ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εξαγωγή χαρακτηριστικών από εικόνες για αναζήτηση και αρχειοθέτηση

Image Feature extraction for searching and archiving

Διπλωματική Εργασία

Ευάγγελος Ι. Μητσιάνης

Επιβλέποντες Καθηγητές
Χρήστος Σωτηρίου
Αναπληρωτής Καθηγητής

Γεράσιμος Ποταμιάνος
Αναπληρωτής Καθηγητής

Επόπτης Καθηγητής
Ιωάννης Κατσαβουνίδης

Βόλος
2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Εξαγωγή χαρακτηριστικών από εικόνες για αναζήτηση και αρχειοθέτηση

Διπλωματική Εργασία

Ευάγγελος Ι. Μητσιάνης

Επιβλέποντες
Χρήστος Σωτηρίου
Αναπληρωτής Καθηγητής

Γεράσιμος Ποταμιάνος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή την 5η Μαρτίου 2015

.....	.....
Χ. Σωτηρίου	Γ. Ποταμιάνος
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής

Κύριος επιβλέπων της παρούσας Διπλωματικής εργασίας ήταν ο καθηγητής κύριος Ιωάννης Κατσαβουνίδης. Επειδή ο κύριος Κατσαβουνίδης είναι σε άδεια άνευ αποδοχών, για την ολοκλήρωση της εργασίας έγινε αλλαγή επιβλέποντα και στην επιτροπή εισήλθε ο καθηγητής κύριος Χρήστος Σωτηρίου.

Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

……..

Μητσιάνης Ευάγγελος
Διπλωματούχος Μηχανικός Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Στην οικογένεια μου

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή μου εκτίμηση στον καθηγητή κύριο Κατσαβουνίδη Ιωάννη για την εμπιστοσύνη που μου έδειξε, την καθοδήγησή του και τις συμβουλές του κατά τη διάρκεια της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κύριο Σωτηρίου Χρήστο για τις πολύτιμες συμβουλές που μου έδωσε κατά τη συγγραφή της παρούσας εργασίας. Ακόμη, ευχαριστώ τον κύριο Ποταμιάνο Γεράσιμο που δέχτηκε να είναι μέλος της εξεταστικής επιτροπής.

Θα ήθελα να εκφράσω ένα μεγάλο ευχαριστώ στον Στέργιο Πουλαράκη για την αμέριστη βοήθειά του, τις συμβουλές του και την υποστήριξή του καθ' όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Τέλος, ιδιαίτερες ευχαριστίες θέλω να εκφράσω στην οικογένειά μου για την υποστήριξή τους, την εμπιστοσύνη τους και την αγάπη τους κατά τη διάρκεια των σπουδών μου.

Μητσιάνης Ευάγγελος

Βόλος, 2015

Περίληψη

Ο αυξανόμενος αριθμός αποθηκευμένων ψηφιακών εικόνων σε μεγάλες βάσεις δεδομένων έχει καταστήσει αναγκαία την αποδοτική αναζήτηση και ανάκτηση τους. Τα συστήματα ανάκτησης εικόνας βρίσκουν εφαρμογές σε τομείς όπως αναγνώριση αντικειμένων, σχεδιασμό εγκαταστάσεων, συστήματα γεωγραφικών πληροφοριών και τηλεπισκόπησης, εξιχνιάσεις εγκλημάτων, ιατρικές διαγνώσεις και αρχικοποίηση φωτογραφιών. Τα βήματα που ακολουθούνται για γίνει η ανάκτηση μιας εικόνας είναι η ανίχνευση και η εξαγωγή των χαρακτηριστικών της, η αντιστοίχιση των χαρακτηριστικών με αυτά των εικόνων της βάσης δεδομένων και τέλος, η αξιοποίηση των αποτελεσμάτων για να γίνει η ανάκτηση. Στην παρούσα εργασία για την ανίχνευση και εξαγωγή χαρακτηριστικών των εικόνων χρησιμοποιείται ο αλγόριθμος SIFT[1]. Η αντιστοίχιση των χαρακτηριστικών πραγματοποιείται με έναν αλγόριθμο Fast NN[2,3]. Στην συνέχεια τα αποτελέσματα αξιοποιούνται βάση ορισμένων χαρακτηριστικών που προτείνουμε, για να γίνει η ανάκτηση της σωστής εικόνας. Για τα πειράματα χρησιμοποιήθηκαν εικόνες από την βάση δεδομένων “The PASCAL Visual Object Classes”[12].

Abstract

The increasing number of stored digital images in large databases has necessitated the efficient search and retrieval of the images. Image retrieval systems have applications in areas such as object recognition, Architectural and engineering design, geographic information systems and remote sensing, solving crimes, medical diagnoses and photo archives. Image retrieval systems are divided into three modules: features detection and extraction, matching each feature of the image in a large database of features extracted for the training images and finally the results are evaluate to retrieve the correct image. In this work we used SIFT[1] algorithm to achieve feature detection and extraction. Matching of the features performed with a tree-base Fast NN[2,3] algorithm. Then we proposed a method to evaluate matching results and retrieve the correct image. For the experiments we use images from the database "The PASCAL Visual Object Classes" [12].

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή του προβλήματος

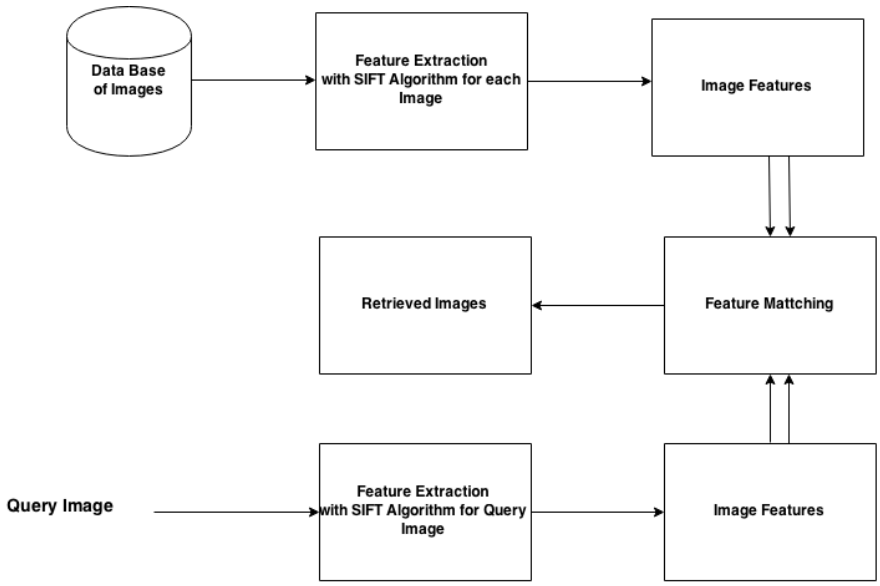
Η ανακάλυψη των ψηφιακών φωτογραφικών μηχανών, οδήγησε στη δημιουργία μεγάλων συλλογών από φωτογραφίες που αποθηκεύονται σε ψηφιακή μορφή. Η ανάγκη για αποθήκευση και ανάκτηση εικόνων με αποδοτικό τρόπο αυξάνεται όσο οι βάσεις δεδομένων μεγαλώνουν σε μέγεθος. Το πεδίο ανάκτησης μίας εικόνας έχει χωριστεί στις περιοχές έρευνας της υπολογιστικής όρασης και της επεξεργασίας εικόνας. Ένα σύστημα ανάκτησης εικόνας (*image retrieval system*) είναι ένα υπολογιστικό σύστημα που αναζητεί και ανακτά εικόνες από μεγάλες βάσεις δεδομένων. Κατά τη διαδικασία ανάκτησης η εικόνα προς αναζήτηση (*query image*) συγκρίνεται με άλλες, έτσι ώστε να βρεθούν αυτές που της μοιάζουν περισσότερο. Θεωρητικά, ένας τρόπος σύγκρισης δύο εικόνων, θα μπορούσε να είναι η διαφορά των pixels που τις αποτελούν. Στον πραγματικό κόσμο όμως, η πιθανότητα δύο εικόνες να είναι ίδιες, *pixel* προς *pixel*, είναι σχεδόν μηδενική. Ένα βασικό πρόβλημα κατά την ανάκτηση μίας εικόνας, είναι η διαφορά ανάλυσης που έχουν οι *query* εικόνες από αυτές που είναι στη βάση δεδομένων. Έτσι, η σύγκριση δεν μπορεί να γίνει με αυτό τον τρόπο. Ένας άλλος λόγος είναι ότι ο χρόνος θα ήταν O(N) για N pixel ανά εικόνα, καθώς για τη σύγκριση δύο και μόνο εικόνων θα έπρεπε να γίνουν N συγκρίσεις pixel (τυπική εικόνα 500x500, 250K *pixels*). Έτσι, το πρώτο πρόβλημα που δημιουργείται είναι ο τρόπος που θα γίνει η αναπαράσταση της εικόνας, με σκοπό να γίνει η εύρεσή της και να μειωθεί ο χρόνος αναζήτησης και ανάκτησης.

Η λύση δόθηκε από αλγορίθμους της υπολογιστικής όρασης, οι οποίοι μετασχηματίζουν την εικόνα σε ένα σύνολο *keywords*. Ένα *keyword*, αναπαριστά ένα ιδιαίτερο χαρακτηριστικό γνώρισμα μίας εικόνας, όπως μία απότομη αλλαγή της φωτεινότητας των *pixels*, ένα σημείο ή μία ολόκληρη περιοχή από *pixels*. Σχεδόν σε όλες τις περιπτώσεις ένα *keyword* αντιπροσωπεύεται από ένα διάνυσμα αριθμών (ακεραίων ή δεκαδικών) και κατ' επέκταση μία εικόνα αντιπροσωπεύεται από ένα σύνολο διανυσμάτων. Η αναπαράσταση της εικόνας με αυτό τον τρόπο λύνει τα προβλήματα διαφορετικής ανάλυσης και ανομοιότητας των *pixels*, καθιστώντας δυνατή την αναζήτηση και την εύρεση των *query* εικόνων. Επίσης, μειώνει σε μεγάλο βαθμό το υπολογιστικό που χρειάζεται για να γίνει μία αναζήτηση και ανάκτηση.

Το επόμενο στάδιο, για να μπορέσουμε να ανακτήσουμε μία εικόνα, είναι η αντιστοίχιση των χαρακτηριστικών διανυσμάτων που την αντιπροσωπεύουν με αυτά των εικόνων της βάσης δεδομένων. Αυτό γίνεται υπολογίζοντας τις αποστάσεις αυτών των

διανυσμάτων και επιστρέφοντας τα διανύσματα στη βάση δεδομένων, τα οποία βρίσκει πιο κοντά σε αυτά προς αναζήτηση. Το πρόβλημα ανάγεται ως πρόβλημα Κοντινότερου Γείτονα (*Nearest Neighbor, NN*). Καθώς το μέγεθος των βάσεων δεδομένων αυξάνεται, η αναζήτηση και η αντιστοίχιση μεγάλων διαστάσεων διανυσμάτων είναι μια κρίσιμη διαδικασία. Έτσι, εμφανίζεται ένα δεύτερο πρόβλημα κατά την αναζήτηση του NN, η οποία θέλουμε να είναι μία αποδοτική διαδικασία. Έχουν προταθεί μέθοδοι που χρησιμοποιούν πίνακες κατακερματισμού (*Hash Tables)* για την αναζήτηση και την ανάκτηση του NN[6, 7], αλλά λύνουν το πρόβλημα μόνο για μικρές διαστάσεις διανυσμάτων. Η χρήση *Hash Table* είναι μη αποδοτική για διανύσματα μεγάλων διαστάσεων και αυτό έγκειται στο γεγονός ότι οι θέσεις αποθήκευσης των διανυσμάτων (*bins*) έχουν συγκεκριμένες διαστάσεις. Λόγω των διαστάσεων των θέσεων αποθήκευσης, χρειάζεται πολλές φορές να γίνει μία γραμμική αναζήτηση μέσα από πολλά διανύσματα για να βρεθεί ο NN και έτσι δε βρίσκουν καλή εφαρμογή. Η λύση στο πρόβλημα δόθηκε από τα kd-trees[8, 10], που παραλλαγές τους χρησιμοποιούνται ευρέως σε εφαρμογές αναζήτησης NN. Σε υψηλό επίπεδο, ένα kd-tree είναι η γενίκευση ενός δυαδικού δέντρου αναζήτησης που αποθηκεύει σημεία στον k-διάστατο χώρο. Στο [1] γίνεται ο ισχυρισμός ότι οι «καλύτεροι» αλγόριθμοι NN, όπως kd-tree αλγόριθμοι, για διανύσματα άνω των 10-διαστάσεων δεν παρουσιάζουν επιτάχυνση σε σχέση με τον αλγόριθμο που ελέγχει όλα τα διανύσματα για να βρει τον NN (*Full Nearest-Neighbor*). Έτσι, προτάθηκε ο αλγόριθμος BBF (*Best Bin First*)[5, 6] ο οποίος για λόγους απόδοσης δίνει μία προσεγγιστική λύση στο πρόβλημα του NN. Ο BBF είναι μία μετατροπή του kd-tree αλγορίθμου και η προσέγγιση βασίζεται στην ιδέα της εύρεσης του NN θέτοντας ένα άνω όριο στον αριθμό των διανυσμάτων που εξετάζονται. Κατά την εκτέλεση της αναζήτησης διατηρείται μια ουρά προτεραιοτήτων με τις αποστάσεις των διανυσμάτων τα οποία δεν έχουν ακόμα εξεταστεί. Με τον παραπάνω τρόπο ο BBF καταφέρνει να περιορίσει την αναζήτηση στα διανύσματα που είναι πιο κοντά στο εξεταζόμενο, βασιζόμενος στο ότι αυτά που είναι πιο κοντά έχουν μεγαλύτερη πιθανότητα να είναι ο NN. Στην παρούσα εργασία προτείνονται τρεις τρόποι αναζήτησης NN, με τον αλγόριθμο Fast NN [2, 3]. Οι μέθοδοι Depth Only Search και Limited Search δίνουν προσεγγιστικά τον NN ενώ η Complete Search δίνει τον ακριβή. Στο [2] αναφέρεται ότι ο Fast NN δημιουργήθηκε για να λύσει το πρόβλημα του κοντινότερου γείτονα κατά την κωδικοποίηση για διανύσματα k-διαστάσεων. Στο [3] χρησιμοποιήθηκε για να λύσει το ίδιο πρόβλημα, αλλά στην αναγνώριση χειρονομιών (*gesture recognition*).

Μετά το τέλος της αντιστοίχισης NN, τα αποτελέσματα αξιοποιούνται έτσι ώστε να γίνει ανάκτηση των εικόνων από τη βάση δεδομένων. Στο Σχήμα 1.1 που ακολουθεί φαίνεται το γενικό διάγραμμα ενός συστήματος ανάκτησης εικόνας.



Σχήμα 1.1 Διάγραμμα ανάκτησης εικόνας

1.2 Στόχοι της Παρούσας Εργασίας

Στόχος της παρούσας εργασίας είναι η δημιουργία μίας μηχανής αναζήτησης και ανάκτησης εικόνων από μία βάση δεδομένων. Όπως φαίνεται στο Σχήμα 1.1, η διαδικασία μπορεί να χωριστεί σε τέσσερα στάδια: την εξαγωγή των χαρακτηριστικών, την αντιστοίχιση των χαρακτηριστικών, τη δημιουργία ευρετηρίου για τη βάση δεδομένων και την αξιολόγηση των αποτελεσμάτων για σωστή ανάκτηση.

Στόχος, για να κάνουμε την εξαγωγή των χαρακτηριστικών, ήταν η υλοποίηση του αλγόριθμου SIFT[1] σε γλώσσα προγραμματισμού C. Ο SIFT παράγει έναν αριθμό από χαρακτηριστικά *keywords,* τα οποία αντιπροσωπεύονται από διανύσματα 128-διαστάσεων.

Το επόμενο στάδιο για την ανάκτηση μιας εικόνας, είναι η χρήση ενός αλγορίθμου για να γίνει η αντιστοίχιση των χαρακτηριστικών διανυσμάτων της εικόνας προς αναζήτηση με αυτά των εικόνων της βάσης δεδομένων. Για αυτό το στάδιο προτείνουμε τρεις μεθόδους αναζήτησης με τον αλγόριθμο Fast NN [2, 3]. Ο αλγόριθμος εφαρμόζεται για πρώτη φορά στο πεδίο της ανάκτησης εικόνας, για να επιλυθεί το πρόβλημα του NN. Επίσης, σε προηγούμενες μελέτες δεν έχει εξεταστεί η συμπεριφορά του αλγορίθμου για διανύσματα άνω των 64-διαστάσεων. Όπως αναφέρεται παραπάνω, τα SIFT χαρακτηριστικά δίνουν διανύσματα 128-διαστάσεων γεγονός που είναι μια πρόκληση για την απόδοση του Fast NN [2, 3].

Στη συνέχεια προτείνουμε τον τρόπο δημιουργίας ενός ευρετηρίου για τις εικόνες της βάσης δεδομένων, ώστε να γίνει σε επόμενο στάδιο η επεξεργασία των αποτελεσμάτων του ευρετηρίου.

Τέλος, αναφέρουμε ένα μέτρο επεξεργασίας και αξιολόγησης των αποτελεσμάτων της αναζήτησης NN για να γίνει η ανάκτηση της εικόνας ή των εικόνων από τη βάση δεδομένων.

1.3 Διάρθρωση της Διπλωματικής

Η παρούσα εργασία χωρίζεται σε επτά κεφάλαια. Στο Κεφάλαιο 2 παρουσιάζεται η αναπαράσταση μίας εικόνας στον ηλεκτρονικό υπολογιστή και γίνονται αναφορές για τη χρησιμότητά της στην καθημερινή ζωή. Στη συνέχεια, γίνεται μία επισκόπηση του τομέα που σχετίζεται με την ανίχνευση και την εξαγωγή χαρακτηριστικών από εικόνες, παρουσιάζοντας τις μεθόδους που χρησιμοποιούνται. Τέλος, ορίζεται το πρόβλημα του Κοντινότερου Γείτονα καθώς και τρόποι που μας βοηθούν στην αύξηση της απόδοσης, ώστε να αποφύγουμε την αναζήτηση και τη σύγκριση όλων των δεδομένων. Στο Κεφάλαιο 3 εξηγείται ο αλγόριθμος SIFT. Ακόμη, γίνεται σύγκριση του αλγορίθμου SIFT με άλλους αλγόριθμους εξαγωγής χαρακτηριστικών. Στο Κεφάλαιο 4 παρουσιάζεται ο αλγόριθμος Fast NN και η χρήση του στην αναζήτηση Κοντινότερου Γείτονα. Ακολουθεί το Κεφάλαιο 5, όπου εξηγούνται τα βήματα που ακολουθήσαμε για να γίνει η ανάκτηση της κατάλληλης εικόνας. Επίσης, αναφέρεται μία μέθοδος αρχειοθέτησης των εικόνων προς αναζήτηση. Στο Κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήσαμε. Τέλος, στο Κεφάλαιο 7 αναφέρονται συμπεράσματα και προτάσεις για μελλοντική επέκταση της εργασίας.

Κεφάλαιο 2

Βιβλιογραφική Ανασκόπηση

2.1 Ψηφιακή Εικόνα

Η εικόνα σήμερα αποτελεί μία από τις σημαντικότερες πηγές πληροφορίας. Τη συναντάμε ως ακίνητη με τη μορφή φωτογραφίας ή κινούμενη ως video. Αυτή μπορεί να είναι ασπρόμαυρη, έγχρωμη ή σε αποχρώσεις του γκρι. Ο όρος επίπεδο του γκρι (*gray level*), συχνά χρησιμοποιείται για μονοχρωματικές εικόνες. Η ψηφιακή εικόνα αποτελεί ό, τι πιο σύγχρονο σε πολλούς τομείς της σημερινής κοινωνίας. Είναι πηγή πληροφοριών στο χώρο της ενημέρωσης (*Internet, multimedia*), της εκπαίδευσης και της υγείας (*medical images*). Επίσης, έχει μεγάλη συνεισφορά στο χώρο του θεάματος και της ψυχαγωγίας (ψηφιακή τηλεόραση, DVD κλπ.).

2.1.1 Αναπαράσταση εικόνας στο Ηλεκτρονικό Υπολογιστή

Η εικόνα μπορεί να οριστεί ως μια δισδιάστατη συνάρτηση f(x,y), όπου τα x, y είναι οι χωρικές συντεταγμένες (*συντεταγμένες επιπέδου*) και η τιμή της συνάρτησης f σε κάθε ζευγάρι συντεταγμένων x, y ονομάζεται ένταση της εικόνας στο σημείο αυτό. Επίσης, η εικόνα μπορεί να θεωρηθεί πως είναι η κατανομή της πληροφορίας στο επίπεδο (x,y). Έτσι, η συνάρτηση f περιγράφει μια επιφάνεια η οποία έχει μεγάλη τιμή σε περιοχές που η εικόνα είναι πιο λευκή, ενώ για μικρές τιμές της f η εικόνα πλησιάζει το μαύρο.

Πρακτικά, κάθε εικόνα για να υποστεί ψηφιακή επεξεργασία θα πρέπει αρχικά να μετατραπεί σε ψηφιακή. Έτσι, θα πρέπει να λάβουμε τιμές της συνάρτησης f(x,y) σε ισαπέχουσες θέσεις x και y. Η πυκνότητα με την οποία θα ληφθούν τα δείγματα καθορίζεται από το θεώρημα δειγματοληψίας.

Θεώρημα δειγματοληψίας [17]

Η απόσταση δύο διαδοχικών δειγμάτων στο επίπεδο (x,y) θα πρέπει να είναι μικρότερη από την ημιπερίοδο των ταχύτερων εναλλαγών της συνάρτησης f (x, y). Με άλλα λόγια, θα πρέπει να δειγματοληπτούμε αρκετά γρήγορα ώστε να προλαβαίνουμε τις γρήγορες εναλλαγές επιπέδου φωτεινότητας της εικόνας. Στη συνέχεια, τα δείγματά μας θα πρέπει να κβαντιστούν σε πεπερασμένο αριθμό σταθμών. Συνήθως οι στάθμες που χρησιμοποιούνται είναι 256 (0…255). Στη στάθμη 255 αντιστοιχούμε το λευκό ενώ στη στάθμη 0 το μαύρο. Έτσι, ένας δυαδικός αριθμός των 8 bits (*1 byte*) επαρκεί για να

περιγράψουμε την τιμή της φωτεινότητας ενός δείγματος της εικόνας, που ονομάζεται εικονοστοιχείο (*picture element-pixel*).

(α) (β)

Εικόνα 2.1 (α),(β) Στην εικόνα (α) φαίνεται μία έγχρωμη αναπαράσταση και στην εικόνα (β) μία αναπαράσταση σε αποχρώσεις του γκρι.

2.1.2 Ψηφιακή Επεξεργασία Εικόνας

Η Ψηφιακή Επεξεργασία Εικόνων (*ΨΕΕ*) είναι η εφαρμογή της ψηφιακής επεξεργασίας σημάτων πάνω σε συγκεκριμένα σήματα (*εικόνες*). Η επεξεργασία, η μετάδοση και η κατανόηση των εικόνων αποτελούν πεδία συνεχώς αναπτυσσόμενης έρευνας. Το μέγεθος μιας εικόνας απαιτεί τεράστια ταχύτητα υλοποίησης των αλγορίθμων για λειτουργία σε πραγματικό χρόνο. Η τεχνολογία ολοκληρωμένων κυκλωμάτων πολύ μεγάλης κλίμακας (*VLSI*), σε συνδυασμό με την ανάπτυξη αρχιτεκτονικών συνεχούς ροής (*pipelining*) με μεγάλο βαθμό παραλληλισμού, έδωσε τη δυνατότητα υλοποίησης πολλών πολύπλοκων αλγόριθμων. Η ταυτόχρονη ελάττωση του κόστους των μνημών (*RAM*), των επεξεργαστών και γενικά της υπολογιστικής ισχύος, έχει κάνει οικονομικά βιώσιμη την ανάπτυξη συστημάτων επικοινωνίας και επεξεργασίας εικόνων ακόμα και για οικιακή χρήση.

