.write('\n')

     # self.send_response(301)
     # self.send_header("Location", "/")
     # self.end_headers()


```

```
45        upfilecontent = query.get('upfile')
46
47        # Store uploaded file as a temporary file in /tmp
48        # Warning: these files won't be deleted after they are closed
49        f = tempfile.NamedTemporaryFile("wb", delete = False)
50        f.write(upfilecontent[0].decode('base64'))
51        f.close()
52        print("File received. Saved as " + f.name)
53
54        # Now call byakugan
55        cmd = './byakugan -p -i input.txt -q ' + f.name
56        args = shlex.split(cmd)
57
58        p = Popen(args, stdout = PIPE)
59        out, err = p.communicate()
60
61        # parse byakugan's output by searching for the Best match line
62        # and sending the best matching image's filepath as a reply
63        # to the http client
64        for line in iter(out.splitlines()):
65          m = re.search("Best match: ", line)
66          if m:
67            reply = line[:m.start()] + line[m.end():]
68
69        print reply
70
71        self.send_response(200)
72        self.send_header('Content-type', 'text/html')
73        self.send_header('Content-length', len(reply));
74        self.end_headers()
75
76        self.wfile.write(reply);
77
78
79        return
80
81      except Exception, err:
82        print Exception, err
83        return
84
85      pass
86
87
88
89  class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
90    pass
91
92
93
94  def main():
95    try:
```

```
96      server = ThreadedHTTPServer(('', 8000), MyHandler)
97      print 'started httpserver...'
98      server.serve_forever()
99    except KeyboardInterrupt:
100     print '^C received, shutting down server'
101     server.socket.close()
102
103 if __name__ == '__main__':
104   main()
```

5.4 Byakugan Client

5.4.1 MainActivity.java

```
1  package org.ithilgore.byakugan;
2
3
4  import java.io.ByteArrayOutputStream;
5  import java.io.File;
6  import java.io.IOException;
7  import java.io.InputStream;
8  import java.util.ArrayList;
9
10
11 import org.apache.http.protocol.HTTP;
12 import org.apache.http.HttpResponse;
13 import org.apache.http.HttpVersion;
14 import org.apache.http.params.*;
15 import org.apache.http.client.HttpClient;
16 import org.apache.http.client.entity.UrlEncodedFormEntity;
17 import org.apache.http.client.methods.HttpPost;
18 import org.apache.http.conn.ClientConnectionManager;
19 import org.apache.http.conn.scheme.PlainSocketFactory;
20 import org.apache.http.conn.scheme.Scheme;
21 import org.apache.http.conn.scheme.SchemeRegistry;
22 import org.apache.http.conn.ssl.SSLSocketFactory;
23 import org.apache.http.impl.client.DefaultHttpClient;
24 import org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
25 import org.apache.http.message.BasicNameValuePair;
26
27
28 import android.app.Activity;
29 import android.app.AlertDialog;
30 import android.content.DialogInterface;
31 import android.content.Intent;
32 import android.content.SharedPreferences;
```

```
33 import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
34 import android.graphics.Bitmap;
35 import android.graphics.BitmapFactory;
36 import android.net.Uri;
37 import android.os.Bundle;
38 import android.os.Environment;
39 import android.preference.PreferenceManager;
40 import android.provider.MediaStore;
41 import android.view.Menu;
42 import android.view.MenuInflater;
43 import android.view.MenuItem;
44 import android.view.View;
45 import android.view.View.OnClickListener;
46 import android.widget.Button;
47 import android.widget.Toast;
48 import android.os.AsyncTask;
49 import android.widget.ProgressBar;
50 import android.util.Log;
51
52
53
54 public class MainActivity extends Activity implements OnSharedPreferenceChangeListener
55 {
56   private Uri outputFileUri;
57   private InputStream inputStream;
58   public static final int PICTURE_ACTIVITY = 35434;
59   private ProgressBar pbDefaultM;
60   private AlertDialog alertDialog;
61   public String ip_addr;
62   private SharedPreferences prefs;
63
64   // Part of being OnSharedPreferenceChangeListener
65   public synchronized void onSharedPreferenceChanged(SharedPreferences sharedPreferences↩
         , String key) {
66     this.ip_addr = null;
67   }
68
69
70   // Called first time user clicks on the menu button
71   @Override
72   public boolean onCreateOptionsMenu(Menu menu) {
73     MenuInflater inflater = getMenuInflater();
74     inflater.inflate(R.menu.menu, menu);
75     return true;
76   }
77
78   // Called when an options item is clicked
79   @Override
80   public boolean onOptionsItemSelected(MenuItem item) {
81     switch (item.getItemId()) {
82
```