Οι αλγόριθμοι επεξεργασίας εικόνων εξυπηρετούν διάφορους σκοπούς, όπως:

- Τη βελτίωση της ποιότητας των εικόνων, με τη χρήση κατάλληλων φίλτρων ή την αποκατάστασή τους στην αρχική τους μορφή μετά από αλλοίωσή τους λόγω επίδρασης θορύβου.

- Την κωδικοποίησή τους, έτσι ώστε η πληροφορία τους να μπορεί να περιγραφεί από μία σειρά όσο γίνεται μικρότερου αριθμού N bit (*συμπίεση δεδομένων*). Αυτό έχει σκοπό τη γρήγορη μετάδοσή τους μέσω διαύλων περιορισμένης

χωρητικότητας (*bandwidth*) ή την αποτελεσματική αποθήκευσή τους σε περιορισμένο αποθηκευτικό χώρο, με ικανοποιητική ποιότητα εικόνας.

- Τη μετατροπή φωτογραφιών σε εικόνες δύο μόνο αποχρώσεων (*μαύρου-άσπρου*), για εκτύπωση ή επίδειξη σε δυαδική μορφή.

- Την τροποποίηση των εικόνων (π.χ. *pixelate*) εφαρμόζοντας επάνω τους «καλλιτεχνικές» φόρμες και απόψεις.

- Την αναγνώριση σκηνής (*scene detection*)

- Την αναγνώριση αντικειμένων (*object recognition*)

2.2 Μετατροπή των εικόνων σε χαρακτηριστικά

Στην υπολογιστική όραση (*computer vision*) και στην επεξεργασία εικόνας (*image processing*) ένα χαρακτηριστικό (*feature*) είναι ένα κομμάτι πληροφορίας που σχετίζεται με μια συγκεκριμένη εφαρμογή. Τα χαρακτηριστικά (*features*) μπορεί να είναι συγκεκριμένες δομές της εικόνας όπως σημεία, απότομες αλλαγές της φωτεινότητας ή αντικείμενα. Ένα χαρακτηριστικό (*feature*) μπορεί, επίσης, να αντιπροσωπεύει και μια ολόκληρη περιοχή από *pixels*, πάνω στην εικόνα. Άλλα χαρακτηριστικά (*features*) μπορεί να σχετίζονται με την κίνηση σε μια ακολουθία από εικόνες, σε ορισμένες μορφές καμπύλων-σύνορα μεταξύ διαφορετικών περιοχών της εικόνας καθώς και με ιδιομορφίες μιας συγκεκριμένης περιοχής. Επίσης, άλλες απλές πληροφορίες που μπορούν να χαρακτηρίζουν μία εικόνα είναι η μέση τιμή *και η διασπορά της φωτεινότητας των pixels*. Η έννοια του χαρακτηριστικού (*feature*) είναι πολύ γενική και η επιλογή του σε μια εφαρμογή, είναι σε μεγάλο βαθμό συνδεδεμένη με το εκάστοτε πρόβλημα προς λύση.

2.2.1 Εξαγωγή χαρακτηριστικών

Στα πεδία έρευνας της μηχανικής εκμάθησης (*machine learning*), της αναγνώρισης προτύπων (*pattern recognition*) και της επεξεργασίας εικόνας (*image processing*) η εξαγωγή χαρακτηριστικών ξεκινά από ένα αρχικό σύνολο μετρούμενων δεδομένων και δημιουργεί παραγόμενες αντιπροσωπευτικές τιμές (*features*), οδηγώντας σε γενικευμένα βήματα εκμάθησης (*learning*). Ποιο συγκεκριμένα:
Αν έχουμε ένα πολύ μεγάλο σύνολο δεδομένων εισόδου σε έναν αλγόριθμο προς επεξεργασία και δεν είναι απαραίτητο να επεξεργαστεί εξολοκλήρου (π.χ όλα τα pixel μιας εικόνας), τότε το σύνολο εισόδου μπορεί να μετασχηματιστεί σε ένα μειωμένο σε

αριθμό σύνολο χαρακτηριστικών (*features or features vector*). Η παραπάνω διαδικασία ονομάζεται εξαγωγή χαρακτηριστικών(*feature extraction*). Τα προκύπτοντα χαρακτηριστικά περιμένουμε να περιλαμβάνουν τη σχετική πληροφορία των δεδομένων εισόδου, έτσι ώστε ο αρχικός σκοπός να επιτευχθεί χρησιμοποιώντας τη μειωμένη αναπαράσταση απ' ότι όλα τα αρχικά δεδομένα. Γενικά, η εξαγωγή χαρακτηριστικών αναφέρεται στον τρόπο που θα αναπαραστήσουμε τα ενδιαφέροντα σημεία μία εικόνας για να μπορέσουμε να τα συγκρίνουμε με άλλα ενδιαφέροντα σημεία.

2.2.2 Ανίχνευση χαρακτηριστικών

Η έννοια της ανίχνευσης ενός χαρακτηριστικού (*feature detection*) στους τομείς της υπολογιστικής όρασης (*computer vision*) και της επεξεργασίας εικόνας (*image processing*), αναφέρεται σε μεθόδους που χρησιμοποιούνται για να πάρουμε πληροφορίες από μια εικόνα και να αποφανθούμε για διάφορες περιοχές της, αν υπάρχει ή όχι το δεδομένο χαρακτηριστικό που ψάχνουμε. Τα χαρακτηριστικά που τελικά παράγονται είναι υποσύνολα των περιοχών της εικόνας, σε μορφή μεμονωμένων σημείων, συνεχών καμπυλών ή συνδεδεμένων περιοχών. Η ανίχνευση χαρακτηριστικών αναφέρεται γενικά σε τρόπους με τους οποίους μπορούμε να βρούμε τα χαρακτηριστικά μέσα σε μία εικόνα.

Δεν υπάρχει γενικός ή ειδικός ορισμός για το τι μπορεί να αποτελεί ένα χαρακτηριστικό. Ανάλογα με το πρόβλημα ή την εφαρμογή, δίνεται και ο κατάλληλος ορισμός. Δεδομένου των παραπάνω, ένα χαρακτηριστικό ορίζεται ως ένα ενδιαφέρον κομμάτι (*interesting part*) της εικόνας και αποτελεί την αρχή πολλών αλγορίθμων. Η ανίχνευση χαρακτηριστικών (*features*) είναι μια διαδικασία χαμηλού επιπέδου στην επεξεργασία εικόνας. Για αυτό το λόγο εφαρμόζεται σαν πρώτο βήμα, εξετάζοντας κάθε θέση της εικόνας, για το αν τα *pixel* της γειτονιάς της μπορούν να αντιπροσωπεύσουν ένα χαρακτηριστικό (*feature*). Καθώς τα χαρακτηριστικά χρησιμοποιούνται σαν αρχή για πολλούς αλγορίθμους, ο αλγόριθμος στο σύνολο του είναι τόσο καλός όσο ο ανιχνευτής χαρακτηριστικού (*feature detector*). Η επιθυμητή ιδιότητα ενός ανιχνευτή χαρακτηριστικού είναι η επαναληψιμότητα (*repeatability*). Αυτή αναφέρεται στην ανίχνευση του ίδιου χαρακτηριστικού σε 2 ή περισσότερες εικόνες της ίδιας σκηνής. Υπάρχουν πολλοί αλγόριθμοι που χρησιμοποιούν την ανίχνευση χαρακτηριστικού (*feature detection*) σαν βήμα αρχικοποίησης. Αυτό έχει ως αποτέλεσμα την ανακάλυψη μεγάλου αριθμού ανιχνευτών (*feature detectors*).

2.2.3 Τύποι χαρακτηριστικών

a) Ακμές (*Edges*)

Οι ακμές είναι συνεχόμενα σημεία που δημιουργούν μέσα σε μια εικόνα όρια μεταξύ περιοχών. Πρακτικά, τις ακμές μπορούμε να τις ορίσουμε ως ένα σύνολο σημείων μέσα σε μια εικόνα, τα οποία έχουν μια έντονη κλίση.

b) Γωνίες / Σημεία *(Corners / Interest points)*

Οι γωνίες/ενδιαφέροντα σημεία, αναφέρονται σε σημειακά χαρακτηριστικά (*features*) μέσα σε μια εικόνα, τα οποία παρουσιάζουν μια δυσδιάστατη τοπική δομή. Οι πρώτοι αλγόριθμοι πραγματοποιούσαν ανίχνευση ακμής και στη συνέχεια γινόταν ανάλυση των ακμών για να βρεθούν γρήγορες αλλαγές κατεύθυνσης. Έκτοτε, με την ανάπτυξη των αλγορίθμων εξετάζονται τα σημεία όπου έχουμε μεγάλα επίπεδα καμπυλότητας στην κλίση της εικόνας και έτσι η ανίχνευση ακμής δεν είναι πλέον αναγκαία. Η καμπυλότητα στην κλίση δεν εμφανίζεται μόνο σε γωνίες αλλά και σε κομμάτια της εικόνας, για παράδειγμα, ένα φωτεινό σημείο με σκούρο φόντο.

c) Περιοχές Ενδιαφέροντος (Blobs / regions of interest or interest points)

Οι περιοχές ενδιαφέροντος προσδίδουν μια συμπληρωματική περιγραφή των δομών μιας εικόνας, απ' ότι οι γωνίες οι οποίες είναι σημειακές δομές. Παρόλα αυτά, οι περιοχές ενδιαφέροντος μπορεί να είναι και αυτές σημειακές, π. χ ένα τοπικό μέγιστο στην εικόνα, το οποίο έχει ως αποτέλεσμα πολλοί ανιχνευτές περιοχών ενδιαφέροντος (*Blobs detectors*) να θεωρηθούν και ανιχνευτές σημείων (*corner detector*). Οι ανιχνευτές περιοχών ενδιαφέροντος μπορούν να ανιχνεύσουν περιοχές σε μία εικόνα, οι οποίες είναι πολύ ομαλές για να ανιχνευθούν από έναν ανιχνευτή γωνίας (*corner detector*). Υπάρχουν, όμως, περιπτώσεις που έχουμε συρρίκνωση της εικόνας και ο ανιχνευτής γωνίας μπορεί να βρει ένα σημείο κορυφής, το οποίο στην αρχική εικόνα να είναι ομαλό. Από τα παραπάνω, παρατηρείται μια αοριστία στη διαφορά μεταξύ των δυο ανιχνευτών. Η αοριστία αυτή μπορεί να διορθωθεί συμπεριλαμβάνοντας μία κατάλληλη έννοια κλίμακας. Λόγο της απόκρισης που έχουν σε διαφορετικές δομές της εικόνας για διαφορετικές κλίμακες, κάποιοι ανιχνευτές περιοχών ενδιαφέροντος αναφέρονται και σε άρθρα σχετικά με την ανίχνευση γωνιών.

d) Κορυφογραμμές/ Προεξοχές (Ridges)

Ένας περιγραφέας προεξοχών (*ridge descriptor*) υπολογίζεται από το γκρίζο επίπεδο μιας εικόνας. Πρακτικά, μια προεξοχή μπορεί να θεωρηθεί μια μονοδιάστατη καμπύλη που αντιπροσωπεύει έναν άξονα συμμετρίας. Επιπρόσθετα, έχει ένα χαρακτηριστικό

τοπικού πλάτους που συνδέεται με ένα σημείο της προεξοχής. Η εξαγωγή κορυφογραμμών/προεξοχών (*Ridges*) είναι πιο δύσκολο αλγοριθμικά από την εξαγωγή χαρακτηριστικών ακμών, γωνιών και περιοχών. Παρότι είναι δύσκολη η εξαγωγή τους, έχουν συχνή χρήση στην εξαγωγή δρόμων από αεροφωτογραφίες και την εξαγωγή αιμοφόρων αγγείων από ιατρικές εικόνες.

Παραδείγματα ανιχνευτών:

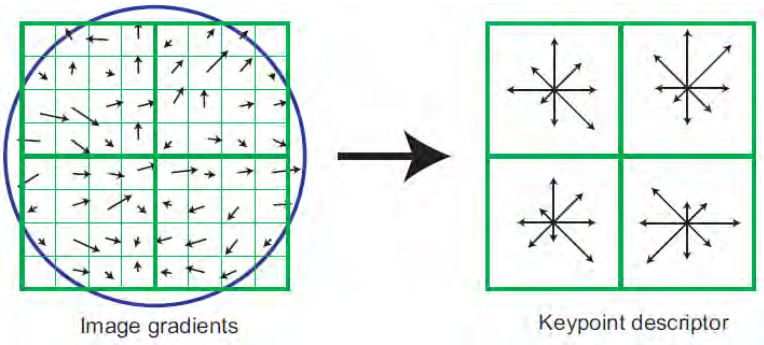
Ανιχνευτές Χαρακτηριστικών /Features detectors	**Edge**	**Corner**	**Blob**	**Ridge**
Canny[35]	**X**			
Sobel[36]	**X**			
Harris & Stephens/Plessey[37]	**X**	**X**		
SUSAN[37]	**X**	**X**		
Shi & Tomasi[37]		**X**		
Level-curve curvature[37]		**X**		
FAST[38]		**X**	**X**	
Laplacian of Gaussian[39]		**X**	**X**	
Difference of Gaussian[40]		**X**	**X**	
Determinant of Hessian[39]		**X**	**X**	
MSER[41]			**X**	
PCBR[42]			**X**	
Grey-level blobs[39]			**X**	
Hough transform[43]				**X**
Structure tensor[44]				**X**

2.2.4 Τοπικοί περιγραφείς

Μετά την ανίχνευση των χαρακτηριστικών, μπορούμε να εξάγουμε ένα τοπικό κομμάτι της εικόνας γύρω από κάθε χαρακτηριστικό. Η εξαγωγή περιλαμβάνει αρκετά βήματα επεξεργασίας της εικόνας. Το αποτέλεσμα αυτών των βημάτων είναι γνωστό ως χαρακτηριστικός περιγραφέας ή χαρακτηριστικό διάνυσμα. Παρακάτω αναφέρεται περιληπτικά η μορφή που έχουν οι τοπικοί περιγραφείς, με βάση τον αλγόριθμο που χρησιμοποιήθηκε για την εξαγωγή τους.

a) *SIFT (Scale-Invariant-Feature-Transform)[1]*

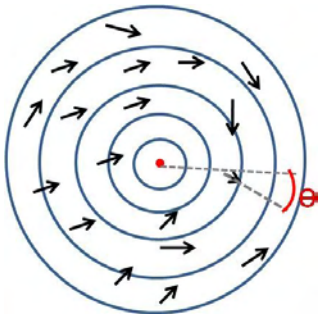
Ο αλγόριθμος SIFT χρησιμοποιείται στην όραση υπολογιστών (*computer vision*) για να ανιχνεύσει και να περιγράψει τοπικά χαρακτηριστικά (*features*) σε εικόνες. Οι περιγραφείς (*descriptors*) που παράγει ο SIFT παραμένουν αμετάβλητοι όταν έχουμε αλλαγή του μεγέθους της εικόνας ή περιστροφή της. Επιπρόσθετα, παρουσιάζουν ανθεκτικότητα σε αλλαγές φωτισμού, παρουσία θορύβου καθώς και σε μικρές αλλαγές της οπτικής γωνίας της κάμερας. Η πλήρης εξήγηση του αλγορίθμου δίδεται στο Κεφάλαιο 3. Στην Εικόνα 2.2 φαίνεται η μορφή που έχει ένας SIFT περιγραφέας.



Εικόνα 2.2 SIFT Descriptor

b) RIFT *(Rotation- Invariant-Feature-Transform)[18]*

Ο αλγόριθμος RIFT είναι μία γενίκευση του SIFT. Ο RIFT περιγραφέας (*descriptor*) κατασκευάζεται με κυκλικά κανονικοποιημένα κομμάτια (*patches*) της εικόνας. Αυτά, χωρίζονται σε ομόκεντρους δακτυλίους ίσου πλάτους και μέσα σε κάθε δακτύλιο υπολογίζεται ένα ιστόγραμμα κλίσης-κατεύθυνσης. Για να διατηρηθεί ο περιγραφέας αναλλοίωτος κατά την περιστροφή, ο προσανατολισμός μετράται σε κάθε σημείο σε σχέση με την κατεύθυνση που δείχνει από το κέντρο προς τα έξω. Στην Εικόνα 2.3 φαίνεται η μορφή που έχει ο RIFT περιγραφέας.



Εικόνα 2.3 RIFT Descriptor

c) G-RIF *(Generalized Robust Invariant Feature)[19]*

Ο G-RIF είναι ένας αλγόριθμος που παράγει ένα γενικό περιγραφέα (*descriptor*) ο οποίος κωδικοποιεί τον προσανατολισμό και την πυκνότητα της ακμής καθώς και πληροφορίες απόχρωσης. Ο αλγόριθμος είναι εμπνευσμένος από τον τρόπο λειτουργίας του οπτικού μας συστήματος, το οποίο έχει την ικανότητα να διαχωρίζει τα στοιχεία μιας εικόνας με βάση την αντίθεση (*π.χ φως, σκοτάδι*) και να αντιληφθεί ένα αντικείμενο μέσα σε ένα περιβάλλον.

d) SURF *(Speeded Up Robust Features)[7]*

Ο SURF είναι ένας υψηλής απόδοσης ανιχνευτής/ περιγραφέας (*detector / descriptor*) σημείων ενδιαφέροντος (*interest point*), τα οποία παραμένουν αμετάβλητα κατά την αλλαγή κλίμακας και κατά την περιστροφή. Ο SURF βασίζεται στις συνελίξεις μεταξύ των integral images για μείωση του χρόνου υπολογισμού και στα δυνατά σημεία των ήδη υπαρχόντων ανιχνευτών και περιγραφέων (*descriptors*). Χρησιμοποιεί ένα γρήγορο μέτρο βασισμένο στον Hessian-matrix για τον ανιχνευτή (*detector*) και για έναν κατανεμημένο περιγραφέα (*descriptor*). Περιγράφει μια κατανομή αποκρίσεων των Haar wavelet στη γειτονιά του σημείου ενδιαφέροντος (*interest point*). Οι περιγραφείς (*descriptors*) που παράγει ο SURF είναι 64 διαστάσεων, οι οποίοι μειώνουν το χρόνο υπολογισμού του χαρακτηριστικού (*feature*) καθώς και το χρόνο που χρειάζεται για το ταίριασμα των χαρακτηριστικών (*feature Matching*).

e) PCA-SIFT *(Principal component analysis-SIFT)[21]*

Ο PCA-SIFT είναι παραλλαγή του αλγόριθμου SIFT. Εδώ ο περιγραφέας (*descriptor*) είναι ένα διάνυσμα των κλίσεων της εικόνας (*image gradients*). Η περιοχή που καταλαμβάνει ένας περιγραφέας είναι 39x39 pixel και το διάνυσμα που προκύπτει είναι 3042 διαστάσεων. Με την χρήση του PCA οι 3042 διαστάσεις μειώνονται σε 36.

f) GLOH *(Gradient location orientation histogram)[22]*

Ο αλγόριθμος GLOH είναι και αυτός μία επέκταση του SIFT. Ο GLOH παράγει ένα SIFT descriptor για ένα λογαριθμικό πολικό πλέγμα της τοποθεσίας που περιέχει 3 θέσεις για ακτινωτή κατεύθυνση (του συνόλου 6, 11, 15) και 8 θέσεις για γωνιακή κατεύθυνση, έτσι υπάρχουν συνολικά 17 θέσεις. Οι κλίσεις των γωνιών κβαντίζονται σε 16 θέσεις, το οποίο έχει ως αποτέλεσμα ένα ιστόγραμμα 272 θέσεων. Το μέγεθος του περιγραφέα (*descriptor*) μειώνεται με τη χρήση του PCA σε 64 διαστάσεις.

2.3 Αντιστοίχιση Χαρακτηριστικών

Ένας από τους λόγους που γίνεται η εξαγωγή χαρακτηριστικών από τις εικόνες είναι για να μπορέσουμε να τις συγκρίνουμε με άλλες, βασιζόμενοι στα χαρακτηριστικά. Όλοι οι αλγόριθμοι που αναφέρονται στην προηγούμενη ενότητα παράγουν ένα χαρακτηριστικό περιγραφέα που στην ουσία είναι ένα διάνυσμα αριθμών k-διαστάσεων. Συνεπώς, η σύγκριση των εικόνων ανάγεται σε συγκρίσεις και αντιστοιχίσεις διανυσμάτων. Ο καλύτερος τρόπος αντιστοίχισης ενός διανύσματος είναι η εύρεση του Κοντινότερου Γείτονα μέσα σε ένα σύνολο άλλων διανυσμάτων.

2.3.1 Πρόβλημα Κοντινότερου Γείτονα (Nearest Neighbor, NN)

Δεδομένου ενός συνόλου διανυσμάτων $T = \{t^i \epsilon R^d, i = 1, \dots, M\}$ και ενός διανύσματος $q \epsilon R^d$, η εύρεση του διανύσματος με τη μικρότερη απόσταση $d(q, t^i)$ ορίζεται ως το πρόβλημα του Κοντινότερου Γείτονα NN. Στην παρούσα εργασία χρησιμοποιούμε $d(q, t) = \|q - t\|^2$, η οποία είναι η Ευκλείδεια απόσταση δύο διανυσμάτων.

2.3.2 Επακριβή Εύρεση Κοντινότερου Γείτονα (Exact Nearest Neighbor)

Η πιο προφανής αναζήτηση Κοντινότερου Γείτονα γίνεται εξετάζοντας γραμμικά όλα τα διανύσματα. Η πολυπλοκότητα της αναζήτησης είναι O(M * d) με M των αριθμό των διανυσμάτων της βάσης δεδομένων και d τις διαστάσεις των διανυσμάτων. Ο PDS (*Partial Distance Search*) βελτιώνει τη Full Search σταματώντας την αναζήτηση όταν ξεπεραστεί ένα όριο μικρότερης απόστασης και πάλι όμως η εξέταση γίνεται γραμμικά. Αλγόριθμοι βασισμένοι σε δυαδικά δέντρα έδωσαν τη λύση μετατρέποντας το χρόνο αναζήτησης από γραμμικό σε σχεδόν λογαριθμικό[2,10,15,16,20]. Σαν πρώτο βήμα, αρχικοποιούν τα *training* δεδομένα σε μία κατάλληλη δενδρική δομή. Τυπικά, η αρχικοποίηση ξεκινά από τη ρίζα του δέντρου κατανέμοντας τα *training* δεδομένα στα παιδιά της βάση κάποιων υπερ-επιπέδων. Η διαδικασία συνεχίζεται για όλα τα παιδιά μέχρι κάθε παιδί να μην μπορεί να χωριστεί άλλο. Στο στάδιο αναζήτησης οι αλγόριθμοι διατρέχουν το δέντρο μέχρι να βρεθούν σε φύλλο και στη συνέχεια αρχίζουν μία οπισθοδρόμηση σε προηγούμενους κόμβους, έτσι ώστε να εξεταστούν και τα υπόλοιπα δεδομένα. Η λειτουργία των αλγορίθμων βασίζεται σε μία έννοια υπερ-επίπεδου και ενός κάτω ορίου.

2.3.3 Προσεγγιστική Εύρεση Κοντινότερου Γείτονα (Approximate Nearest Neighbor, ANN)

Η αναζήτηση *exact* NN όταν οι διαστάσεις των διανυσμάτων γίνονται μεγάλες είναι μη αποδοτικές. Για το λόγο αυτό έχουν προταθεί πολλοί αλγόριθμοι που βρίσκουν προσεγγιστικά τον Κοντινότερο Γείτονα με ταχύτητα και ακρίβεια. Οι προσεγγιστικοί αλγόριθμοι χωρίζονται σε τρεις κατηγορίες όπως εξηγείται στο [20]:

a) Δέντρα Διαχωρισμού (*Partitioning trees*)

Οι αλγόριθμοι που βασίζονται σε δέντρα διαχωρισμού (ή δυαδικά δέντρα) κάνουν μία εις βάθος αναζήτηση και τερματίζουν, γεγονός που τους κάνει να έχουν ταχεία απόκριση. Σε άλλες περιπτώσεις θέτοντας ένα όριο στον αριθμό των *backtracking* ή στο χρόνο αναζήτησης μπορούμε να έχουμε ταχύτητα και καλή ακρίβεια στην εύρεση του NN.

b) Πίνακες Κατακερματισμού (*Hash Tables*)

Η LSH (*Locality Sensitive Hashing*)[21] είναι μία μέθοδος για να πετύχουμε μείωση των διαστάσεων, αντιστοιχίζοντας παρόμοια διανύσματα στις ίδιες θέσεις του πίνακα κατακερματισμού με μεγάλη πιθανότητα. Η απόδοση των πινάκων κατακερματισμού βασίζεται σε μεγάλο βαθμό στην επιλογή της κατάλληλης συνάρτησης κατακερματισμού. Για το λόγο αυτό έχουν προταθεί μέθοδοι που βελτιστοποιούν τις αναζητήσεις σε πίνακες κατακερματισμού, όπως *parameter sensitive hashing*[22], *kernelized LSH*[23] και *optimized kernel hashing*[24]. Παρόλο που οι πίνακες κατακερματισμού χρησιμοποιούνται σε πολλές εφαρμογές, στο [20] παρουσιάζεται ότι υστερούν σε σχέση με αλγορίθμους που χρησιμοποιούν δέντρα διαχωρισμού.

c) Τεχνικές ANN βασισμένες σε Γράφους (*Graph-base techniques*)

Σε αυτή την κατηγορία οι αλγόριθμοι παρομοιάζουν τα διανύσματα της βάσης δεδομένων σαν ακμές ενός Γράφου, όπου κάθε ακμή ενώνει κάθε διάνυσμα με τον/τους Κοντινότερους Γείτονες. Η αναζήτηση ξεκινά τυχαία από ένα σημείο αρχικοποίησης και προχωρά προς το διάνυσμα που αναζητείται. [14]

2.4 Αλγόριθμοι που χρησιμοποιήθηκαν στην παρούσα εργασία

Στα πλαίσια της παρούσας εργασίας για την ανίχνευση και την εξαγωγή χαρακτηριστικών χρησιμοποιήθηκε ο αλγόριθμος SIFT. Ο λόγος που χρησιμοποιήθηκε είναι διότι, δίνει τα πιο ευσταθή χαρακτηριστικά σε σχέση με τους άλλους αλγορίθμους.
Για την αναζήτηση NN χρησιμοποιήθηκε ο αλγόριθμος Fast NN [2, 3] σε τρεις παραλλαγές μεθόδων, δύο προσεγγιστικών αναζητήσεων NN και μίας *exact* αναζήτησης. Ο λόγος που χρησιμοποιήσαμε τον Fast NN είναι για να κάνουμε γρήγορη αναζήτηση NN και για να μελετηθεί η συμπεριφορά του αλγορίθμου για διαστάσεις διανυσμάτων άνω των 64-διαστάσεων. Επίσης, θέλαμε να δούμε την απόδοσή του όταν χρησιμοποιηθεί σε ένα σύστημα αναζήτησης και ανάκτησης εικόνων.

Κεφάλαιο 3

Γενική Περιγραφή SIFT αλγορίθμου

Ο SIFT είναι ένας πολύ διαδεδομένος αλγόριθμος εξαγωγής χαρακτηριστικών στα πεδία έρευνας της υπολογιστικής όρασης (*computer vision*). Αποτελεί την αρχή και τη βάση πολλών εφαρμογών, καθώς και την έμπνευση για άλλους μετέπειτα αλγόριθμους (έχει πάνω από 27K αναφορές). Η χρήση του παρατηρείται σε εφαρμογές όπως αναγνώριση αντικειμένων (*object recognition*), πλοήγηση ρομπότ (*robotic mapping and navigation*), image stitching, 3D modeling, αναγνώριση χειρονομιών (*gesture recognition*) και βίντεο παρακολούθησης (*video tracking*). Όπως έχει αναφερθεί στην Ενότητα 2.2, ο αλγόριθμος παράγει πολύ ανθεκτικά χαρακτηριστικά (*features*). Κάθε χαρακτηριστικό προέρχεται από ένα σημείο πάνω στην εικόνα το οποίο είναι ένα ενδιαφέρον σημείο. Τα χαρακτηριστικά ορίζονται στο χωρικό πεδίο αλλά και στο πεδίο συχνότητας. Μια τυπική εικόνα μεγέθους 500x500 pixel παράγει περίπου 2000 ευσταθή χαρακτηριστικά. Κάθε χαρακτηριστικό μπορεί να αναπαρασταθεί βάση ενός διανύσματος-περιγραφεά το οποίο είναι μοναδικό για κάθε χαρακτηριστικό. Όπως αναφέρεται στην Ενότητα 2.2, επιθυμητή ιδιότητα ενός χαρακτηριστικού περιγραφεά είναι η επαναληψιμότητα (*repeatability*). Η μοναδικότητά του διανύσματος μας δίνει τη δυνατότητα να αντιστοιχήσουμε σωστά ένα χαρακτηριστικό μέσα σε μία τεράστια βάση δεδομένων από άλλα χαρακτηριστικά.

Ο αλγόριθμος εξαγωγής αποτελείται από 4 κύρια βήματα.

Βήμα 1^{ο}: Εύρεση ακροτάτων σημείων στην κλίμακα και στο χώρο. Κάθε ακρότατο είναι ένα υποψήφιο σημείο για να γίνει χαρακτηριστικό *(Scale – Space Extrema Detection*)

Βήμα 2^{ο}: Εντοπισμός των σημείων με βάση την ευστάθεια που έχουν παρουσία θορύβου, για να αποτελέσουν χαρακτηριστικά σημεία της εικόνας (*Keypoint localization*)

Βήμα 3^{ο}: Ανάθεση προσανατολισμού στα σημεία[25], με βάση τους προσανατολισμούς των τοπικών στοιχείων της εικόνας γύρω από κάθε σημείο (*Orientation assignment*)

Βήμα 4^{ο}: Δημιουργία του διανύσματος-περιγραφέα για κάθε σημείο που έχει περάσει από τα προηγούμενα στάδια (*Keypoint descriptor*)

3.1 Scale – Space Extrema Detection

3.1.1 Εισαγωγή

Αρχικά, οι έγχρωμες εικόνες μετατρέπονται σε μονόχρωμες (*gray level image*), καθώς αυτό μειώνει τα δεδομένα και είναι γνωστό ότι το ανθρώπινο μάτι είναι πιο ευαίσθητο στη φωτεινότητα παρά στο χρώμα[17]. Το πρώτο στάδιο του αλγορίθμου βασίζεται στη δημιουργία δύο πυραμίδων. Η πρώτη, αποτελείται από την εικόνα θολωμένη σε διαφορετικές κλίμακες (*scale-space*). Η θόλωση της εικόνας γίνεται με έναν Gaussian Smoothing kernel που υπο ορισμένες προϋποθέσεις είναι ο μονός «scale-space kernel» που μπορεί να χρησιμοποιηθεί για να πετύχουμε γραμμική μετάβαση καθώς πάμε από τη μία κλίμακα στην άλλη, όπως έχουν δείξει οι Koenderink[26] και Lindeberg[27]. Έτσι, έχουμε μια συνάρτηση $L(x, y, \sigma)$ που ορίζεται ως η συνέλιξη του Gaussian kernel με την εικόνα $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \qquad (1)$$

όπου είναι το σύμβολο της συνέλιξης για x, y και ,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}. \qquad (2)$$

Η πρώτη πυραμίδα αποτελείται από εικόνες *L(x, y, σ)*. Το σύνολο των εικόνων *L(x, y, σ)* που έχουν το ίδιο μήκος και το ίδιο πλάτος, αποτελεί μία οκτάβα της πυραμίδας των *Gaussian Blurred* εικόνων.

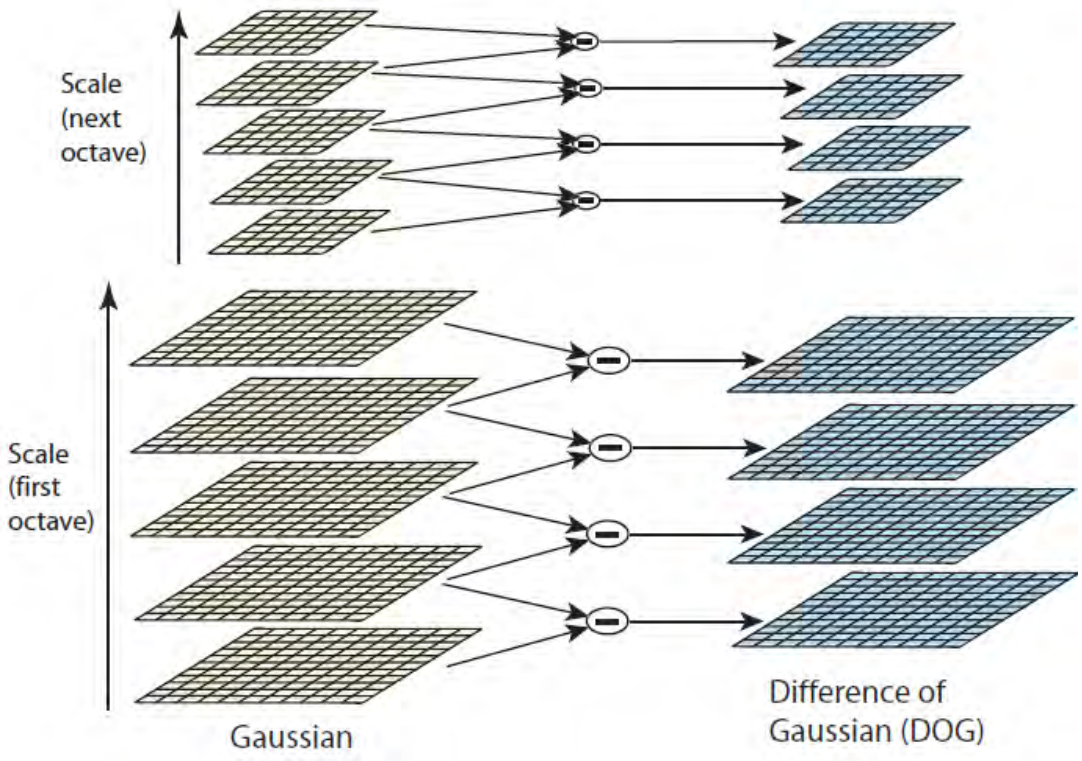
Η δεύτερη πυραμίδα προκύπτει από την πρώτη και ονομάζεται *Difference of Gaussian Pyramid*. Όπως έχει προταθεί στο [28], η ανίχνευση σημείων σε σταθερή θέση με αποδοτικό τρόπο γίνεται με τη χρήση των ακρότατων των *Difference of Gaussian* εικόνων. Μία *Difference of Gaussian* εικόνα ορίζεται ως *D(x, y, σ)* και είναι η διαφορά δύο *Gaussian Blurred* εικόνων, οι οποίες διαφέρουν στην τυπική απόκλιση κατά ένα παράγοντα k:

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$
$$L(x, y, k\sigma) - L(x, y, \sigma). \qquad (3)$$

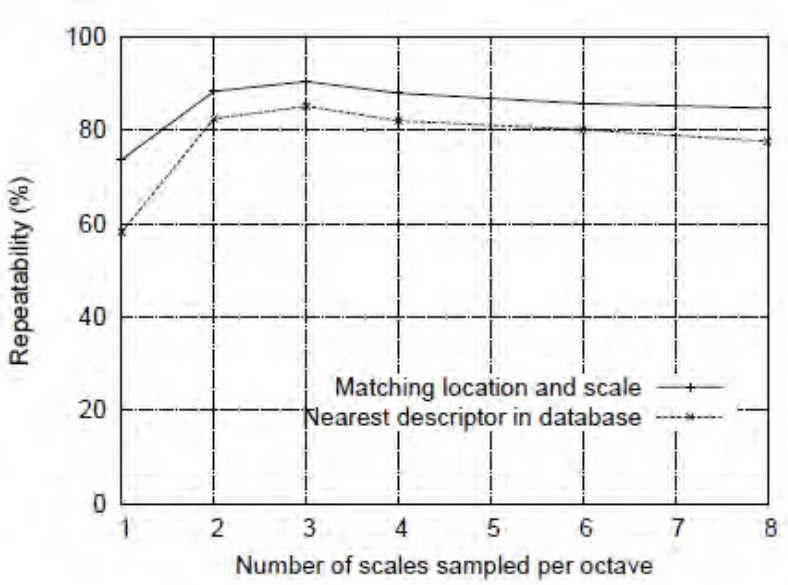
Οι λόγοι που επιλέχτηκε η συνάρτηση (3) είναι ποικίλοι. Αρχικά, είναι αρκετά εύκολος και γρήγορος ο υπολογισμός της, καθώς παράγεται από τη διαφορά δύο εικόνων. Αλλά ο κύριος λόγος είναι το γεγονός ότι αποτελεί μία πολύ κοντινή προσέγγιση της κανονικοποιημένης σε κλίμακα *Laplacian of Gaussian*, $\sigma^2\nabla^2 G$, όπως έχει δειχθεί από τον Lindeberg[27]. Για να παραμείνουν τα keypoints αναλλοίωτα σε

μεταβολές της κλίμακας *(scale invariance)*, πρέπει να χρησιμοποιηθεί η κανονικοποιημένη *Laplacian of Gaussian* με συντελεστή σ^2. Ο Mikolajczyk[29] έδειξε ότι τα ακρότατα της $\sigma^2\nabla^2 G$ παράγουν πιο ευσταθή χαρακτηριστικά σε σχέση με αλλά που παράγονται από την Hessian determinant[39] ή τον Harris corner detector[37].

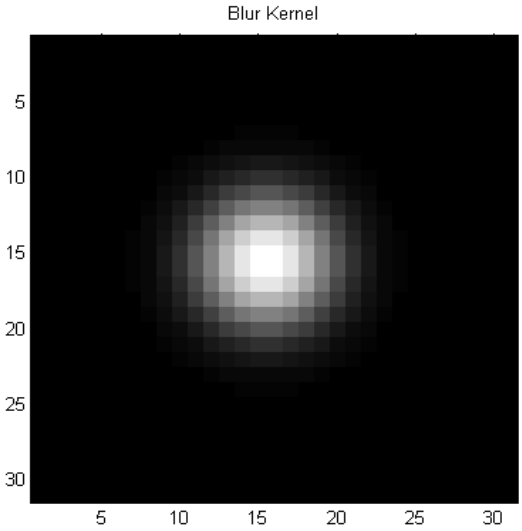
Στην Εικόνα 3.1 φαίνεται ένα παράδειγμα για τον τρόπο που είναι δομημένες οι δύο πυραμίδες. Στα αριστερά, έχουμε τις *Gaussian Blurred* εικόνες και στα δεξιά τις *Difference of Gaussian.* Κάνοντας συνέλιξη της αρχικής μας εικόνας *I(x,y)* με Gaussian kernels παράγονται *Gaussian Blurred* εικόνες που διαφέρουν κατά ένα παράγοντα k. Κάθε οκτάβα είναι χωρισμένη σε ένα ακέραιο αριθμό διαστημάτων (*intervals)*, s, έτσι ώστε $k = 2^{1/s}$. Στην εργασία που έχει παρουσιαστεί στο [1] δε δίνεται ακριβής ορισμός των *s* διαστημάτων (*intervals*). Με τον όρο διαστήματα, εννοεί τον απαιτούμενο αριθμό κλιμάκων για τις οποίες, πρέπει να γίνει δειγματοληψία σε κάθε οκτάβα. Όπως φαίνεται και στο Διάγραμμα 3.2 ο αριθμός κλιμάκων (*scales*) για κάθε οκτάβα που μας δίνει τη μεγαλύτερη επαναληψιμότητα στην ανίχνευση των keypoints είναι *s* = 3.



Εικόνα 3.1



Διάγραμμα 3.2



Εικόνα 3.3 Gaussian Kernel

3.1.2 Δημιουργία Gaussian και Difference of Gaussian πυραμίδων

Ο λόγος που δημιουργούνται οι δύο πυραμίδες είναι ουσιαστικά για να μπορέσουμε να εντοπίσουμε τα ακρότατα των *DoG* εικόνων. Καθώς η διαδικασία εντοπισμού των ακρότατων βασίζεται στη δημιουργία των δύο πυραμίδων παρουσιάζουμε πρώτα τον τρόπο που δημιουργούνται οι πυραμίδες. Για να έχουμε τις εικόνες χωρισμένες κατά παράγοντα k, τα *σ* των *Gaussian kernel* πρέπει να είναι συγκεκριμένα. Ένα παράδειγμα για το πως δημιουργείται η πρώτη οκτάβα θα βοηθήσει στην κατανόηση:

Αρχικά, γίνεται η συνέλιξη της αρχικής εικόνας *I(x,y)* με ένα *Gaussian kernel G(x, y, σ_0)* και παράγεται από την (1) μια εικόνα L_0. Η μορφή που έχει ένας *Gaussian kernel* φαίνεται στην Εικόνα 3.3. Στη συνέχεια, γίνεται σταδιακά συνέλιξη μεταξύ L_0 και *G(x, y, σ_i)* για να παραχθεί η i^{th} εικόνα της *Gaussian Pyramid*, η οποία είναι ισοδύναμη με τη συνέλιξη της αρχικής εικόνας και ενός *Gaussian kernel G(x,y,kσ_0)*. Το αποτέλεσμα από τη συνέλιξη μίας εικόνας και δυο *Gaussian kernel* μπορεί να δειχθεί πολύ εύκολα, αν μεταφέρουμε τις συναρτήσεις μας στο πεδίο της συχνότητας με τη βοήθεια του μετασχηματισμού Fourier,

$$G_{\sigma_i} G_{\sigma_0} * f(x) \rightarrow \mathcal{G}_{\sigma_i} \times \mathcal{G}_{\sigma_0} \times \mathcal{F} \quad (4)$$

Ο μετασχηματισμός Fourier μιας Γκαουσιανής συνάρτησης, e^{ax^2} είναι:

$$\mathrm{F}[e^{ax^2}](\mathrm{t}) = \sqrt{\frac{\pi}{\alpha}} \times e^{-\pi^2 t^2 / a} \quad (5)$$

Αντικαθιστώντας στην (4) και εξισώνοντας με μια Γκαουσιανή που έχει $\sigma_G = k\sigma_0$ έχουμε:

$$e^{-t^2\sigma_i^2} e^{-t^2\sigma_0^2} = e^{-t^2 k^2 \sigma_0^2} \quad (6)$$

Παίρνοντας λογάριθμους έχουμε:

$$\sigma_i^2 + \sigma_0^2 = k^2\sigma^2 \leftrightarrow \sigma_i = \sigma_0\sqrt{k^2 - 1} \quad (7)$$

Έτσι, δημιουργείται η επόμενη *Gaussian Blurred* εικόνα, κάνοντας συνέλιξη την L_{i-1} και ενός *Gaussian kernel, G(x, y, σ_i)* με $\sigma_i = \sigma_0\sqrt{k^2 - 1}$. Δηλαδή,

$$L_i = G(x, y, \sigma_i) * L_{i-1} \, . \quad (8)$$

Για την εύρεση των τοπικών ακροτάτων χρειαζόμαστε εκτός από τα *s intervals* και δύο ακόμα επίπεδα *DoG* σε κάθε οκτάβα, ο λόγος παρουσιάζεται στην επόμενη ενότητα. Αφού, τα επίπεδα μιας οκτάβας για *DoG* είναι (*s+2*) τα επίπεδα για *Gaussian blurred* εικόνες θα είναι (*s+3*). Σύμφωνα με την παραπάνω μεθοδολογία, θα δημιουργηθεί ένας

αριθμός από (*s+3*) εικόνες L με κλίμακα σ_0, $k\sigma_0$, $k^2\sigma_0$, $k^3\sigma_0$, … κτλ. Αυτές οι εικόνες θα αποτελέσουν μία οκτάβα της πυραμίδας των *Gaussian Blurred*. Όταν ολοκληρωθεί η δημιουργία μίας οκτάβας *Gaussian Blurred*, η εικόνα i^{th} που έχει $\sigma_i = 2\sigma_0$ γίνεται υπο-δειγματοληψία παίρνοντας κάθε δεύτερο pixel σε κάθε γραμμή και στήλη. Αυτή θα είναι πάντα η εικόνα που απέχει 2 θέσεις από την κορυφή της κάθε οκτάβας. Έτσι, η προκύπτουσα εικόνα μετά την υπο-δειγματοληψία είναι η πρώτη της επόμενης οκτάβας και η επεξεργασία είναι πάλι η ίδια. Ο αριθμός των οκτάβων δεν είναι σταθερός και εξαρτάται σε μεγάλο βαθμό από το μέγεθος που έχει η αρχική εικόνα άλλα και από τις παραμέτρους που έχουμε δώσει. Ένας απλός τρόπος για τον υπολογισμό των οκτάβων είναι:

$$num_{octavs} = log_2\big(max(width, height)\big) - log_2(const), \text{ όπου } const = 16{,}32, \ldots.$$

Παίρνοντας τη διαφορά των προσκείμενων *Gaussian Blurred L(x,y,σ)* εικόνων (εικόνες που διαφέρουν κατά παράγοντα k) δημιουργείται η δεύτερη πυραμίδα *DoG*.
Όπως αναφέρθηκε παραπάνω, η αρχική εικόνα *I(x,y)* δεν είναι μέρος της πρώτης πυραμίδας. Η *Gaussian* πυραμίδα ξεκίνα με την L_0. Ο λόγος που η *I(x,y)* υφίσταται θόλωση πριν τη διαδικασία και δεν αποτελεί την πρώτη εικόνα είναι για να αποφύγουμε να είναι πολύ κοντά μεταξύ τους τα keypoints, το οποίο θα επέφερε περιορισμούς στο ρυθμό ανίχνευσης. Στο [1] χρησιμοποιείται $\sigma_0 = 1.6$ για *pre-smooth*. Κάνοντας *pre-smooth* την εικόνα, οι υψηλές συχνότητες της εικόνας εξομαλύνονται, πράγμα που θα μας έκανε να χάσουμε πολλά πιθανά ακρότατα. Έτσι, για να γίνει πλήρη χρήση της εικόνας και να αυξηθεί ο αριθμός των keypoints, διπλασιάζουμε το μέγεθος της εικόνας κάνοντας υπερ-δειγματοληψία. Για αποφυγή αναδίπλωσης γίνεται η υπόθεση ότι η *I(x,y)* εικόνα είναι θολωμένη κατά ένα παράγοντα $\sigma = 0.5$ και έτσι κάνοντας διπλασιασμό η νέα εικόνα θα έχει $\sigma = 1$. Στις Εικόνες 3.4 και 3.5 δίνονται σαν παραδείγματα οι δύο πρώτες οκτάβες των *Gaussian Blurred* και των *DoG*, αντίστοιχα. Η επεξεργασία έγινε με *Gaussian kernel* μεγέθους 13.