```
83      case R.id.prefs:
84        startActivity(new Intent(this, PrefsActivity.class));
85        break;
86      }
87
88      return true;
89    }
90
91
92    /* Override the onCreate method */
93    @Override
94    public void onCreate(Bundle savedInstanceState)
95    {
96      super.onCreate(savedInstanceState);
97
98        this.prefs = PreferenceManager.getDefaultSharedPreferences(this);
99        this.prefs.registerOnSharedPreferenceChangeListener(this);
100
101     setContentView(R.layout.main);
102     final Button cameraButton = (Button)findViewById(R.id.camera_button);
103
104     pbDefaultM = (ProgressBar) findViewById(R.id.pbDefault);
105
106     alertDialog = new AlertDialog.Builder(this).create();
107
108     cameraButton.setOnClickListener(new OnClickListener() {
109
110       public void onClick(View v) {
111         // Check if there's external storage available and that it's writeable.
112         boolean mExternalStorageAvailable = false;
113         boolean mExternalStorageWriteable = false;
114         String state = Environment.getExternalStorageState();
115
116         if (Environment.MEDIA_MOUNTED.equals(state)) {
117           // We can read and write the media
118           mExternalStorageAvailable = mExternalStorageWriteable = true;
119         } else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
120           // We can only read the media
121           mExternalStorageAvailable = true;
122           mExternalStorageWriteable = false;
123         } else {
124           // Something else is wrong. It may be one of many other states, but all we ↩
                 need
125           // to know is we can neither read nor write
126           mExternalStorageAvailable = mExternalStorageWriteable = false;
127         }
128
129         if (mExternalStorageAvailable && mExternalStorageWriteable) {
130           Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE); // Normally↩
                  you would populate this with your custom intent.
131           File file = new File(Environment.getExternalStorageDirectory(),"imagefile.jpg"↩
```

```
                );
132             outputFileUri = Uri.fromFile(file);
133             cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
134             startActivityForResult(cameraIntent, PICTURE_ACTIVITY);
135           }
136         }
137       });
138   }
139
140   private HttpClient createHttpClient()
141   {
142       HttpParams params = new BasicHttpParams();
143       HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
144       HttpProtocolParams.setContentCharset(params, HTTP.DEFAULT_CONTENT_CHARSET);
145       HttpProtocolParams.setUseExpectContinue(params, true);
146
147       SchemeRegistry schReg = new SchemeRegistry();
148       schReg.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));
149       schReg.register(new Scheme("https", SSLSocketFactory.getSocketFactory(), 443));
150       ClientConnectionManager conMgr = new ThreadSafeClientConnManager(params, schReg);
151
152       return new DefaultHttpClient(conMgr, params);
153   }
154
155
156   class PostToServer extends AsyncTask<String, Integer, String> {
157
158     @Override
159     protected String doInBackground(String... image_str) {
160
161       String the_string_response = "";
162
163       try {
164
165         ArrayList<BasicNameValuePair> nameValuePairs = new ArrayList<BasicNameValuePair↩
              >();
166
167         nameValuePairs.add(new BasicNameValuePair("upfile", image_str[0]));
168
169         HttpClient httpclient = createHttpClient();
170
171         /* get address from user settings, or else place a default value */
172         String URL = prefs.getString("server_ip", "http://192.168.1.3:80");
173         //final String URL = "http://192.168.1.3:80";
174
175
176         HttpPost httppost = new HttpPost(URL);
177
178         httpclient.getParams().setParameter("http.protocol.unambiguous-statusline", ↩
              false);
179
```