3.1.3 Εύρεση τοπικών ακροτάτων

Όπως φαίνεται στην Εικόνα 3.6 για την εύρεση των τοπικών ακροτάτων στην D_i *(x, y, σ)* κάθε keypoint συγκρίνεται με τους οχτώ γείτονές του στην τρέχουσα εικόνα D_i και τους εννιά γείτονές στην D_{i-1} και στην D_{i+1}. Για να αποτελέσει ένα pixel ακρότατο, πρέπει να έχει τη μεγαλύτερη ή τη μικρότερη ένταση από όλα τα γειτονικά pixel. Συνεπώς, θα πρέπει η ανίχνευση των keypoints να ξεκινά από την D_1 και όχι από την D_0 και κατά αντιστοιχία να τελειώνει στην αμέσως προηγούμενη της τελευταίας *DoG*. Παρατηρούμε, λοιπόν, ότι εκτός από τον αριθμό των *s intervals* που τον καθορίζουμε εμείς χρειαζόμαστε δύο ακόμα επίπεδα για την ανίχνευση των ακρότατων, δηλαδή, *s+2 DoG*. Έτσι, αφού ο αριθμός των *DoG* είναι *s+2,* ο αριθμός των *Gaussian* θα είναι *s+3,*

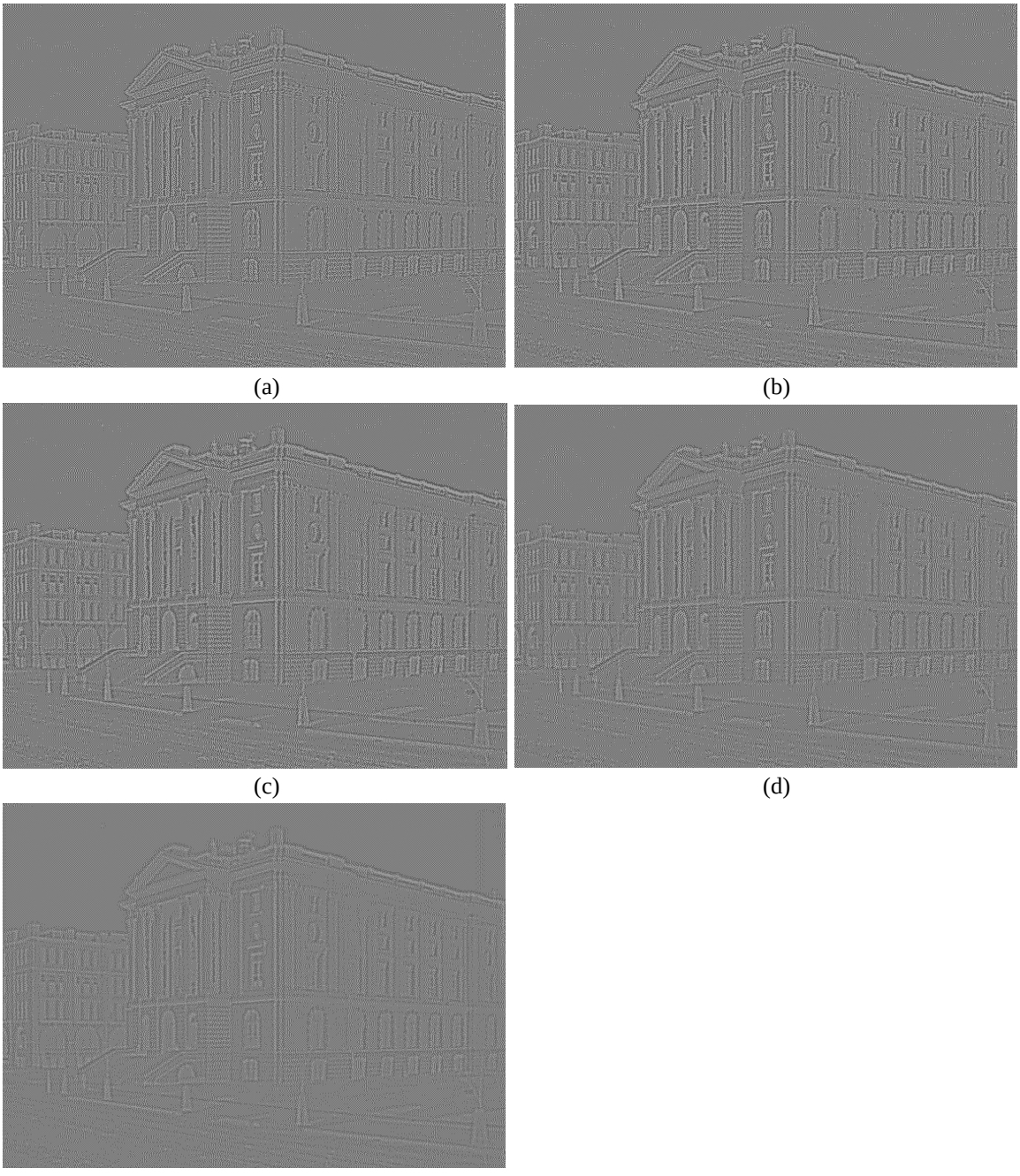
όπως αναφέρθηκε παραπάνω. Άρα, κάθε ακρότατο είναι ένα υποψήφιο χαρακτηριστικό (*feature*). Στην Εικόνα 3.7 φαίνονται οι θέσεις των ακροτάτων, όπως βρέθηκαν από την παραπάνω διαδικασία.

(a) (b)

(c) (d)

(e) (f)

Εικόνα 3.4 . Οι εικόνες (a)-(f) έχουν επεξεργαστεί με Gaussian kernel μεγέθους 13x13, $\sigma_0 = 1.6$ και $k = 2^{1/3} \sim 1.25$.

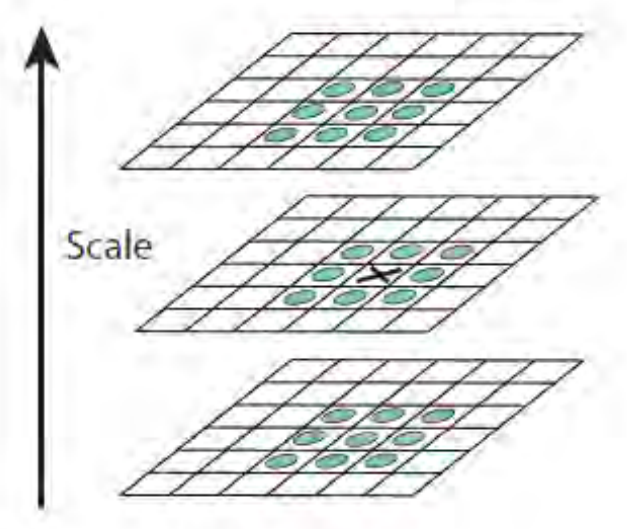


(a) (b)

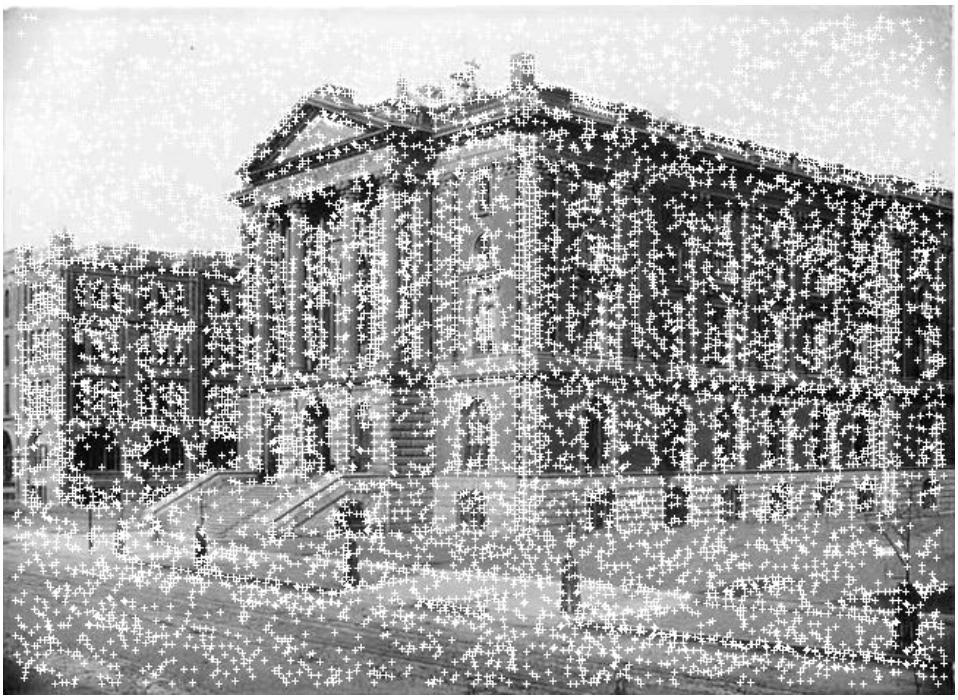
(c) (d)

(e)

Εικόνα 3.5 . Οι DoG που παράγονται από τις εικόνες που παρουσιάζονται στην Εικόνα 3.6



Εικόνα 3.6



Εικόνα 3.7 All Extrema

3.2 Accurate Keypoint Localization

Αφού τα υποψήφια χαρακτηριστικά (*features*) βρεθούν, περνάμε στο δεύτερο βήμα του αλγορίθμου. Για να βελτιώσουμε την ευστάθεια και τη μετέπειτα αντιστοίχιση των χαρακτηριστικών, αντί να κρατάμε την ακριβή θέση του ακρότατου, υπολογίζουμε την παρεμβαλλόμενη θέση, η οποία θα βρίσκεται στην sub-pixel περιοχή. Ο υπολογισμός του παρεμβάλλοντος ακρότατου βοηθά στην απόρριψη των υποψηφίων σημείων που παρουσιάζουν μικρή αντίθεση (*low contrast*) και σημείων που βρίσκονται πάνω σε ακμές. Ο λόγος που απορρίπτουμε αυτά τα σημεία είναι διότι παρουσιάζουν μεγάλη ευαισθησία στο θόρυβο.

3.2.1 Εύρεση παρεμβάλλοντος ακρότατου

Η εύρεση του παρεμβάλλοντος ακρότατου γίνεται με τη χρήση μιας 3D τετραγωνικής συνάρτησης[30]. Η συνάρτηση που χρησιμοποιήθηκε ήταν το ανάπτυγμα της σειράς Taylor, θέτοντας σε αυτή την *D(x, y, σ)*.

$$D(x_{offset}) = D(x_0) + \left(\frac{\partial D}{\partial x_{offset}}\Big|_{x0}\right)^T x_{offset} + \frac{1}{2} x_{offset}{}^T \left(\frac{\partial^2 D}{\partial x_{offset}{}^2}\Big|_{x0}\right) x_{offset} \qquad (9)$$

Στην (9) η D και οι παραγωγοί της υπολογίζονται για τη θέση του ακρότατου $x_0 = (x,y, \sigma)$, ενώ το x_{offset} είναι η απόσταση από το x_0, $x_{offset} = (\delta x, \delta y, \delta \sigma)$. Θέλοντας να βρούμε το ακρότατο της (9) κατά τα γνωστά από το διαφορικό λογισμό, παίρνουμε την παράγωγό της ως προς το x_{offset}και εξισώνουμε με το μηδέν. Έτσι προκύπτει:

$$\hat{x} = -\left(\frac{\partial^2 D}{\partial x_{offset}{}^2}\Big|_{x0}\right)\left(\frac{\partial D}{\partial x_{offset}}\Big|_{x0}\right) \qquad (10)$$

Οι πρώτης και δεύτερης τάξης παράγωγοι της συνάρτησης *D*, μπορούν να βρεθούν εύκολα παίρνοντας τη διαφορά των γειτονικών *pixel* από κάθε *keypoint*. Στην περίπτωση που το $\hat{x} > 0.5$ για οποιοδήποτε από τα *x, y, σ,* το μέγιστο βρίσκεται σε άλλη θέση. Αλλάζοντας το x_0 ως προς *x, y* ή *σ*, ξανακάνουμε τη διαδικασία για το νέο x_0. Τέλος, για κάθε σημείο θα έχουμε και το x_{offset} που προσθέτοντας στα *x, y, σ* μας δείχνει το πραγματικό μέγιστο. Η διαδικασία για εύρεση του παρεμβαλλόμενου ακροτάτου επαναλαμβάνεται για περιορισμένο αριθμό επαναλήψεων. Αν ο αριθμός ξεπεραστεί και δεν έχει βρεθεί η θέση του ακρότατου, το keypoint κρίνεται ασταθές και απορρίπτεται. Τα ευσταθή *keypoints* που μένουν μετά την εύρεση του παρεμβαλλόμενου ακρότατου φαίνονται στην Εικόνα 3.8.

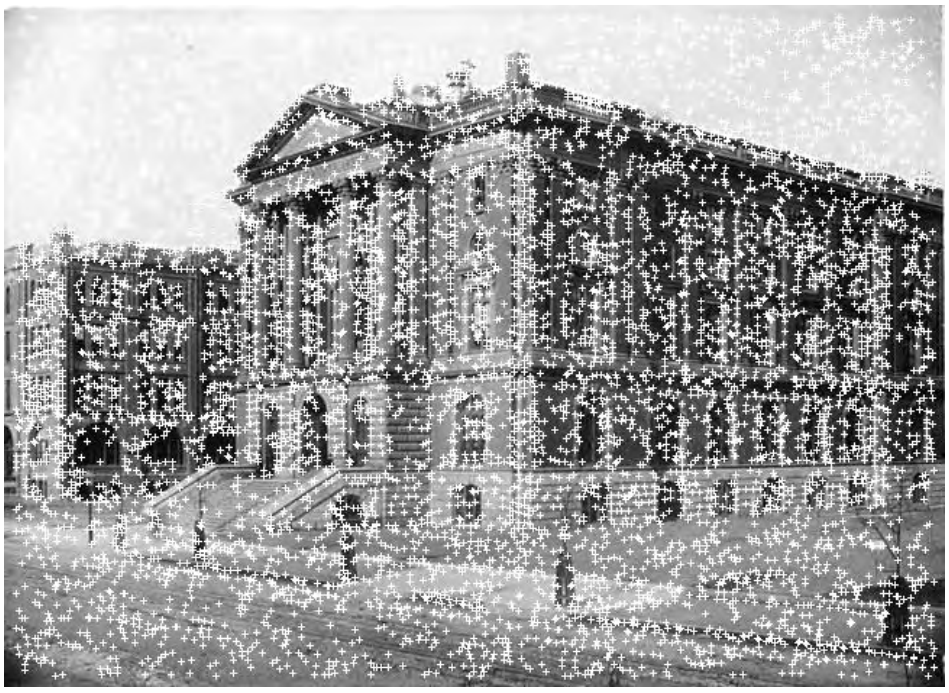
3.2.2 Απόρριψη σημείων χαμηλής αντίθεσης

Όπως αναφέρθηκε παραπάνω, μετά τον υπολογισμό των παρεμβαλλόμενων ακρότατων μπορούμε να απορρίψουμε τα σημεία που παρουσιάζουν χαμηλή αντίθεση. Αντικαθιστώντας, την (10) στην (9) έχουμε:

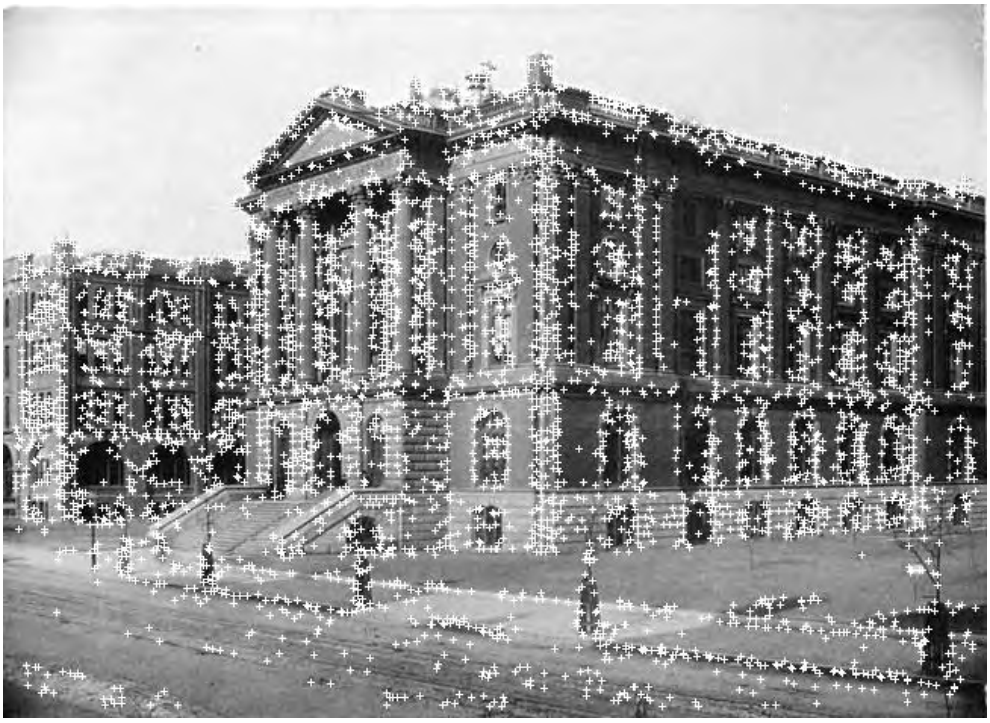
$$D(\hat{x}, \hat{y}, \hat{\sigma}) = D(x_0 + \delta x, y_0 + \delta y, \sigma_0 + \delta\sigma) =$$

$$D(x_0, y_0, \sigma_0) + \frac{1}{2}\left(\frac{\partial D}{\partial x_{offset}}\Big|_{x0}\right)^T (x_0 + \delta x, y_0 + \delta y, \sigma_0 + \delta\sigma).$$

Στο [1] προτείνεται να απορρίπτονται τα ακρότατα με $|D(\hat{x})| < 0.3$ όταν το εύρος έντασης των pixel είναι [0, 1]. Τα ακρότατα που είναι ευσταθή όταν έχουμε χαμηλή αντίθεση παρουσιάζονται στην Εικόνα 3.9 .



Εικόνα 3.8 Ευσταθή keypoints μετά την εύρεση πραγματικού ακρότατου



Εικόνα 3.9 keypoints μετά το threshold χαμηλής αντίθεσης

3.2.3 Απόρριψη ασταθών σημείων που βρίσκονται κατά μήκος των ακμών

Για να πετύχουμε μεγαλύτερη ευστάθεια, εκτός από την απόρριψη των σημείων χαμηλής αντίθεσης απορρίπτουμε και σημεία τα οποία είναι κατά μήκος ακμών. Παρουσία θορύβου τα keypoints που βρίσκονται πάνω σε ακμές γίνονται ασταθή και ως εκ τούτου πρέπει να απορριφθούν. Μία ακμή είναι ένα σύνολο από *pixel* που παρουσιάζουν τον ίδιο προσανατολισμό[17], παρουσία όμως θορύβου οι τιμές των *pixel* αλλάζουν με τυχαίο τρόπο και έτσι η ακμή χάνει την υπόστασή της, γεγονός που δεν επιτρέπει την επαναληψιμότητα του χαρακτηριστικού. Πρώτο βήμα είναι η εύρεση της κύριας καμπυλότητας στην κλίμακα και τη θέση του σημείου, η οποία μπορεί να υπολογιστεί μέσο ενός 2x2 Hessian πίνακα[31], **H**:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Οι παράγωγοι υπολογίζονται παίρνοντας τις διαφορές των γειτονικών *pixel* για κάθε *keypoint*. Οι ιδιοτιμές της **H** είναι ανάλογες της κύριας καμπυλότητας. Για να αποφύγουμε να βρούμε τις ιδιοτιμές, μιας και το μόνο που χρειαζόμαστε είναι ο λόγος τους, μπορούμε να χρησιμοποιήσουμε το ίχνος του πίνακα και την ορίζουσα. Το ίχνος (*trace*) είναι το άθροισμα των διαγώνιων στοιχείων ενός πίνακα. Αν ο πίνακας είναι

τετραγωνικός τότε αποτελεί και το άθροισμα των ιδιοτιμών. Η ορίζουσα ενός πίνακα είναι ίση με το γινόμενο των ιδιοτιμών. Υποθέτοντας ότι α, β είναι οι ιδιοτιμές και α > β έχουμε:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$
$$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta$$

Έστω r = α/β, αντικαθιστώντας στις παραπάνω σχέσεις και διαιρώντας το τετράγωνο του ίχνους με την ορίζουσα έχουμε:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$$

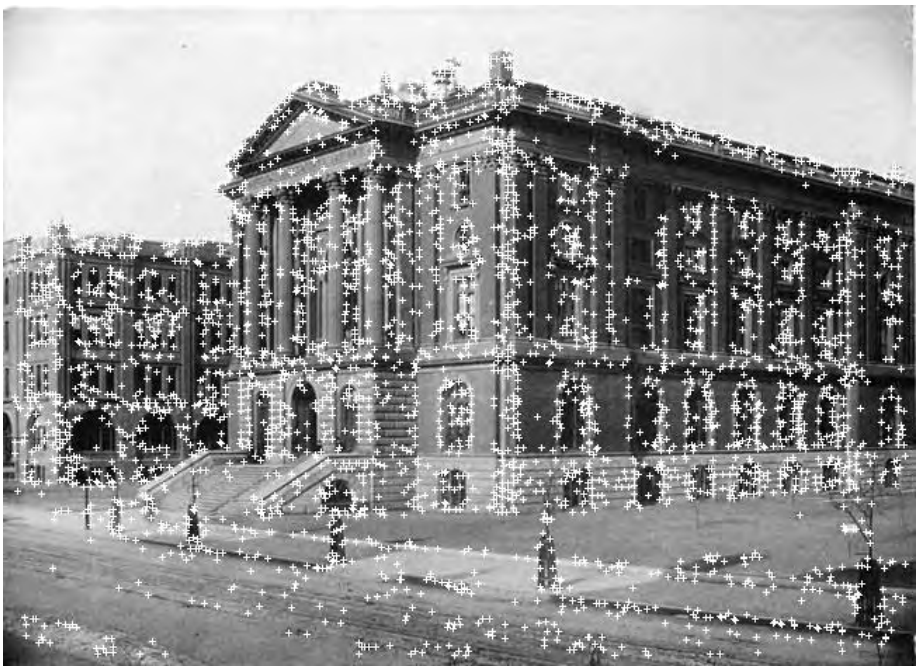
Η παραπάνω συνάρτηση έχει ελάχιστο όταν α = β, ενώ σε διαφορετική περίπτωση αυξάνεται. Θέτοντας μία τιμή στο r διάφορη της μονάδας, η παράσταση $\frac{Tr(H)^2}{Det(H)}$ θα παρουσιάσει ένα άνω όριο:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

Στο [1] προτείνεται r = 10. Στις θέσεις των keypoints που έμειναν μετά από αυτό το βήμα βρίσκονται τα χαρακτηριστικά μας (*features*). Το επόμενο βήμα του αλγορίθμου εκτελείται πάνω σε αυτές τις θέσεις προκειμένου τα χαρακτηριστικά (*features*) να αποκτήσουν ανθεκτικότητα στις περιστροφές. Τα keypoints που έμειναν μετά την εφαρμογή του threshold $\frac{(r+1)^2}{r}$ στην κύρια καμπυλότητα για r = 10, είναι αυτά της Εικόνας 3.10 .

3.3 Orientation Assignment

Με τα παραπάνω βήματα έχουμε πετύχει την ευστάθεια των χαρακτηριστικών μας σε αλλαγές μεγέθους της εικόνας και κατά την παρουσία θορύβου. Επόμενο βήμα είναι τα χαρακτηριστικά να δείχνουν ευστάθεια κατά την περιστροφή. Για να επιτευχτεί αυτό αναπαριστούμε τα χαρακτηριστικά σε σχέση με τον κύριο προσανατολισμό που εμφανίζει η περιοχή γύρω από τα keypoints.



Εικόνα 3.10 keypoints μετά την εφαρμογή threshold στην κύρια καμπυλότητα

3.3.1 Υπολογισμός του μέτρου και της γωνίας κλίσης για τα pixel γύρω από το keypoint

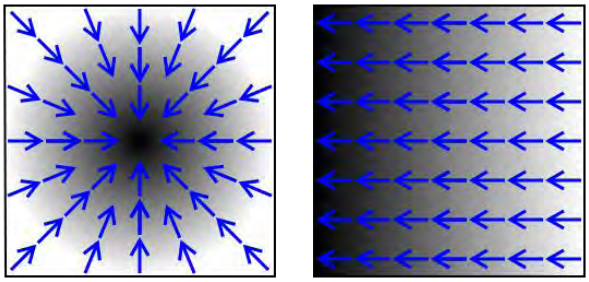
Για κάθε *keypoint* έχουμε αποθηκεύσει την κλίμακα στην οποία βρέθηκε. Από την κλίμακα (*scale*) του *keypoint* μπορούμε να βρούμε την πιο κοντινή *L(x,y) Gaussian Blurred* εικόνα. Στη συνέχεια, ορίζουμε μια περιοχή γύρω από το *keypoint* όπου θα γίνει η επεξεργασία. Για τα παραπάνω *pixel* βρίσκουμε τα ακόλουθα μεγέθη:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + ((L(x,y+1) - L(x,y-1))^2}$$

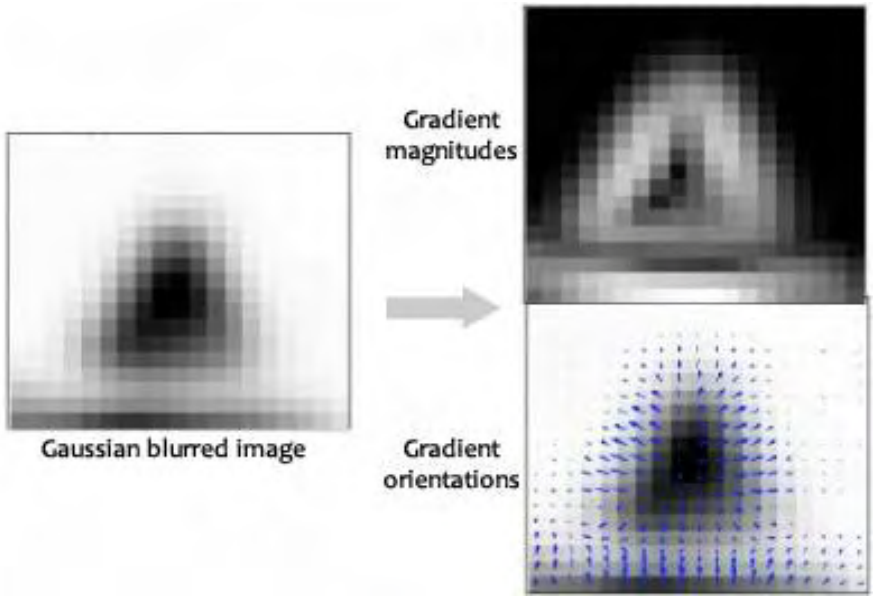
$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Όπου $m(x,y)$ *και* $\theta(x,y)$, το μέτρο και η γωνία της κλίσης φωτεινότητας των *pixel*[11], αντίστοιχα. Η ιδέα είναι να συλλέξουμε τα $m(x,y)$ *και* $\theta(x,y)$ των pixel της περιοχής και να βρούμε τον κύριο προσανατολισμό τους. Στους παραπάνω τύπους το L αναφέρεται στη φωτεινότητα που έχει το *pixel* στη συγκεκριμένη θέση. Στην Εικόνα 3.11 φαίνεται ένα παράδειγμα της κλίσης φωτεινότητας μία περιοχής. Στη συνέχεια, τα υπόλοιπα βήματα γίνονται με βάση τον κύριο προσανατολισμό και έτσι πετυχαίνουμε ευστάθεια των χαρακτηριστικών σε περιπτώσεις που έχουμε περιστροφή της εικόνας.

Ένα παράδειγμα της μορφής που έχει η περιοχή για το μέτρο και την κλίση παρουσιάζεται στην Εικόνα 3.12.



Εικόνα 3.11



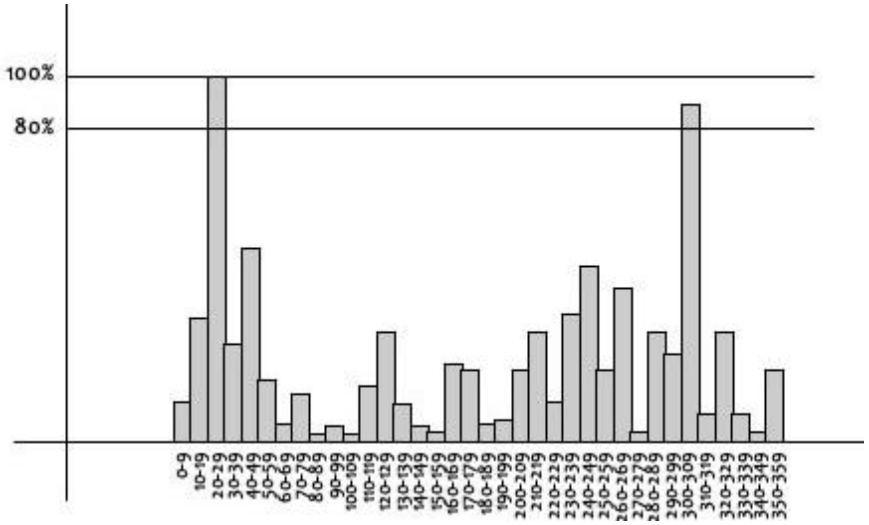
Εικόνα 3.12

3.3.2 Δημιουργία ιστογράμματος με τα δεδομένα του προηγούμενου βήματος

Με βάση τα παραπάνω, δημιουργούμε ένα ιστόγραμμα το οποίο έχει 36 θέσεις αποθήκευσης. Κάθε θέση αντιπροσωπεύει ένα εύρος 10 μοιρών, για παράδειγμα η θέση 1→ [10,19], η θέση 3 → [30,39]. Έτσι καταλαβαίνουμε ότι με 36 θέσεις καλύπτουμε το εύρος των 360 μοιρών, που μπορεί να μας επιστρέφει κάθε φορά η $\theta(x, y)$. Στη συνέχεια, δημιουργούμε ένα *Gaussian kernel*, ο οποίος έχει μέγεθος ίσο με την περιοχή που έχουμε

ορίσει την επεξεργασία και σ που είναι 1.5 φορές η κλίμακα του keypoint. Ο πολλαπλασιασμός κάθε $m(x,y)$ με μία τιμή *Gaussian kernel* γίνεται για να κάνουμε ένα σταθμισμένο (*weighted*) *average* τις τιμές των γωνιών, δίνοντας μεγαλύτερο βάρος στις θέσεις που είναι κοντά στο keypoints και λιγότερο σε αυτές που είναι πιο μακριά. Για καλύτερη κατανόηση παραθέτουμε ένα παράδειγμα της διαδικασίας.

Αν (x, y) = (45, 76) τότε το pixel στη θέση (x-9, y-9) θα είναι (36, 67). Αν για παράδειγμα έχει γωνία κλίσης $\theta(36{,}67) = 25$, τότε προσθέτουμε στη δεύτερη θέση του ιστογράμματος το $m(x,y)$ πολλαπλασιασμένο με την τιμή που συμπίπτει με το συγκεκριμένο pixel, αν κεντράρουμε τον *Gaussian kernel* στο κεντρικό (x, y) = (45, 76). Ένα παράδειγμα του ιστογράμματος που παράγεται, φαίνεται στην Εικόνα 3.13.

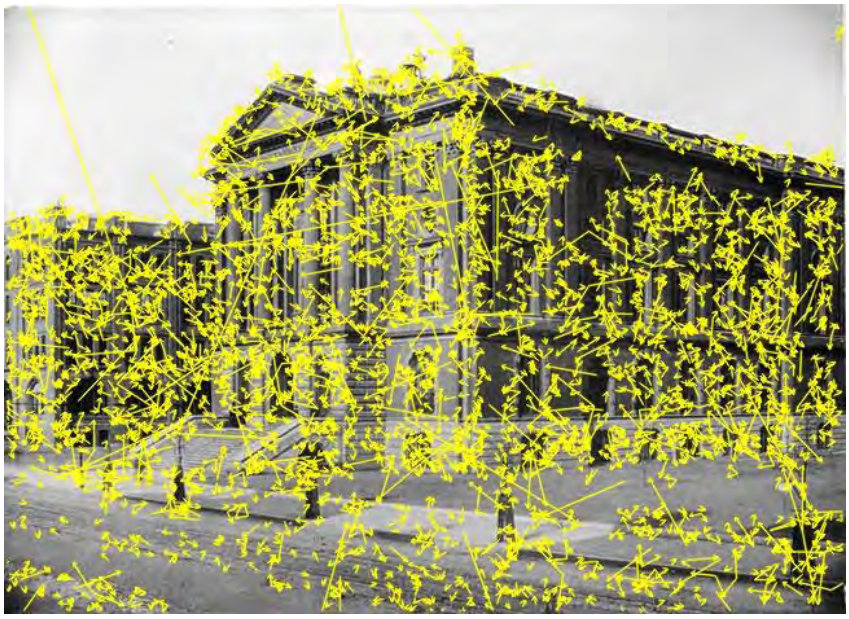


Εικόνα 3.13 Ο άξονας y μας δείχνει το άθροισμα των μέτρων τα οποία έχουν επιβαρυνθεί με τις τιμές του Gaussian kernel και ο άξονας x το εύρος γονιών για κάθε θέση αποθήκευσης.

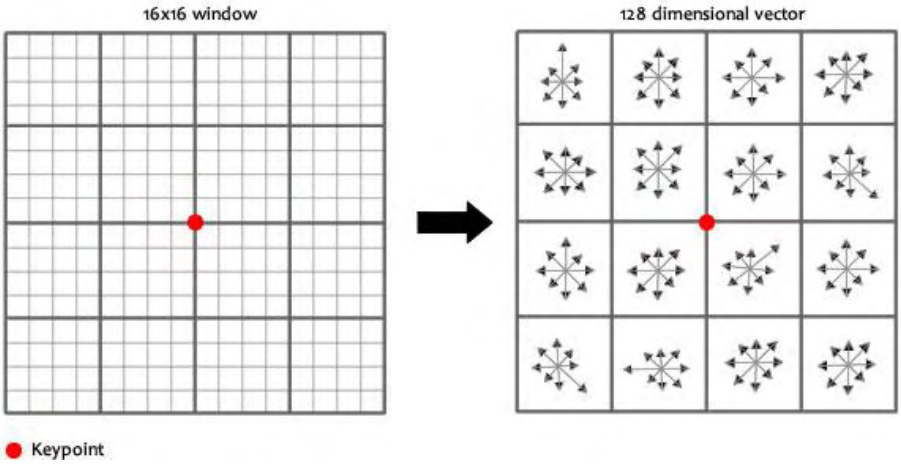
3.3.3 Κυρίαρχος προσανατολισμός

Μετά τον υπολογισμό του ιστογράμματος, αναζητείται σε αυτό η μέγιστη τιμή του καθώς και οποιαδήποτε άλλη τιμή είναι πάνω από το 80% της μέγιστης. Στη συνέχεια, γίνεται η προσαρμογή μιας παραβολής στα σημεία που ορίζονται από κάθε μέγιστο και των σημείων των προσκείμενων θέσεων, ώστε να έχουμε καλύτερη ακρίβεια για την τιμή του μεγίστου. Για κάθε τιμή που βρέθηκε από το ιστόγραμμα δημιουργείται ένα χαρακτηριστικό (*feature*), το οποίο έχει κύριο προσανατολισμό την παρεμβαλλόμενη τιμή της γωνίας-θέσης. Αυτό το βήμα είναι πολύ σημαντικό, καθώς ο περιγραφέας (descriptor) κάθε χαρακτηριστικού (*feature*) υπολογίζεται με βάση τον κύριο προσανατολισμό. Δηλαδή, αν αρχικά έχουμε για το χαρακτηριστικό μας (*feature*) ένα κύριο προσανατολισμό θ μοίρες και μετά περιστρέψουμε την εικόνα κατά θ_i,

περιμένουμε το νέο χαρακτηριστικό (*feature*) να έχει κύριο προσανατολισμό $\theta + \theta_i$. Έτσι, ο περιγραφέας θα είναι ο ίδιος αφού τα pixel έχουν και αυτά μετατοπιστεί κατά θ_i. Οι κυρίαρχοι προσανατολισμοί φαίνονται στην Εικόνα 3.14 .



Εικόνα 3.14



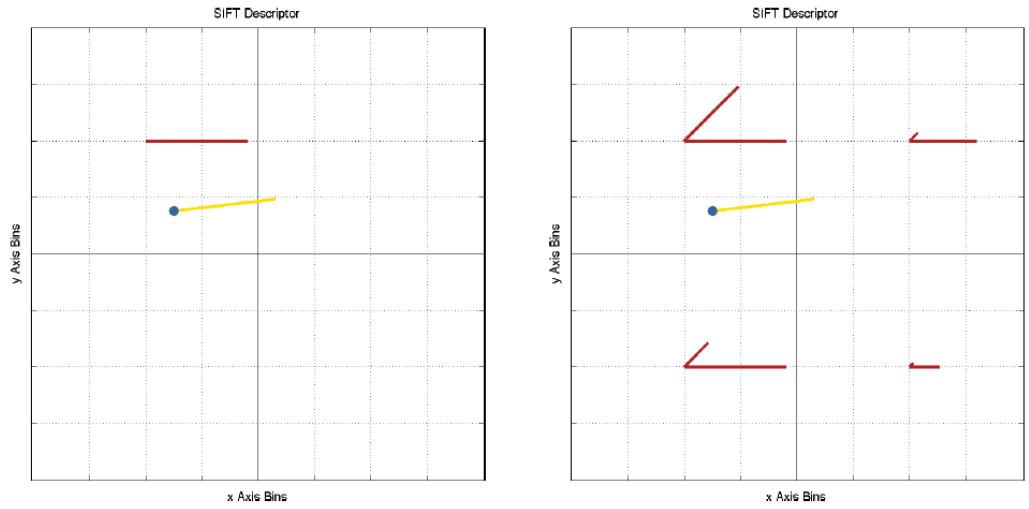
Εικόνα 3.15

3.4 Keypoint Descriptor

Ο υπολογισμός του περιγραφέα είναι το τελευταίο και το κυριότερο βήμα του αλγορίθμου. Μέχρι στιγμής, από τα τρία πρώτα βήματα έχουμε πετύχει ευστάθεια σε μεταβολές κλίμακας και περιστροφής. Στο βήμα αυτό φτιάχνουμε ένα περιγραφέα, ο οποίος θα είναι μοναδικός για κάθε χαρακτηριστικό (*feature*), έτσι ώστε σε μετέπειτα επεξεργασία αν έχουμε δύο περιγραφείς στην ίδια εικόνα να γνωρίζουμε ότι αναφέρονται σε κάτι εντελώς διαφορετικό. Η μορφή του περιγραφέα φαίνεται στην Εικόνα 3.15.

3.4.1 Δημιουργία SIFT Descriptor

Όπως φαίνεται στην Εικόνα 3.15 ορίζουμε μία περιοχή 128 (16x16) pixel γύρω από το keypoint, η οποία με τη σειρά της χωρίζεται σε υποπεριοχές (*block*s) των 16 (4x4) pixel. Για τα 16 pixel κάθε υποπεριοχής υπολογίζονται, με τους τύπους από το τρίτο βήμα, τα μεγέθη $m(x,y)$ *και* $\theta(x,y)$. Για κάθε υποπεριοχή (*block*) δημιουργείται ένα ιστόγραμμα 8 θέσεων, που όπως προηγουμένως κάθε θέση αντιπροσωπεύει ένα εύρος γωνιών. Η διαφορά είναι ότι τώρα το εύρος για κάθε θέση είναι μεγαλύτερο και καλύπτει 45 μοίρες και όχι 10, όπως στο τρίτο βήμα. Για παράδειγμα, η θέση 1→ [45,89], η θέση 2 → [90,134], ουσιαστικά χωρίζουμε τον κύκλο σε ογδοημόρια. Για να δώσουμε λιγότερη έμφαση στα *pixel* που είναι πιο μακριά από το κέντρο του περιγραφέα, κάθε μέτρο που μπαίνει στο ιστόγραμμα πολλαπλασιάζεται με ένα συντελεστή που προέρχεται από ένα *Gaussian kernel*, ο οποίος είναι κεντραρισμένος στη μέση των 128 *pixel* (το κόκκινο σημείο στην Εικόνα 3.15). Για να αποφύγουμε τις μεταβολές του περιγραφέα και να κάνουμε πιο ομαλή την κατανομή των δεδομένων στα ιστογράμματα χρησιμοποιούμε 3D-παρεμβολή. Έτσι, κάθε $m(x,y)$ που εισάγεται σε μία θέση του ιστογράμματος πολλαπλασιάζεται εκτός από το συντελεστή του *Gaussian kernel* και με ένα βάρος, που σχετίζεται με την απόσταση που έχει η $\theta(x,y)$ σε σχέση με το κέντρο των προσκείμενων θέσεων στο ιστόγραμμα. Αν για παράδειγμα, $\theta(x,y) = 78$ μοίρες, η απόσταση από τα κέντρα των προσκείμενων θέσεων, δηλαδή, θέση 1→ [45,89] *και θέση* 2 → [90,134], είναι $|d_1| = \left|\frac{45+89}{2} - 78\right| = 11, |d_2| = \left|\frac{90+134}{2} - 78\right| = 34$ τα οποία τα κανονικοποιούμε ως προς το εύρος κάθε θέσης (45 μοίρες). Έτσι, το $m(x,y) \times G \times d_1/45$ προστίθεται στη θέση 1 και το $m(x,y) \times G \times d_2/45$ στη θέση 2 του ιστογράμματος. Επίσης, μπορεί να γίνει και παρεμβολή σχετικά με τα γειτονικά ιστογράμματα. Στην Εικόνα 3.16 παρουσιάζεται ένα παράδειγμα.



(a) Descriptor without Interpolation (b) Descriptor with Interpolation

Εικόνα 3.16 Στην εικόνα παρουσιάζεται η μορφή ορισμένων υποπεριοχών του περιγραφέα με και χωρίς παρεμβολή

Όπως αναφέρθηκε παραπάνω, για να έχουμε ευστάθεια κατά τις περιστροφές ο περιγραφέας υπολογίζεται με βάση τον κύριο προσανατολισμό. Από τις γωνίες κλίσης $\theta(x, y)$ των pixel της περιοχής αφαιρείται ο κύριος προσανατολισμός (πριν μπουν στα ιστογράμματα) έτσι ώστε, όλοι οι προσανατολισμοί να είναι σχετικοί με τον κύριο του χαρακτηριστικού (*feature*). Ο περιγραφέας ορίζεται ως το διάνυσμα που περιέχει όλες τις τιμές των 16 ιστογραμμάτων. Αφού έχουμε 8 θέσεις για κάθε ιστόγραμμα το διάνυσμα θα έχει διάσταση 128 στοιχείων.

3.4.2 Ανθεκτικότητα Descriptor σε αλλαγές φωτισμού

Για να κάνουμε τον περιγραφέα να παρουσιάζει ευστάθεια και για γραμμικές αλλαγές φωτισμού, το διάνυσμα του περιγραφέα κανονικοποιείται στη μονάδα. Αυτό γίνεται διαιρώντας κάθε στοιχείο του διανύσματος με το άθροισμα όλων των στοιχείων αυτού. Επίσης, για να έχουμε ευστάθεια και σε μη-γραμμικές αλλαγές φωτισμού, κάθε τιμή του διανύσματος που ξεπερνά το 0.2 τη θέτουμε ίση με αυτό και κανονικοποιούμε ξανά το διάνυσμα.

Τα αποτελέσματα που αποθηκεύονται για κάθε εικόνα μετά την εφαρμογή του SIFT είναι το όνομα της, ο αριθμός των χαρακτηριστικών της, ο τύπος της και τα χαρακτηριστικά διανύσματα που την προσδιορίζουν. Με τον όρο «τύπο», αναφερόμαστε το περιεχόμενο της εικόνας π. χ αυτοκίνητο, άνθρωπος, ποδήλατο..κα. Η δομή κάθε χαρακτηριστικού αποτελείται από τη θέση του κέντρου του (x, y) που βρίσκεται πάνω στην εικόνα, τον κυρίαρχο προσανατολισμό του, την κλίμακα στην οποία βρέθηκε (*scale*) και το χαρακτηριστικό του διάνυσμα.

3.5 Υλοποίηση SIFT σε γλώσσα προγραμματισμού C

Για να διαβάσουμε τις τιμές των *pixel* των εικόνων χρησιμοποιήθηκε η βιβλιοθήκη tiff.h [32]. Στη συνέχεια, έγινε η μετατροπή των έγχρωμων εικόνων σε *gray-level* χρησιμοποιώντας τη φωτομετρική μέθοδο (*luminance-preserving*)[33], η οποία είναι βασισμένη στον τρόπο που το ανθρώπινο μάτι μπορεί να αντιληφθεί τα χρώματα. Το ανθρώπινο μάτι είναι πιο ευαίσθητο στο πράσινο χρώμα και έτσι αυτό σταθμίζεται με μεγαλύτερο συντελεστή[17]. Ο τύπος που χρησιμοποιήσαμε για τη φωτεινότητα ήταν 0.21 R + 0.72 G + 0.07 B.

Για να δημιουργήσουμε την πυραμίδα με τις *Gaussian Blurred* εικόνες χρησιμοποιήσαμε ένα *Gaussian Kernel* 13x13. Ο λόγος ήταν ότι για μικρότερες διαστάσεις τα *keypoints* που εντοπίζαμε ήταν πολύ λίγα, ενώ για μεγαλύτερες από 13x13 υπήρχε ελάχιστη αύξηση του αριθμού τους. Επίσης, χρησιμοποιώντας *kernel* μεγαλύτερων διαστάσεων αυξάνουμε τους υπολογισμούς που πρέπει να γίνουν για να θολώσουμε την εικόνα.

Για να υπολογίσουμε τον Hessian matrix χρησιμοποιήσαμε τις εξισώσεις:

$$D_{xx} = \frac{\partial^2 D}{\partial x^2} = (D(x+1,y) + D(x-1,y) - 2 \times D(x,y))$$

$$D_{yy} = \frac{\partial^2 D}{\partial y^2} = (D(x,y+1) + D(x,y-1) - 2 \times D(x,y))$$

$$D_{xy} = \frac{\partial^2 D}{\partial x \partial y} = (D(x+1,y+1) + D(x-1,y+1) + D(x+1,y-1) + D(x-1,y-1) - 4 * D(x,y)$$

Με *D* την τιμή του *pixel*, της *DoG*, στη συγκεκριμένη θέση γύρω από το *keypoint*.

Για να βρούμε τον προσανατολισμό του χαρακτηριστικού ορίσαμε μία περιοχή ± 9 *pixel* γύρω από το keypoint και υπολογίσαμε τα μεγέθη m(x,y), θ(x,y) με τους τύπους που δίνονται στην Ενότητα 3.3.1.

Στα υπόλοιπα μέρη του αλγορίθμου χρησιμοποιήσαμε τις παραμέτρους που προτείνονται στο [1]. Τα αποτελέσματα συγκρίθηκαν με αυτά που δίνει το SIFT demo program[34]. Η θέση και ο προσανατολισμός των χαρακτηριστικών επαληθεύτηκαν κατά ένα ποσοστό ~97%. Το διάνυσμα του περιγραφέα είχε μικρότερη ακρίβεια. Λόγω έλλειψης χρόνου για περαιτέρω πειράματα, για τα αποτελέσματα που παρουσιάζονται στο Κεφάλαιο 6 χρησιμοποιήθηκαν τα χαρακτηριστικά όπως αυτά δίνονται από το SIFT demo program[34].

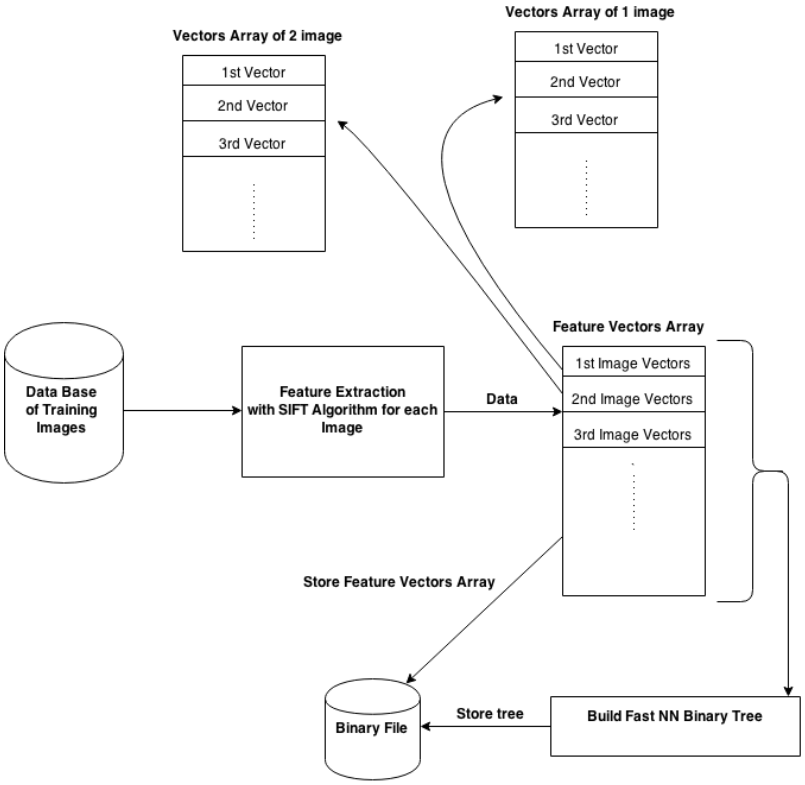
Αξίζει να σημειωθεί ότι πέρα από τις διαφορές στο θέμα ακρίβειας, πετύχαμε μία μείωση του χρόνου υπολογισμού των χαρακτηριστικών της τάξεως του ~15%.