```
180         httppost.addHeader("User-Agent", "byakugan client");
181
182         httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
183         HttpResponse response = httpclient.execute(httppost);
184         the_string_response = convertResponseToString(response);
185
186       } catch(Exception e) {
187         Toast.makeText(MainActivity.this, "ERROR " + e.getMessage(), Toast.LENGTH_LONG).↩
              show();
188         System.out.println("Error in http connection " + e.toString());
189         e.printStackTrace();
190       }
191
192       return the_string_response;
193     }
194
195
196     @Override
197     protected void onProgressUpdate(Integer... values) {
198
199       super.onProgressUpdate();
200     }
201
202
203     @Override
204     protected void onPostExecute(String result) {
205
206
207       alertDialog.setTitle("Byakugan Result:");
208       alertDialog.setMessage(result);
209       alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
210          public void onClick(DialogInterface dialog, int which) {
211             // here you can add functions
212          }
213       });
214       alertDialog.show();
215
216       //Toast.makeText(MainActivity.this, "Response " + result, Toast.LENGTH_LONG).show↩
              ();
217     }
218   }
219
220
221   public class ProgressBarAsyncTask extends AsyncTask<Integer, Integer, Boolean> {
222
223     private static final String LOG_TAG = "PB_ASYNC_TASK";
224     private ProgressBar pbM;
225     int teSecondsProgressedM;
226
227     /**
228      * The parameters in the constructor of the class are the controls from the
```

```
229      * main activity that we will update as the background work is progressing.
230      *
231      * @param pb: the progress bar control.
232      * @param secondProgressed: an edit text with the percentage of seconds
233      *                          progressed.
234      */
235     public ProgressBarAsyncTask(ProgressBar pb, int secondProgressed) {
236       Log.d(LOG_TAG, "Constructor");
237
238       pbM = pb;
239       teSecondsProgressedM = secondProgressed;
240     }
241
242     /**
243      * This method will be called before the execution of the task. Here we
244      * are activating the visibility of the progress controls of the main
245      * activity.
246      */
247     @Override
248     protected void onPreExecute() {
249       Log.d(LOG_TAG, "Pre-Execute");
250
251       super.onPreExecute();
252       pbM.setVisibility( View.VISIBLE);
253       //teSecondsProgressedM.setVisibility( View.VISIBLE);
254     }
255
256     /**
257      * This method will be called after the invocation of the
258      * publishProgress( progress) method in the background task. Here is where
259      * we update the progress controls.
260      *
261      * @param progress: the amount of progress of the background task
262      */
263     @Override
264     protected void onProgressUpdate(Integer... progress) {
265       Log.d(LOG_TAG, "Progress Update: " + progress[0].toString());
266
267       super.onProgressUpdate( progress[0]);
268       pbM.setProgress( progress[0]);
269       //teSecondsProgressedM.setText( progress[0].toString());
270     }
271
272     /**
273      * This method is called after the execution of the background task. Here
274      * we reset the progress controls and set their visible property to
275      * invisible again, to hide them.
276      *
277      * @param result: is the result of the background task, and it is passed to
278      *                this method with the result returned in the
279      *                doInBackGroundMethod()
```

```
280       */
281      @Override
282      protected void onPostExecute(Boolean result) {
283        Log.d(LOG_TAG, "Post-Execute: " + result);
284
285        super.onPostExecute(result);
286        pbM.setVisibility(View.INVISIBLE);
287        //teSecondsProgressedM.setVisibility( View.INVISIBLE);
288        //teSecondsProgressedM.setText("");
289        pbM.setProgress(0);
290      }
291
292      /**
293       * This method is called for executing the background task in the AsyncTask.
294       * For this tutorial we are only sleeping the thread for the number of
295       * seconds passed as parameter of the function.
296       *
297       * @param numSeconds: life of the task
298       * @return the result of the background task
299       */
300      @Override
301      protected Boolean doInBackground(Integer... numSeconds) {
302        Log.d(LOG_TAG, "doInBackground: " + numSeconds[0]);
303
304        try {
305          int totalSecs = numSeconds[0].intValue();
306          Log.d(LOG_TAG, "Total SECS: " + totalSecs);
307
308          for (int i = 1; i <= totalSecs; i++) {
309            Log.d(LOG_TAG, "Sleeping " + i);
310            Thread.sleep(1000);
311
312            float percentage = ((float)i / (float)totalSecs) * 100;
313            Log.d(LOG_TAG, "Percentage of progress: " + percentage);
314
315            publishProgress( new Float( percentage).intValue());
316          }
317        } catch (InterruptedException e) {
318          e.printStackTrace();
319          return false;
320        }
321
322        return true;
323      }
324    }
325
326
327    @Override
328    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
329      super.onActivityResult(requestCode, resultCode, intent);
330
```