Κεφάλαιο 4

Αντιστοίχιση Χαρακτηριστικών με Fast NN

Όπως έχει αναφερθεί παραπάνω, στόχος της εργασίας είναι η ανάκτηση εικόνων από μία βάση δεδομένων. Στο προηγούμενο κεφάλαιο παρουσιάστηκε ο αλγόριθμος SIFT, ο οποίος χρησιμοποιήθηκε για να γίνει η ανίχνευση και η εξαγωγή των χαρακτηριστικών από τις εικόνες. Πιο συγκεκριμένα, κάθε εικόνα μετά την εφαρμογή του SIFT αποτελείται από έναν αριθμό διανυσμάτων (*128-διαστάσεων*) τα οποία τη χαρακτηρίζουν. Το πρώτο στάδιο, για να γίνει η ανάκτηση μίας *query* εικόνας, είναι η ανεξάρτητη αντιστοιχία κάθε χαρακτηριστικού διανύσματος της εικόνας με τα διανύσματα που έχουν εξαχθεί από τις εικόνες που βρίσκονται στη βάση δεδομένων (*training images*).

Το παρόν κεφάλαιο παρουσιάζει τις μεθόδους που χρησιμοποιήθηκαν, με σκοπό την αναζήτηση και την αντιστοίχιση των χαρακτηριστικών διανυσμάτων. Όπως αναφέρεται στο κεφάλαιο 2.3.1, η αντιστοίχιση δύο διανυσμάτων ορίζεται ως το πρόβλημα του Κοντινότερου Γείτονα (NN). Ο αλγόριθμος που χρησιμοποιήσαμε για την αναζήτηση NN είναι ο Fast NN [2, 3].



Διάγραμμα 4.1 Αποθήκευση χαρακτηριστικών και δέντρου αναζήτησης

4.1 Αρχικοποίηση Δέντρου Αναζήτησης

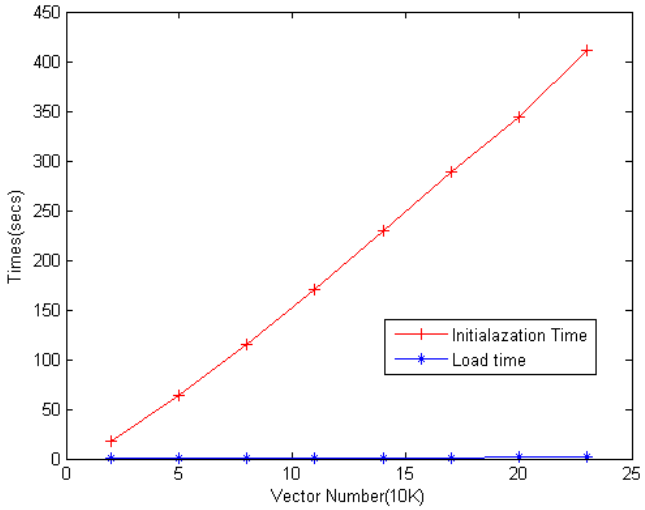
Αρχικά, για να μπορέσει να πραγματοποιηθεί η αναζήτηση και η αντιστοίχιση των χαρακτηριστικών διανυσμάτων πρέπει να δημιουργηθεί μία βάση δεδομένων από τα χαρακτηριστικά διανύσματα των training εικόνων. Στο Διάγραμμα 4.1 παρουσιάζεται η διαδικασία που ακολουθείται για το σκοπό αυτό. Μετά την εξαγωγή τους, τα διανύσματα των training εικόνων αποθηκεύονται σε έναν πίνακα λαμβάνοντας το καθένα ένα μοναδικό κλειδί (*vector_id*). Στη συνέχεια, ο πίνακας εισάγεται σε μία συνάρτηση η οποία αρχικοποιεί ένα δυαδικό δέντρο με τον εξής τρόπο:

α. Ο πίνακας με τα διανύσματα αποθηκεύεται στη ρίζα του δυαδικού δέντρου.

β. Στα δεδομένα εφαρμόζεται ο αλγόριθμος k-Means [4], ο οποίος τα χωρίζει σε δύο υποσύνολα Q_1 και Q_2. Στη συνέχεια, υπολογίζεται το μέγιστο υπερ-επίπεδο $H(\mathscr{w})$ που χωρίζει τα κέντρα των Q_1, Q_2 , όπου το $\mathscr{w}$ είναι το διάνυσμα που ορίζει το $H(\mathscr{w})$. Το $\mathscr{w}$ είναι το διάνυσμα το οποίο είναι κάθετο στο διάνυσμα που ενώνει τα κέντρα των Q_1 και Q_2. Στο δέντρο δημιουργούνται δύο παιδιά το Q_1 και το Q_2, ένα αριστερό και ένα δεξί. Υποθέτοντας ότι Q_1 είναι το αριστερό παιδί, για να περιέχεται ένα διάνυσμα $\mathscr{x}$ στο Q_1 θα πρέπει εξ ορισμού να ισχύει $d_s[\mathscr{x}, H(\mathscr{w})] < 0$, όπου $d_s[\mathscr{x}, H(\mathscr{w})]$ είναι το εσωτερικό γινόμενο των $\mathscr{x}$ και $\mathscr{w}$ και ορίζει την Ευκλείδεια απόσταση του x από το $H(\mathscr{w})$, με $\mathscr{x}$ ένα επαυξημένο διάνυσμα $\mathscr{x} = [1, x_1, \dots, x_n]$. Έτσι, τα δεδομένα ανάλογα με την απόστασή τους ταξινομούνται δεξιά ή αριστερά.

γ. Το δεύτερο βήμα επαναλαμβάνεται μέχρι τα φύλλα του δέντρο να περιέχουν ένα μόνο διάνυσμα.

Αυξάνοντας τις εικόνες σε μία βάση δεδομένων έχει ως αποτέλεσμα να αυξάνονται και τα διανύσματα που πρέπει να αποθηκευτούν στο δένδρο. Η παραπάνω αρχικοποίηση είναι μία χρονοβόρα διαδικασία για μεγάλο αριθμό εικόνων. Έτσι, για λόγους απόδοσης, αποφεύγουμε την αρχικοποίηση κάθε φορά που κάνουμε μία αναζήτηση. Μετά την πρώτη αρχικοποίηση το δέντρο αποθηκεύεται σε ένα δυαδικό αρχείο και ανακτάται όταν θέλουμε να αναζητήσουμε μία εικόνα στη συγκεκριμένη βάση δεδομένων. Η διαφορά των χρόνων ανάγνωσης και αρχικοποίησης φαίνεται στην Εικόνα 4.2 Tree_build_times.



Εικόνα 4.2 Tree build times.

4.2 Αναζήτηση δέντρου για εύρεση Κοντινότερου Γείτονα

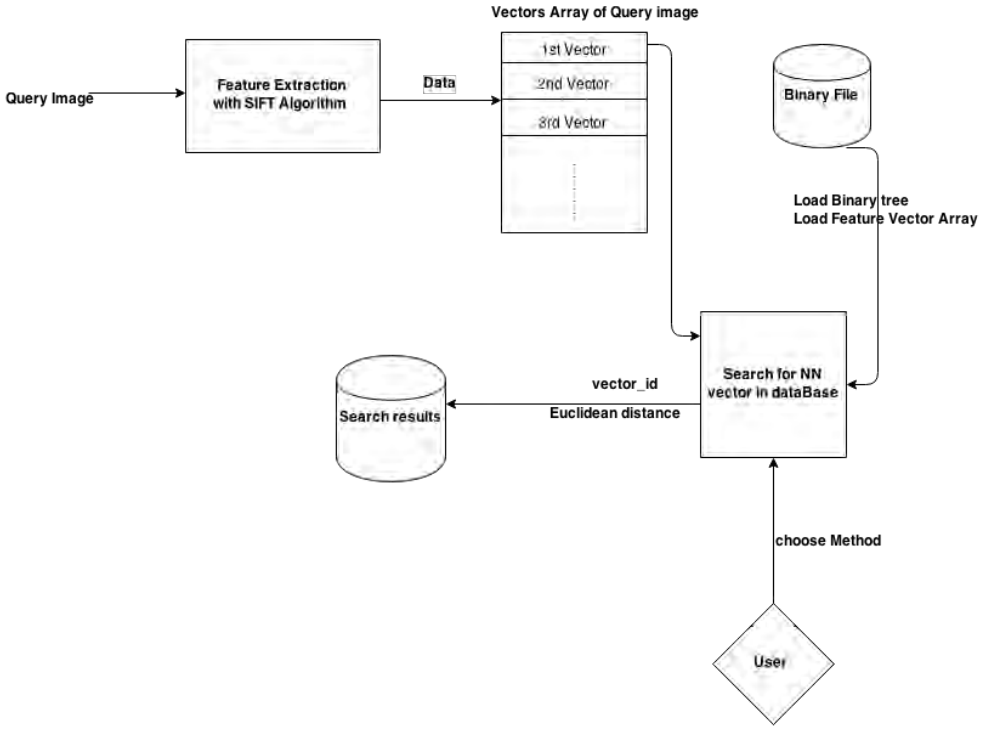
Μετά το τέλος της αρχικοποίησης μπορούμε να αναζητήσουμε χαρακτηριστικά διανύσματα στη βάση δεδομένων. Στο χρήστη δίνεται η δυνατότητα επιλογής της μεθόδου που θέλει να εκτελεστεί κατά την αναζήτηση. Παρακάτω παρουσιάζονται οι μέθοδοι αναζήτησης, με τα πλεονεκτήματα και μειονεκτήματα που έχει η κάθε μία. Οι μέθοδοι που αναφέρονται είναι παραλλαγές του αλγορίθμου Fast NN. Τα αποτελέσματα συγκρίθηκαν με τον αλγόριθμο Full NN (*Full-Nearest-Neighbor*) ο οποίος εξετάζει όλα τα διανύσματα της βάσης δεδομένων έτσι ώστε να βρει τον NN.

Έστω ότι έχουμε μία εικόνα Q, η οποία χαρακτηρίζεται από ένα αριθμό q διανυσμάτων. Δεδομένου ενός διανύσματος q_i, γίνεται μια αναζήτηση στη βάση δεδομένων για να βρεθεί ο NN του q_i. Κάθε μέθοδος δέχεται σαν είσοδο ένα διάνυσμαq, τον πίνακα διανυσμάτων των training εικόνων και το δυαδικό δέντρο αναζήτησης. Μετά το τέλος μίας αναζήτησης επιστρέφεται το *vector_id* του διανύσματος x που βρέθηκε να είναι πιο κοντινό στο q_i καθώς και η Ευκλείδεια απόσταση του διανύσματος q_i από το x , $d[q_i, x]$. Η αναζήτηση συνεχίζεται και για τα υπόλοιπα διανύσματα της Q εικόνας. Η διαδικασία φαίνεται στο Διάγραμμα 4.2 .

a) Depth Only Search (DOS) Fast NN

Η DOS είναι μία «*greedy*» μέθοδος, η οποία κάνει μία αναζήτηση από τη ρίζα του δέντρου μέχρι να φτάσει σε φύλλο και τερματίζεται. Κατά την αναζήτηση υπολογίζεται κάθε φορά η προσημασμένη απόσταση του q_i και του υπερ-επιπέδου $H(w)$ που σχετίζεται με τον τρέχοντα κόμβο. Αν $d_s[q_i, H(w)] < 0$ τότε προχωράμε την αναζήτηση

στον αριστερό κόμβο, ενώ σε διαφορετική περίπτωση προχωράμε στο δεξί. Η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε σε ένα φύλλο και μετά τερματίζει.



Διάγραμμα 4.2

b) Complete Search Fast NN

Η μέθοδος Complete Fast NN είναι μία επέκταση της DOS. Αρχικά, γίνεται μία αναζήτηση με τη DOS έως ότου φτάσουμε σε ένα φύλλο του δέντρου. Στη συνέχεια, αρχίζει μία οπισθοδρόμηση (*backtrackings*) στους προηγούμενους κόμβους, έτσι ώστε να εξεταστούν και τα υπόλοιπα διανύσματα-φύλλα του δέντρου. Για αποδοτικούς λόγους αποφεύγουμε να εξετάσουμε όλα τα διανύσματα και έτσι η αναζήτηση βασίζεται σε ένα κάτω όριο Ευκλείδειας απόστασης $d[q,x] \geq LB$ με $LB = |d_s[q,H(w)] - d_s[x,H(w)]|$. Κατά την αναζήτηση μέχρι να φτάσουμε σε φύλλο οι προσημασμένες αποστάσεις έχουν υπολογιστεί και με μία μόνο αφαίρεση μπορούμε να υπολογίσουμε την LB. Η διαδικασία συνεχίζεται μέχρι να ισχύει $d[q,x] < LB$ για όλα τα x διανύσματα, γεγονός που σημαίνει ότι βρήκαμε το κοντινότερο διάνυσμα.

c) Limited Search Fast NN

Η Limited Fast NN είναι μία παραλλαγή της Complete Fast NN. Η τελευταία, τερματίζει μία αναζήτηση μετά από Β αριθμό οπισθοδρομικών βημάτων, οπού Β είναι οι οπισθοδρομήσεις μέχρι $d[q,x] < LB$ για όλα τα x διανύσματα. Στη Limited Fast NN μπορούμε να θέσουμε ακριβώς τον αριθμό Β και η αναζήτηση να σταματήσει όταν αυτός ξεπεραστεί. Καθώς ο αριθμός των οπισθοδρομήσεων μεγαλώνει η πιθανότητα να έχουμε βρει τον Κοντινότερο Γείτονα αυξάνεται. Έτσι, μπορούμε να υποθέσουμε ότι μετά από Χ βήματα η αναζήτηση μπορεί να σταματήσει. Όταν ο αριθμός των Χ βημάτων ξεπεραστεί η αναζήτηση τερματίζει.

Από τα παραπάνω, είναι εύκολο να καταλάβουμε ότι η DOS μέθοδος είναι η πιο γρήγορη από τις τρεις που αναφέρθηκαν, αφού αποτελεί το πρώτο στάδιο που κάνουν οι άλλες δύο για να ξεκινήσουν την αναζήτηση. Συγκρίνοντας τους χρόνους που χρειάζεται η κάθε μία για να αναζητήσει ένα q διάνυσμα, 128-διαστάσεων, βλέπουμε ότι η DOS είναι ~3K φορές πιο γρήγορη από τη Complete, ενώ η σχέση της με τη Limited εξαρτάται από τον αριθμό των *backtrackings* που ορίζουμε κάθε φορά. Για παράδειγμα, για 500 *backtrackings* η DOS είναι ~18 φορές πιο γρήγορη ενώ για 1500 *backtrackings* είναι ~47. Συγκρίνοντάς τη, τέλος, με μία Full NN αναζήτηση βλέπουμε ότι η DOS είναι ~11K φορές πιο γρήγορη. Το μειονέκτημά της είναι ότι τερματίζει στο πρώτο φύλλο που θα φτάσει, γεγονός που δε μας εγγυάται ότι βρήκε τον Κοντινότερο Γείτονα του q. Δεδομένου ότι η Complete Fast NN και ο Full NN βρίσκουν τον ακριβή Κοντινότερο Γείτονα 100% και η DOS τον βρίσκει περίπου τις μισές φορές, παρουσιάζεται ένα πρόβλημα ακρίβειας.

Η Complete Fast NN μέθοδος, όπως αναφέρθηκε παραπάνω, είναι πιο αργή από τη DOS και τη Limited, αλλά έχει πλεονέκτημα στην ακρίβεια εύρεσης του Κοντινότερου Γείτονα σε σχέση με τις άλλες, αφού τον βρίσκει 100%. Τα αποτελέσματα έδειξαν ότι είναι ~3 φορές πιο γρήγορη από τον Full NN.

Τέλος, η Limited προσφέρει έναν καλό λόγο ακρίβειας απόδοσης, καθώς είναι πιο ακριβής από τη DOS και πιο γρήγορη από τη Complete. Τα πειράματα έδειξαν ότι η ακρίβεια της στην εύρεση του Κοντινότερου Γείτονα είναι ~70% σε σχέση με τη Complete, ενώ είναι περίπου 600 φορές πιο γρήγορη για 500 *backtrackings*. Η σχέση της Limited με τις άλλες μεθόδους εξαρτάται εξολοκλήρου από τον αριθμό των *backtrackings* που ορίζουμε.

Αφού η αναζήτηση για τα διανύσματα της εικόνας Q ολοκληρωθεί με μία από τις παραπάνω μεθόδους, τα αποτελέσματα αξιοποιούνται έτσι ώστε να επιλεγεί η κατάλληλη εικόνα και να επιστραφεί στο χρήστη.

Κεφάλαιο 5

Ανάκτηση Εικόνας (Image Retrieval)

Όπως φαίνεται στο γενικό διάγραμμα του συστήματος ανάκτησης εικόνας [Διάγραμμα 1.1], το τελικό στάδιο της διαδικασίας είναι η επιστροφή της εικόνας ή των εικόνων στο χρήστη. Στα προηγούμενα κεφάλαια, παρουσιάσαμε τον αλγόριθμο που χρησιμοποιήσαμε για την εξαγωγή των χαρακτηριστικών (*Feature Extraction*) από τις εικόνες καθώς και τις μεθόδους αναζήτησης που χρησιμοποιήσαμε για να βρούμε τους Κοντινότερους Γείτονες των χαρακτηριστικών διανυσμάτων (*Feature Matching*). Στο παρόν κεφάλαιο, αναλύεται η διαδικασία που ακολουθήσαμε για να αξιολογήσουμε τα δεδομένα που μας έδωσε η αναζήτηση των χαρακτηριστικών διανυσμάτων. Μετά την αξιολόγηση θα μπορέσουμε να αποφασίσουμε ποια ή ποιες εικόνες θέλουμε να ανακτήσουμε.

5.1 Database Indexing

Για τις ανάγκες της παρούσας εργασίας δημιουργήθηκε μια βάση δεδομένων από εικόνες [12] και σε κάθε μια εφαρμόστηκε ο αλγόριθμος SIFT, με σκοπό να γίνει η εξαγωγή και η ανίχνευση των χαρακτηριστικών τους.

Στο σύστημα που υλοποιήσαμε η ανάκτηση μίας εικόνας γίνεται μέσω μιας διαδικασίας ψηφοφορίας. Μία εικόνα Q, που εισάγεται στο σύστημα προς αναζήτηση, έχει στη διάθεσή της έναν αριθμό από «ψήφους». Κάθε ψήφος αντιστοιχεί σε ένα χαρακτηριστικό διάνυσμα q της εικόνας. Έτσι, μία εικόνα έχει τόσες ψήφους όσα είναι τα χαρακτηριστικά της διανύσματα. Μετά το τέλος της αναζήτησης NN του q, επιστρέφεται ένα μοναδικό *vector_id*. Κάθε *vector_id* είναι ένας δείκτης προς την εικόνα της βάσης δεδομένων, στην οποία ανήκει το χαρακτηριστικό διάνυσμα που βρέθηκε ως NN. Προκειμένου να μπορέσουμε να αξιολογήσουμε τα δεδομένα που παίρνουμε από την αναζήτηση, χρειάστηκε να δημιουργηθεί ένας πίνακας από *indexes*. Κάθε index αντιπροσώπευε μία εικόνα της βάσης δεδομένων. Τα πεδία του index ήταν το όνομα της εικόνας, ένας δείκτης προς τα χαρακτηριστικά διανύσματα που αντιπροσωπεύουν την εικόνα, ο αριθμός των διανυσμάτων που τη χαρακτηρίζουν, ο τύπος της και δύο ακόμα πεδία μετρητών. Ο ένας μετρητής αποθηκεύει το άθροισμα των ψήφων που παίρνει κάθε εικόνα της βάσης, ενώ ο άλλος αποθηκεύει το άθροισμα των Ευκλείδειων αποστάσεων $d[q,x]$, με x το διάνυσμα που βρέθηκε πιο κοντά. Κάθε φορά που το διάνυσμα μίας εικόνας επιστρέφεται σαν NN, ο μετρητής ψήφων της δομής της εικόνας αυξάνεται κατά

μία μονάδα και στο μετρητή των Ευκλείδειων αποστάσεων προσθέτουμε την $d[q,x]$. Η διαδικασία συνεχίζεται για όλα τα διανύσματα της εικόνας Q.

5.2 Παράμετροι ορθής ανάκτησης εικόνας

Αρχικά, υποθέσαμε ότι μόνο ο αριθμός των ψήφων θα ήταν ικανός να μας υποδείξει την εικόνα που αναζητούμε. Στα πειράματα όμως που έγιναν, φάνηκε ότι ο αριθμός των ψήφων δεν ήταν αρκετός σε όλες τις περιπτώσεις για να επιλέξουμε και να ανακτήσουμε τη σωστή εικόνα. Αν για παράδειγμα, μία εικόνα έχει y διανύσματα και μία άλλη w, με w > y, σε περίπτωση που είχαν και οι δύο ίδιο αριθμό ψήφων (*x_votes*) εξετάζοντας μόνο τις ψήφους δεν θα μπορούσαμε να αποφασίσουμε ποιά θα ανακτήσουμε. Έτσι, χρησιμοποιήσαμε το ποσοστό των ψήφων ανά εικόνα, δηλαδή διαιρέσαμε τις ψήφους με τον αριθμό των χαρακτηριστικών διανυσμάτων και το χαρακτηρίσαμε ως επαναληψιμότητα των χαρακτηριστικών (*repeatability*). Το παραπάνω μέτρο, χρησιμοποιήθηκε έτσι ώστε να ενισχύσουμε την επιλογή της κατάλληλης εικόνας, με βάση τον αριθμό των χαρακτηριστικών της διανυσμάτων. Διαιρώντας τις ψήφους με w και y θα έχουμε μία παράσταση όπου $\frac{x_{votes}}{w} < \frac{x_{votes}}{y}$ και η εικόνα που θα επιλεγόταν θα ήταν η πρώτη. Το σκεπτικό της επαναληψιμότητας δεν λύνει το πρόβλημα μόνο όταν υπάρχει ισοψηφία. Τα πειράματα έδειξαν ότι μία εικόνα μπορεί να πάρει το ανώτερο τόσες ψήφους όσα είναι τα χαρακτηριστικά της διανύσματα. Αν οι ψήφοι που έλαβε η εικόνα προσεγγίζουν τον αριθμό των χαρακτηριστικών της, ανεξάρτητα από τις ψήφους που έχουν πάρει οι άλλες, η πιθανότητα να είναι αυτή η εικόνα που αναζητούμε είναι μεγαλύτερη. Με βάση τα παραπάνω ο πίνακας δομών (*indexes*), των εικόνων της βάσης δεδομένων, εξετάζεται για να βρεθούν και να επιστραφούν οι *top_match indexes* που έχουν τα μεγαλύτερα ποσοστά ψήφων/αριθμό χαρακτηριστικών. Ο αριθμός *top_match,* δίδεται σαν παράμετρος στο σύστημα από το χρήστη.

5.3 Περιγραφή Χαρακτηριστικών

Για τις ανάγκες του πρώτου πειράματος, πήραμε ένα δείγμα από τις εικόνες της βάσης δεδομένων και εφαρμόσαμε σε αυτές κάποιους μετασχηματισμούς. Στόχος του πειράματος ήταν να ανακαλύψουμε πως μπορούμε να ανακτήσουμε σωστά τις μετασχηματισμένες εικόνες. Οι μετασχηματισμοί αφορούσαν διάφορα στάδια πρόσθεσης θορύβου, περιστροφής και αύξησης αντίθεσης. Στη συνέχεια, εφαρμόσαμε στις εικόνες τον αλγόριθμο SIFT για να εξάγουμε τα χαρακτηριστικά τους διανύσματα. Μετά την εξαγωγή των χαρακτηριστικών προσπαθήσαμε να αναζητήσουμε τις μετασχηματισμένες εικόνες μέσα στη βάση δεδομένων. Τα αποτελέσματα της αναζήτησης κάθε εικόνας ήταν μία λίστα με τα *top_match indexes*, όπως παρουσιάζεται παραπάνω.

Το πρώτο χαρακτηριστικό που χρησιμοποιήσαμε είναι το *repeatability*. Έτσι, στη λίστα με τα *indexes* που επιστρέφεται μετά από μία αναζήτηση επιστρέφουμε την εικόνα της οποίας το *index* έχει το μεγαλύτερο *repeatability* και απορρίπτουμε όλες τις άλλες. Το ποσοστό σωστής ανάκτησης, ήταν 100% για ορισμένους μετασχηματισμούς. Υπήρχαν, όμως, περιπτώσεις όπου επιστρέφοντας την εικόνα με την μεγαλύτερη επαναληψιμότητα δεν είχαμε τόσο καλά αποτελέσματα. Με βάση τα παραπάνω, χρειάστηκε να ορίσουμε και ένα δεύτερο χαρακτηριστικό εκτός της επαναληψιμότητας, έτσι ώστε να κάνουμε σωστή ανάκτηση.

Μία αναζήτηση θα μας επιστρέψει πάντα έναν Κοντινότερο Γείτονα, ανεξαρτήτως της Ευκλείδειας απόστασης που έχει το προς αναζήτηση διάνυσμα από με αυτό που επιστρέφεται. Με άλλα λόγια, ακόμα και αν η εικόνα που αναζητούμε δεν υπάρχει στη βάση δεδομένων και πάλι η αναζήτηση θα βρει τα κοντινότερα διανύσματα, τα οποία στην πλειοψηφία τους θα έχουν μεγάλες Ευκλείδειες αποστάσεις. Με βάση την Ευκλείδεια απόσταση ορίζουμε ένα χαρακτηριστικό που είναι ο μέσος όρος των Ευκλείδειων αποστάσεων που βρέθηκαν για μία εικόνα. Στην Ενότητα [5.1] αναφέρεται ότι μαζί με τον αριθμό των ψήφων αποθηκεύεται και το άθροισμα των Ευκλείδειων αποστάσεων. Επομένως, διαιρώντας το άθροισμα αυτό με τον αριθμό των ψήφων (*ratio*) βρίσκουμε τη μέση απόσταση κάθε διανύσματος, που η αναζήτηση το επέστρεψε ως κοντινότερο. Εύκολα καταλαβαίνουμε, ότι όσο πιο κοντινά είναι τα διανύσματα τόσο το *ratio* τείνει προς το μηδέν. Δεδομένης μίας λίστας *indexes* που έχουν το μεγαλύτερο *repeatability* μετά την αναζήτηση, ψάχνουμε για να βρούμε την εικόνα που παρουσιάζει το μικρότερο *ratio.* Τα πειράματα έδειξαν ότι όταν επιστρέψουμε την εικόνα με το μικρότερο *ratio*, τα ποσοστά σωστής ανάκτησης αυξάνονται.

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

Για τις ανάγκες των πειραμάτων δημιουργήθηκε μία βάση δεδομένων 250 εικόνων [12] η οποία περιείχε πέντε κατηγορίες αντικειμένων (ανθρώπους, ποδήλατα, μηχανές, αυτοκίνητα, λεωφορεία). Πήραμε ένα δείγμα 50 εικόνων από τη βάση δεδομένων και φτιάξαμε ένα σύνολο *Query1*, το οποίο είχε 10 εικόνες από κάθε κατηγορία. Για τις ανάγκες του πρώτου πειράματος, οι *query* εικόνες υποβλήθηκαν σε μετασχηματισμούς θορύβου, περιστροφής και αύξησης αντίθεσης. Οι συναρτήσεις *imnoise* και *imrotate* που χρησιμοποιήσαμε στα πειράματα είναι του εργαλείου Matlab.

Για πρόσθεση θορύβου χρησιμοποιήθηκε η συνάρτηση *imnoise (image,Type,Mean,variance)*. Ο θόρυβος που βάλαμε ήταν '*Gaussian noise*', για παράγοντες *variance* = 0.02, 0.04, 0.06, 0.08, 0.1. Για τις *Query1* εικόνες υπολογίσαμε το μέσο όρο του PSNR $= 10log_{10}(\frac{MAX_I^2}{MSE})$, με $MAX_I = 255$ για εικόνες και MSE το μέσο τετραγωνικό σφάλμα, για τις διαφορετικές περιπτώσεις *variance*. Στον Πίνακα 6.1 φαίνεται το PSNR για τις αντίστοιχες τιμές *variance.*

Πίνακας 6.1	
Variance	***PSNR***
0.02	18db
0.04	15db
0.06	13db
0.08	11db
0.1	9db

Οι περιστροφή έγινε για $\pm 5, \pm 10, \pm 20, \ldots \pm 90$ μοίρες με χρήση της συνάρτησης *imrotate (image, angle)* και η αύξηση αντίθεσης με μία συνάρτηση contrast_stretch για συντελεστές 0.8, 0.9, …, 1.2. Εκτός από τις 50 *Query*1 εικόνες που αναφέρονται παραπάνω, φτιάξαμε άλλο ένα σύνολο από 50 εικόνες οι οποίες δεν ανήκαν στις εικόνες της βάσης μας, περιείχαν όμως τις ίδιες κατηγορίες αντικειμένων (*Query*2). Τέλος, μετασχηματίζοντας τις εικόνες του *Query*2 συνόλου με τους παραπάνω μετασχηματισμούς, φτιάξαμε ένα ακόμα σύνολο *Query*3. Τέλος, δημιουργήθηκε ένα *Query*4 σύνολο που περιείχε εντελώς διαφορετικές κατηγορίες εικόνων από με αυτές στην βάση δεδομένων. Τα σύνολα *Query* χρησιμοποιήθηκαν για τα πειράματα τα οποία παρουσιάζονται παρακάτω.

6.1 Περιγραφή Πειραμάτων

6.1.1 Πείραμα 1ο

Η βασική ιδέα του πειράματος είναι ότι έχοντας μία εικόνα που έχει υποστεί κάποιο μετασχηματισμό, μπορούμε να αναγνωρίσουμε και να ανακτήσουμε την *original* εικόνα, αν βέβαια αυτή υπάρχει στη βάση δεδομένων. Στο παρόν πείραμα, μία εικόνα επιστρέφεται στο χρήστη. Για τις ανάγκες του πρώτου πειράματος χρησιμοποιήσαμε το σύνολο εικόνων *Query*1, όπως αυτό περιγράφεται παραπάνω. Συγκεκριμένα, προσπαθήσαμε να χρησιμοποιήσουμε τα χαρακτηριστικά επαναληψιμότητα και μέση απόσταση, έτσι ώστε να μπορέσουμε να αποφασίσουμε ποιά είναι η σωστή εικόνα. Μετά το τέλος της αναζήτησης/ψηφοφορίας, διατρέξαμε στον πίνακα των *indexes* για να βρούμε αυτά με τη μεγαλύτερη επαναληψιμότητα. Μία λίστα από *top_match indexes* επιστράφηκε και έγινε η επεξεργασία της. Πρώτα, πήραμε τα αποτελέσματα με βάση τη επαναληψιμότητα, δηλαδή, προσπαθήσαμε να βρούμε το ποσοστό των σωστών ανακτήσεων επιστρέφοντας από τα *top_match indexes* την εικόνα με το μεγαλύτερο επαναληψιμότητα. Στη συνέχεια, πήραμε τα αποτελέσματα με βάση τη μέση απόσταση, επιστρέφοντας από τα *top_match indexes* την εικόνα που είχε το μικρότερο.

Στο επόμενο στάδιο, εκτελέσαμε το ίδιο πείραμα με βασική ιδέα ότι επιστρέφοντας στο χρήστη ένα μικρό αριθμό από *k* εικόνες, η εικόνα που ψάχνει θα είναι μέσα σε αυτές. Για την πραγματοποίηση αυτού χρησιμοποιήσαμε τον αριθμό των k *top_match* εικόνων σαν συμπληρωματικό στοιχείο, χωρίς περεταίρω επεξεργασία. Η διαφοροποίηση με παραπάνω είναι ότι η διαδικασία ταυτοποίησης αφήνεται στο χρήστη, ο οποίος βλέποντας τις εικόνες αποφασίζει ποια είναι αυτή που αναζητεί.

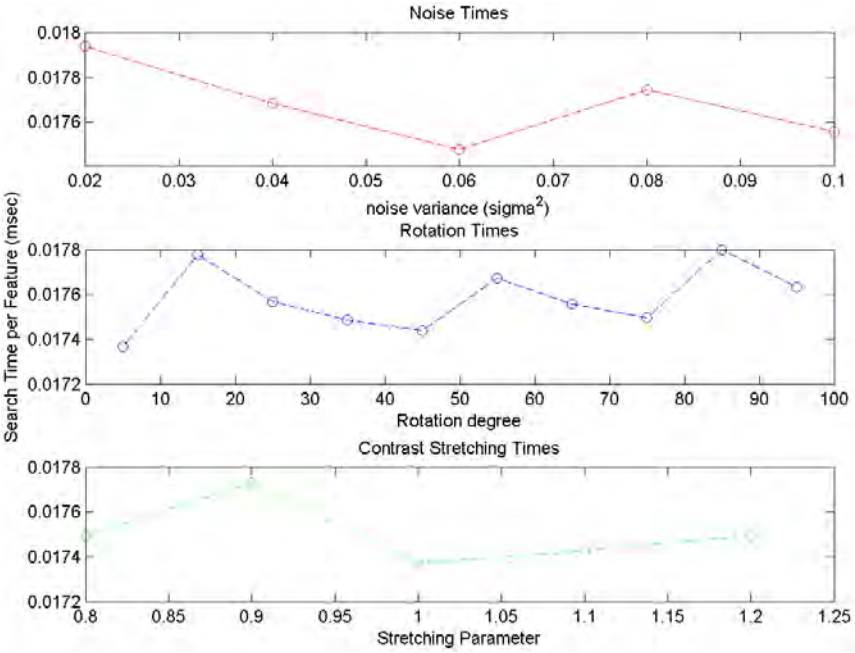
Αποτελέσματα 1^{ου} πειράματος

Τα αποτελέσματα των σωστών ανακτήσεων φαίνονται στους παρακάτω Πίνακες 6.2-6.4. Η αναζήτηση NN έγινε με τη χρήση Depth only Fast NN, Complete Fast NN και Limited Fast NN.

Πίνακας 6.2 Depth only Fast NN Search		
	Repeatability	**Ratio**
Gaussian Noise	93.6%	96.8%
Rotation	100%	55,4%
Contrast Stretching	100%	97.6%

Στον Πίνακα 6.1 παρατηρούμε ότι για τις εικόνες, στις οποίες έχουμε προσθέσει θόρυβο τα ποσοστά σωστών ανακτήσεων είναι καλύτερα αν επιστρέψουμε αυτές που

έχουν το μικρότερη μέση απόσταση. Για τις εικόνες που έχουν περιστραφεί και για αυτές στις οποίες έχει γίνει αύξηση της αντίθεσης επαληθεύτηκε το 100% των περιπτώσεων λαμβάνοντας υπόψη την επαναληψιμότητα. Τα ποσοστά λαμβάνοντας υπόψη τη μέση απόσταση είναι αρκετά χαμηλά, διότι περιστρέφοντας την εικόνα τα διανύσματα αλλάζουν κατά ένα παράγοντα και αυτό τα κάνει να είναι πιο κοντά σε απόσταση με διανύσματα άλλων εικόνων και όχι με αυτά της εικόνας που αναζητείται. Αν όμως κάποιος παρατηρήσει παράλληλα και την επαναληψιμότητα των εικόνων που έχουν το μικρότερη μέση απόσταση στις λανθασμένες επιστροφές, θα καταλάβει ότι απλά έτυχε μερικά διανύσματα να έχουν μικρότερη απόσταση σε σχέση με αυτά της σωστής εικόνας.



Εικόνα 6.1 Times Depth only Fast NN Search

Στην Εικόνα 6.1 παρουσιάζονται οι γραφικές παραστάσεις των χρόνων που χρειάστηκαν για να βρεθεί ένα διάνυσμα μέσα στη βάση δεδομένων, με τη μέθοδο Depth only Fast NN Search για κάθε περίπτωση μετασχηματισμού. Στον άξονα y παρουσιάζονται οι τιμές των χρόνων και δίνονται σε millisecond, ενώ για τον άξονα x η κάθε μία γραφική παράσταση έχει τις μεταβολές των παραμέτρων για κάθε μετασχηματισμό, όπως αυτές δίνονται στην αρχή του κεφαλαίου. Δεδομένου των παραπάνω, για να ανακτηθεί μια εικόνα με ~1000 χαρακτηριστικά διανύσματα χρειάζονται περίπου 0.02sec ή 20msec. Στο χρόνο που δίνεται έχει συμπεριληφθεί και ο χρόνος επεξεργασίας των αποτελεσμάτων.

Πίνακας 6.3 Complete Fast NN Search		
	Repeatability	**Ratio**
Gaussian Noise	96.8%	94.4%
Rotation	100%	34.2%
Contrast Stretching	100%	96.8%

Πίνακας 6.4 Limited Fast NN Search		
	Repeatability	**Ratio**
Gaussian Noise	96%	96%
Rotation	100%	42.5%
Contrast Stretching	100%	97.2%

Για τη Complete Fast NN Search και τη Limited Fast NN Search τα αποτελέσματα αξιολογήθηκαν όπως και στη DOS. Αξίζει να σημειωθεί και πάλι ότι η Complete μέθοδος βρίσκει τον πραγματικό NN και όχι κάποια προσέγγιση αυτού. Παρατηρούμε ξανά ότι λαμβάνοντας υπόψη τη μέση απόσταση, τα ποσοστά είναι χαμηλά για τον ίδιο λόγο όπως και στην περίπτωση της DOS μεθόδου.

Στην Εικόνα 6.2 παρουσιάζεται η γραφική παράσταση των χρόνων που χρειάστηκαν για να βρεθεί ένα χαρακτηριστικό διάνυσμα μέσα στη βάση δεδομένων, με τη μέθοδο Complete Fast NN Search. Στο άξονα y δίνονται οι χρόνοι αναζήτησης σε second και για τον άξονα x ισχύουν ότι και για τη Depth only Fast NN.

Στην Εικόνα 6.3 δίνονται οι γραφικές παραστάσεις των χρόνων για τη μέθοδο Limited Fast NN Search για διάφορους αριθμούς *backtracking.* Στα πειράματα ο αριθμός από *backtracking* που χρησιμοποιήθηκαν ήταν 500, 1000, 1500, …6000. Παρατηρώντας την γραφική παράσταση βλέπουμε ότι αυξάνοντας τον αριθμό των *backtracking* ο χρόνος αναζήτησης, όπως ήταν φυσικό, αυξάνεται.

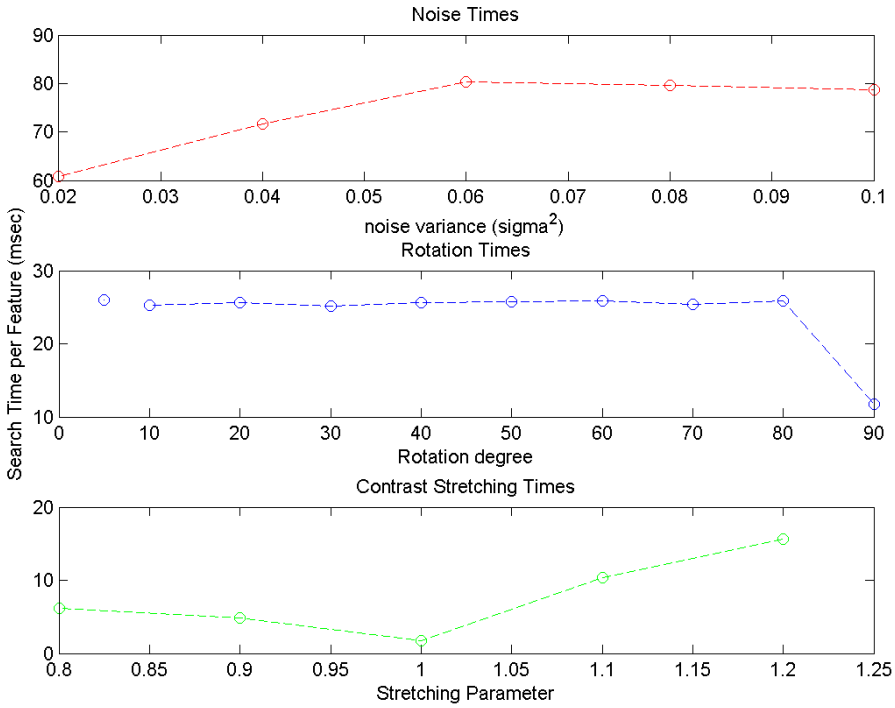
Συγκρίνοντας τα αποτελέσματα των χρόνων για τις τρεις μεθόδους, παρατηρούμε ότι η DOS είναι η πιο γρήγορη και δεν υστερεί σε ακρίβεια από τις άλλες δύο. Όπως αναφέρθηκε και στο κεφάλαιο 4 είναι περίπου ~3K φορές πιο γρήγορη από τη Complete και ~40 φορές από τη Limited, ανάλογα με τα *backtracking* που έχουμε ορίσει. Τα αποτελέσματα επαληθεύτηκαν και φαίνονται στις γραφικές παραστάσεις.

Στην Εικόνα 6.4 φαίνεται η γραφική παράσταση των χρόνων της Full NN Search. Οι συγκρίσεις των χρόνων αναζήτησης των Fast NN και της Full NN παρουσιάζονται στον Πίνακα 6.5. Επειδή το speedup ήταν διαφορετικό και ανάλογο του μετασχηματισμού, στο Πίνακα 6.5 δίνεται η χειρότερη επίδοση που μπορεί να έχει κάθε μέθοδος.

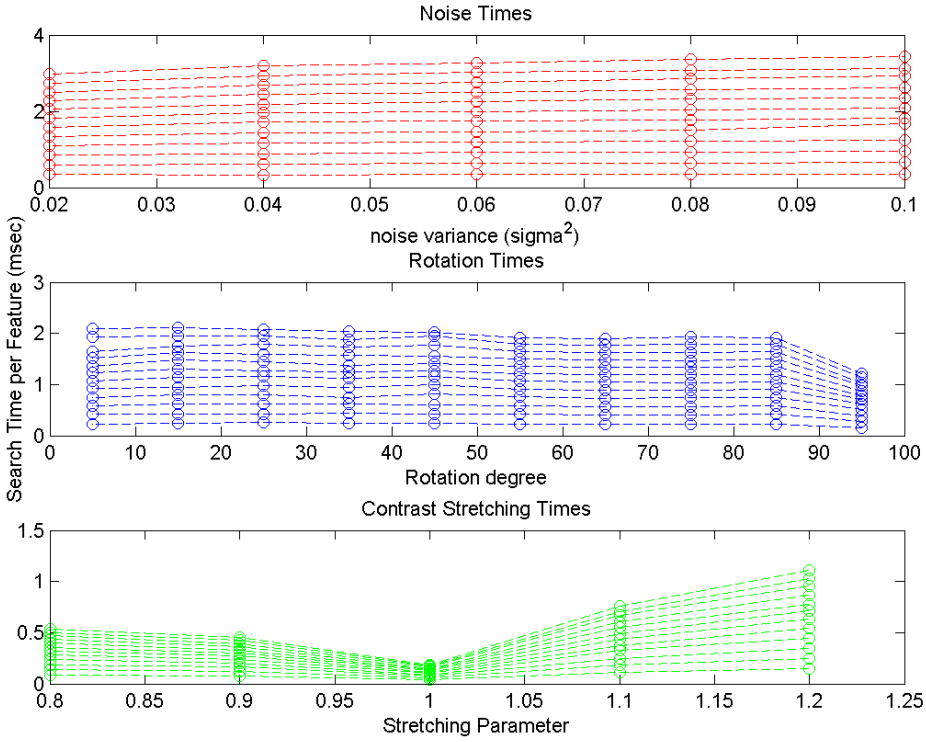
Πίνακα 6.5 Fast NN speedup over Full NN	
Method	***Fast NN speedup***
Full NN	1x
Fast NN Depth only Search	11000x
Fast NN Limited Search	800x – 80x
Fast NN Complete Search	2.5x

Η αναζήτηση NN έγινε για διανύσματα 128-διαστάσεων. Είναι η πρώτη φόρα που ο Fast NN χρησιμοποιείται για αναζήτηση τόσο μεγάλων διαστάσεων διανυσμάτων. Έτσι, δεν γνωρίζαμε εκ των προτέρων τη συμπεριφορά του. Παρατηρούμε ότι για μία εφαρμογή γρήγορης ανάκτησης, η μέθοδος αναζήτησης Fast NN Complete δε θα ήταν η κατάλληλη να χρησιμοποιηθεί, αφού προσφέρει ένα μικρό *speedup* σε σχέση με τη Full NN. Επίσης, τα αποτελέσματα δεν έχουν μεγάλες διαφορές από τις άλλες δύο μεθόδους. Επομένως, μπορούμε να πούμε ότι μία exact NN αναζήτηση για τόσο μεγάλα διανύσματα είναι ασύμφορη, διότι ούτε γρήγορη είναι ούτε έχει μεγάλες διαφορές στην ακρίβεια. Οι DOS και Limited είναι δύο προσεγγιστικές μέθοδοι εύρεσης NN και παρατηρούμε ότι παρουσιάζουν ένα σχετικά μεγάλο speedup σε σχέση με τη Full NN.
Με τη DOS μέθοδο τα αποτελέσματα σωστών ανακτήσεων φαίνονται σε υψηλά επίπεδα, αλλά ίσως δεν είναι πάντα τόσο ευσταθή, αφού μικρές διαφορές στην επαναληψιμότητα και στη μέση απόσταση μπορούν να τα αλλάξουν. Με τη DOS μέθοδο αρκετές φορές τα *indexes* έχουν πολύ κοντινές τιμές για τα δύο χαρακτηριστικά. Παραμένει βέβαια μία πολύ γρήγορη μέθοδος προσεγγιστικής αναζήτησης και θα μπορούσε να χρησιμοποιηθεί σε *real time* εφαρμογές. Αν θέλουμε να έχουμε πιο ευσταθή αποτελέσματα μπορούμε να επιλέξουμε τη Limited με κάποιο μικρό αριθμό *backtracking* , π. χ. 200-1000. Έτσι, θα έχουμε ένα αρκετά καλό speedup και πιο ευσταθή αποτελέσματα, αφού η Limited δίνει μία καλύτερη προσέγγιση του NN.

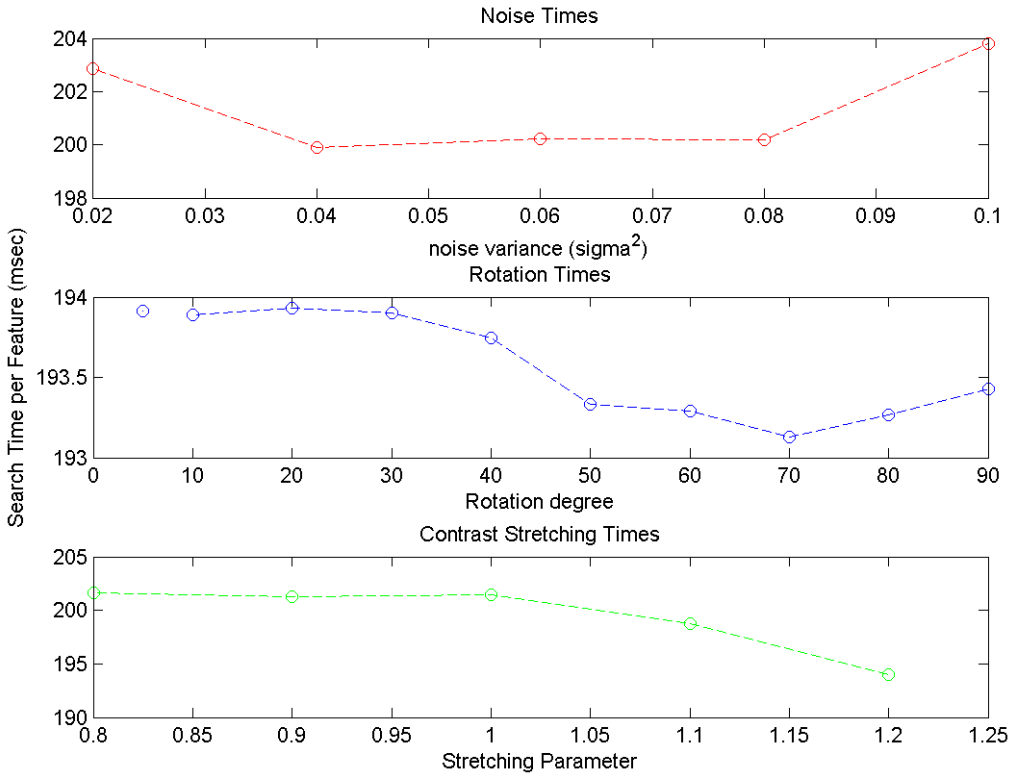
Σύμφωνα με τα παραπάνω, ο χρήστης μπορεί να επιλέξει τη μέθοδο που θέλει βάση τα κριτήρια ευστάθειας και ταχύτητας.