```
331     if (requestCode == PICTURE_ACTIVITY && resultCode == Activity.RESULT_OK) {
332
333       File file = new File(Environment.getExternalStorageDirectory(),"imagefile.jpg");
334       if (file.exists()) {
335         ByteArrayOutputStream stream = new ByteArrayOutputStream();
336
337         // Enclose this in a scope so that when it's over we can call the garbage ↩
                collector (the phone doesn't have a lot of memory!)
338         {
339           Bitmap image = BitmapFactory.decodeFile(file.getPath());
340           Bitmap scaled = Bitmap.createScaledBitmap(image, (int)(image.getWidth() * 0.3)↩
                , (int)(image.getHeight() * 0.3), false);
341           scaled.compress(Bitmap.CompressFormat.JPEG, 90, stream);
342           image.recycle();
343           scaled.recycle();
344         }
345         System.gc();
346
347         byte [] byte_arr = stream.toByteArray();
348         String image_str = Base64.encodeBytes(byte_arr);
349
350         int seconds = 35;
351         ProgressBarAsyncTask pbTask = new ProgressBarAsyncTask(pbDefaultM, seconds);
352         pbTask.execute(seconds);
353
354         new PostToServer().execute(image_str);
355
356       }
357     }
358   }
359
360
361
362   public String convertResponseToString(HttpResponse response) throws ↩
        IllegalStateException, IOException {
363
364     String res = "";
365     StringBuffer buffer = new StringBuffer();
366     inputStream = response.getEntity().getContent();
367     int contentLength = (int) response.getEntity().getContentLength(); //getting content↩
          length...
368     //Toast.makeText(MainActivity.this, "contentLength : " + contentLength, Toast.↩
          LENGTH_LONG).show();
369
370
371     if (contentLength < 0) {
372
373     } else {
374       byte[] data = new byte[512];
375       int len = 0;
376       try {
```

```
377         while (-1 != (len = inputStream.read(data)) ) {
378           buffer.append(new String(data, 0, len)); //converting to string and appending ↩
                  to stringbuffer
379         }
380       } catch (IOException e) {
381         e.printStackTrace();
382       }
383
384       try {
385         inputStream.close(); // closing the stream
386       } catch (IOException e) {
387         e.printStackTrace();
388       }
389
390       res = buffer.toString(); // converting stringbuffer to string...
391       //Toast.makeText(MainActivity.this, "Result : " + res, Toast.LENGTH_LONG).show();
392       //System.out.println("Response => " + EntityUtils.toString(response.getEntity()));
393
394     }
395
396     return res;
397   }
398
399 }
```

5.4.2 PrefsActivity.java

```
1  package org.ithilgore.byakugan;
2
3  import android.os.Bundle;
4  import android.preference.PreferenceActivity;
5
6  public class PrefsActivity extends PreferenceActivity {
7
8    @Override
9    protected void onCreate(Bundle savedInstanceState) {
10     super.onCreate(savedInstanceState);
11     addPreferencesFromResource(R.xml.prefs);
12   }
13
14 }
```

Αναφορές

[1] M. Brown and D. Lowe. Invariant features from interest point groups. 2002.

[2] T. Tuytelaars H. Bay and L. Van Gool. Surf: Speeded up robust features. 2008.

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proceedings of the Alvey Vision Conference*, pages 147 – 151, 1998.

[4] Tinne Tuytelaars Herbert Bay, Andreas Ess and Luc Van Gool. Speeded-up robust features (surf) (revised). 2008.

[5] J.J. Koenderink. The structure of images. page 50:363 { 370, 1984.

[6] T. Lindeberg. Scale-space for discrete signals. page 12(3):234{254, 1990.

[7] T. Lindeberg. Feature detection with automatic scale selection. pages 30(2):79 – 116, 1998.

[8] D. Lowe. Object recognition from local scale-invariant features. 1999.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints, cascade filtering approach. page 60(2):91 { 110, 2004.

[10] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. pages 525 – 531, 2001.

[11] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. pages 257 – 263, 2003.

[12] Marius Muja and David G. Lowe. Flann - fast library for approximate nearest neighbors.

[13] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. 2006.

[14] P.A. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. pages 511 – 518, 2001.