Εικόνα 6.2 Times Complete Fast NN Search



Εικόνα 6.3 Times Limited Fast NN Search for different values backtracking

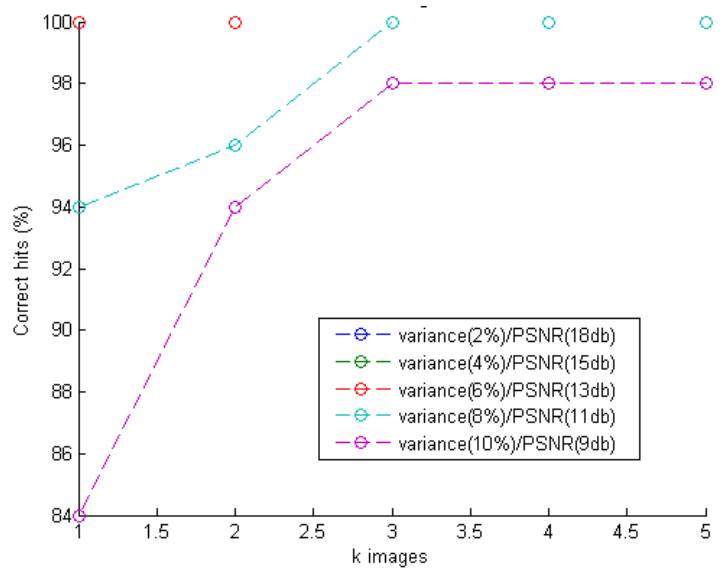


Εικόνα 6.4 Times for Full NN Search

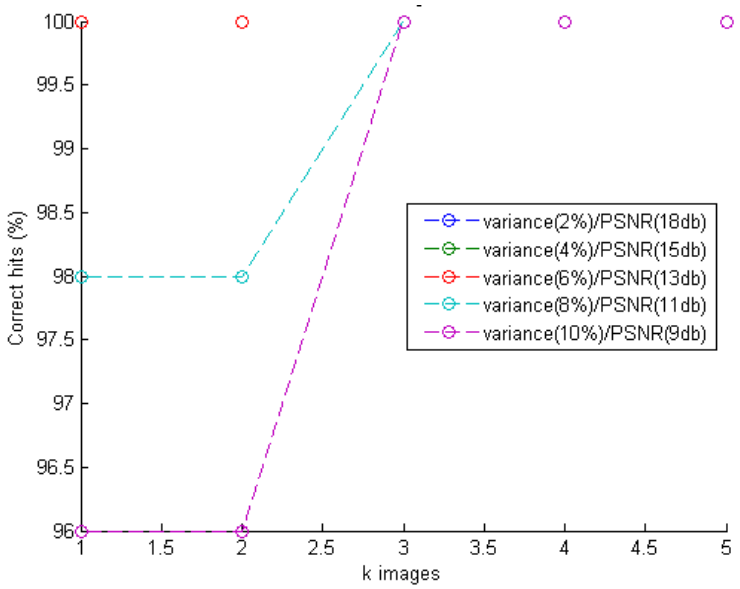
Ως επέκταση του 1ου πειράματος, επιλέξαμε να επιστρέφουμε στο χρήστη ένα αριθμό από k εικόνες και αυτός να μπορεί να διαλέγει ποιά από αυτές είναι αυτή που αναζητεί.

Οι Εικόνες 6.5-6.7 παρουσιάζουν τα αποτελέσματα για την περίπτωση Gaussian Noise. Οι άλλοι δύο μετασχηματισμοί μας έδιναν πάντα, 100% σωστά αποτελέσματα και οι γραφικές παραστάσεις ήταν τετριμμένες. Στον κάθετο άξονα φαίνεται το ποσοστό των σωστών ανακτήσεων και στον οριζόντιο η διαφοροποίηση του αριθμού των επιστρεφόμενων εικόνων.

Στην Εικόνα 6.5 παρουσιάζονται τα ποσοστά επιτυχημένων επιστροφών για τη μέθοδο Depth only Fast NN. Παρατηρούμε ότι για επιστρεφόμενο αριθμό εικόνων k = 1 στις τρεις πρώτες περιπτώσεις θορύβου έχουμε 100% σωστά αποτελέσματα. Όταν όμως, ο PSNR πέφτει στα 11db($\sigma^2 = 0.08$) και 9 db($\sigma^2 = 0.1$) τα ποσοστά αλλάζουν σε 94% και 84%, αντίστοιχα. Αυξάνοντας τον αριθμό των επιστρεφόμενων εικόνων σε k = 2, βλέπουμε ότι το ποσοστό των επαληθεύσεων για εικόνες με PSNR θορύβου 11db($\sigma^2 = 0.08$) αυξάνεται κατά 2%, ενώ για εικόνες με PSNR θορύβου 9db($\sigma^2 = 0.08$) αυξάνεται κατά 10%. Στη συνέχεια, παρατηρούμε ότι για k = 3 τα ποσοστά των επαληθεύσεων φτάνουν στο ανώτερο όριο, με 100% για εικόνες με PSNR 11db($\sigma^2 = 0.08$) και 98% για PSNR 9db($\sigma^2 = 0.08$) και δεν παρουσιάζεται αλλαγή επιστρέφοντας παραπάνω εικόνες. Οι χρόνοι αναζήτησης είναι όμοιοι με τους παραπάνω στην Εικόνα 6.1.



Εικόνα 6.5 Depth only Fast NN Search for k images

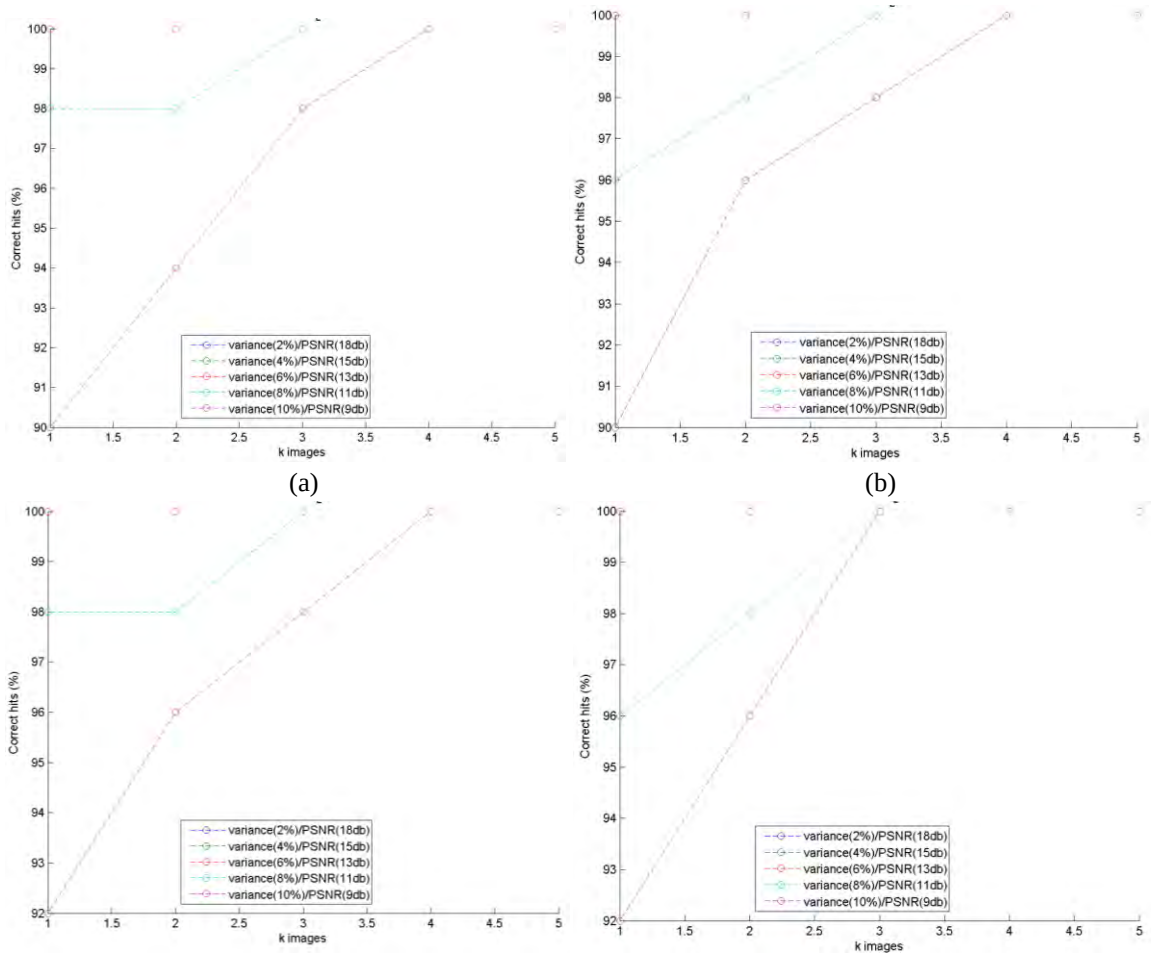


Εικόνα 6.6 Complete Fast NN Search for k images

Η γραφική παράσταση της Εικόνας 6.6, παρουσιάζει τα ποσοστά επιτυχημένων επιστροφών για τη μέθοδο Complete Fast NN. Παρατηρούμε ότι για επιστρεφόμενο αριθμό εικόνων k = 1 στις τρεις πρώτες περιπτώσεις θορύβου έχουμε 100% σωστά αποτελέσματα. Όταν όμως, ο PSNR πέφτει στα 11db($\sigma^2 = 0.08$) και 9 db($\sigma^2 = 0.1$) τα ποσοστά αλλάζουν σε 98% και 96%, αντίστοιχα. Αυξάνοντας τον αριθμό των επιστρεφόμενων εικόνων σε k = 2, βλέπουμε ότι τα ποσοστά παραμένουν τα ίδια. Στη συνέχεια όμως, παρατηρούμε ότι για k = 3 τα ποσοστά των επαληθεύσεων φτάνουν και τα δύο στο 100%. Έτσι, καταλαβαίνουμε ότι αν επιστρέψουμε τρεις εικόνες με τη

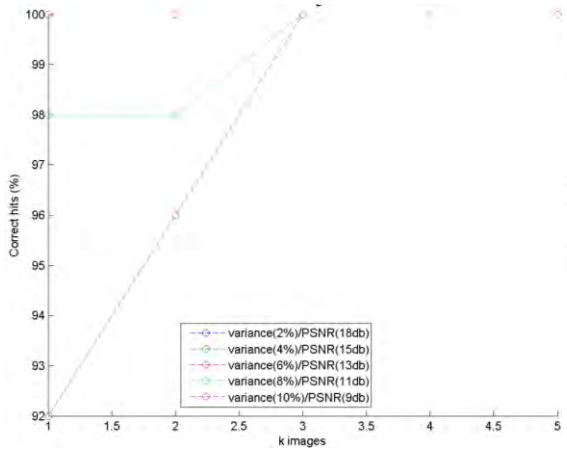
Complete Fast NN μέθοδο, η εικόνα που ψάχνουμε είναι πάντα μέσα σε αυτές. Οι χρόνοι αναζήτησης είναι ίδιοι με αυτούς στην Εικόνα 6.2. Σε σχέση με την προηγούμενη μέθοδο υπάρχει μία αισθητή διαφορά στα αποτελέσματα για τον ίδιο αριθμό k εικόνων. Η Complete Fast NN βρίσκει τον πραγματικό NN, υστερεί όμως στο χρόνο εκτέλεσης. Παρατηρούμε, όμως, ότι για k = 3 τα αποτελέσματα είναι ίδια για PSNR 11db($\sigma^2 = 0.08$) και διαφέρουν κατά 2% για PSNR 9db($\sigma^2 = 0.1$). Επομένως, μεταξύ τριών εικόνων και οι δύο μέθοδοι σχεδόν βρίσκουν την εικόνα που αναζητείται, με τη διαφορά ότι η DOS το κάνει ~3K φορές πιο γρήγορα.

Οι γραφικές παραστάσεις της Εικόνας 6.6 αφορούν τα αποτελέσματα της Limited Fast NN Search μεθόδου για διαφορετικό αριθμό *backtracking*. Μπορεί να παρατηρήσει κανείς ότι υπάρχουν διαφορές για τα ποσοστά επαλήθευσης όσο ο αριθμός των *backtracking* αυξάνεται. Η γραφική παράσταση (h) είναι για 4000 *backtracking*. Για μεγαλύτερο αριθμό *backtracking* οι γραφικές παραστάσεις δεν άλλαζαν και έτσι δεν τις έχουμε συμπεριλάβει. Προφανώς, αν δεν θέσουμε περιορισμό στα *backtracking* τα αποτελέσματα θα είναι ίδια με τη Complete Fast NN. Οι χρόνοι είναι ίδιοι με αυτούς της Εικόνας 6.3.

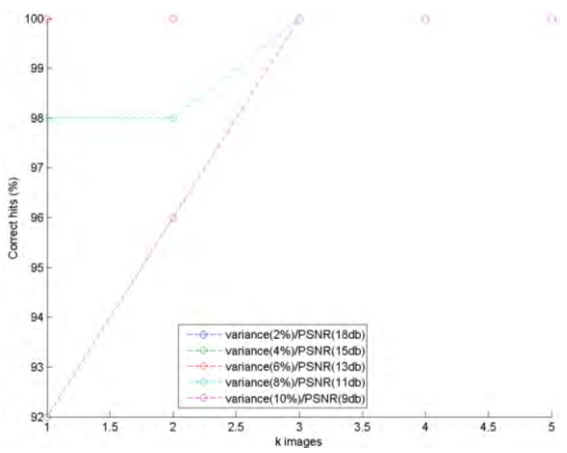


(a) (b)

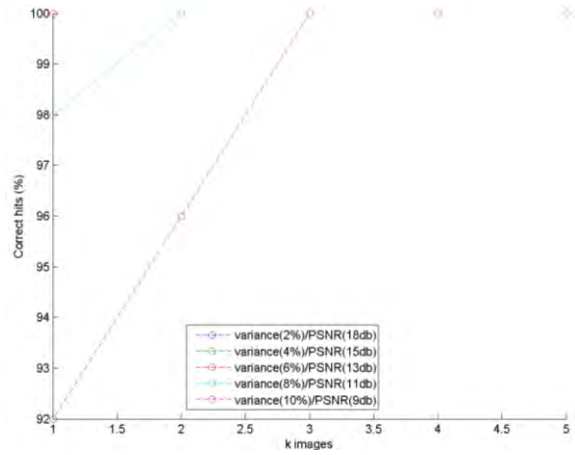
(c) (d)



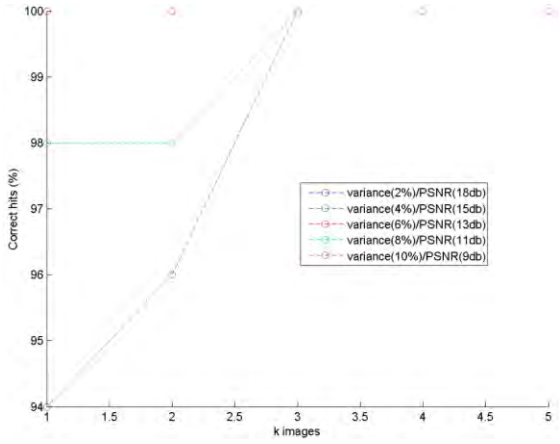
(e)



(f)



(g)



(h)

Εικόνα 6.7 Limited Fast NN Search for k images οι (a)-(h) γραφικές παραστάσεις είναι η κάθε μια για διαφορετικό αριθμό *backtracking.*

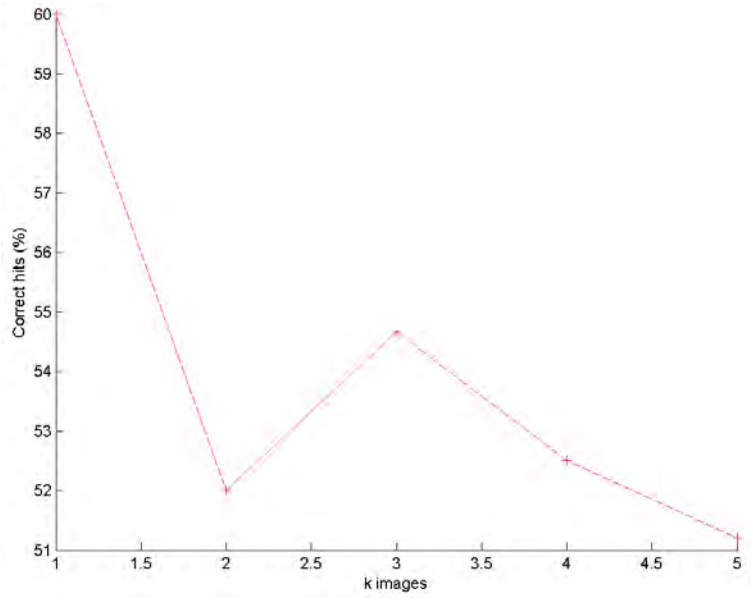
6.1.2 Πείραμα 2ο

Το δεύτερο πείραμα βασίζεται στην ιδέα του προβλήματος ταξινόμησης (classification problem). Δεδομένης μια εικόνας που περιέχει ένα συγκεκριμένο αντικείμενο, αναζητούμε στη βάση δεδομένων εικόνες που περιέχουν το ίδιο αντικείμενο. Στο πείραμα επιστρέψαμε και πάλι τη λίστα *top_match indexes* με το μεγαλύτερο *repeatability* και αυξήσαμε τον αριθμό επιστρεφόμενων εικόνων στο χρήστη. Προφανώς, εάν επιστρέψουμε όλες τις εικόνες της βάσης δεδομένων μπορεί να βρει αυτές που θέλει, όμως δεν επιθυμούμε να ψάξει όλες τις εικόνες. Επομένως, επιστρέφεται μία λίστα από k εικόνες που η πιθανότητα να είναι ίδιας κατηγορίας με αυτή που αναζητεί είναι

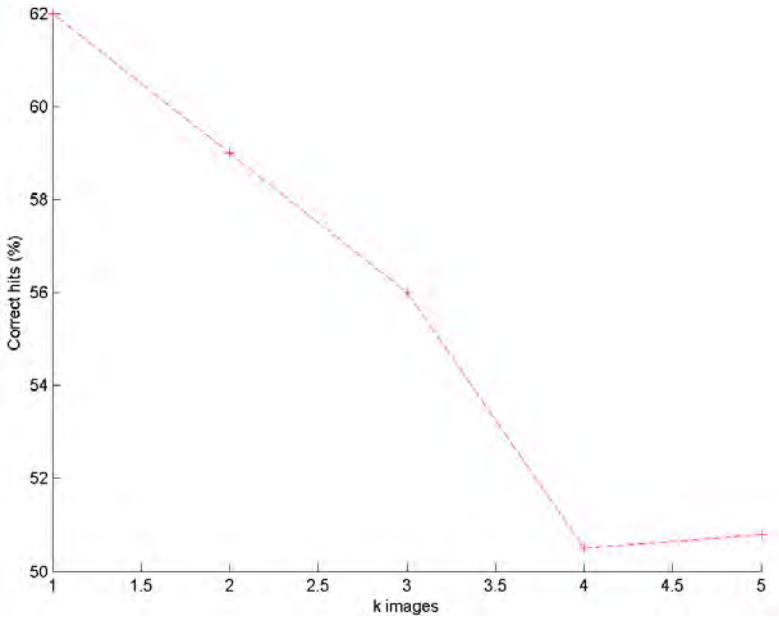
μεγαλύτερη. Επεκτείνοντας λίγο ακόμα το συγκεκριμένο πείραμα, προσπαθήσαμε να εξετάσουμε πως επηρεάζεται η ανάκτηση, σε περίπτωση που οι εικόνες έχουν υποστεί μετασχηματισμούς. Αν, δηλαδή, σε μία εικόνα που παραμορφώθηκε μπορεί να αναγνωριστεί το περιεχόμενο. Για το πρώτο σκέλος του πειράματος χρησιμοποιήθηκε το *Query2* σύνολο εικόνων, ενώ για το δεύτερο το *Query3.*

Αποτελέσματα 2ου πειράματος

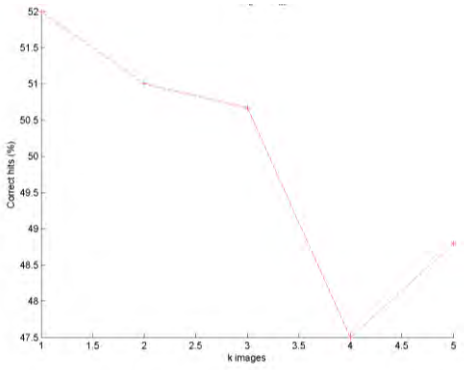
Όπως περιγράφεται παραπάνω, στο συγκεκριμένο πείραμα χρησιμοποιήσαμε το σύνολο των εικόνων *Query2*. Με δεδομένη μία εικόνα του συνόλου *Query2*, αναζητήσαμε στη βάση δεδομένων προκειμένου να βρούμε εικόνες που να είναι ίδια κατηγορίας. Όπως και παραπάνω, το κριτήριο αναζήτησης ήταν το *repeatability* που παρουσίαζαν τα *indexes* των εικόνων. Τα αποτελέσματα των μεθόδων παρουσιάζονται στις παρακάτω εικόνες γραφικές παραστάσεις.



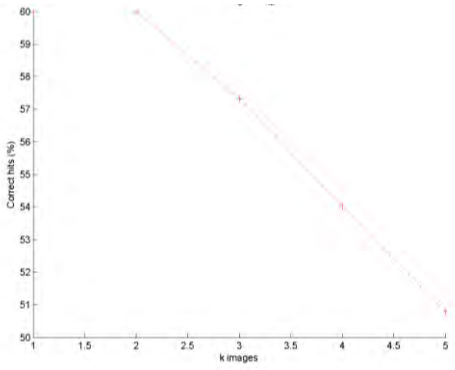
Εικόνα 6.8 Depth only Fast NN Search



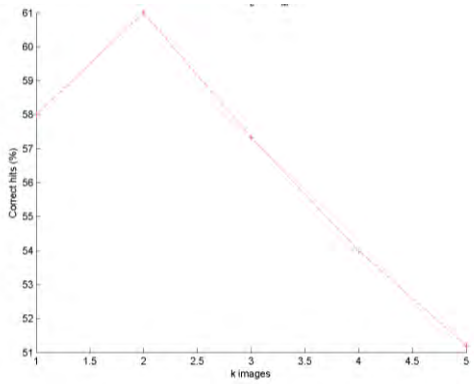
Εικόνα 6.9 Complete Fast NN Search



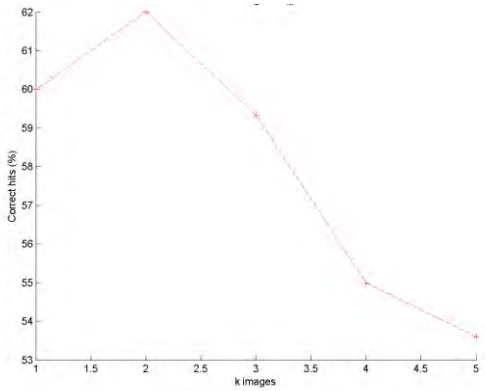
(a)



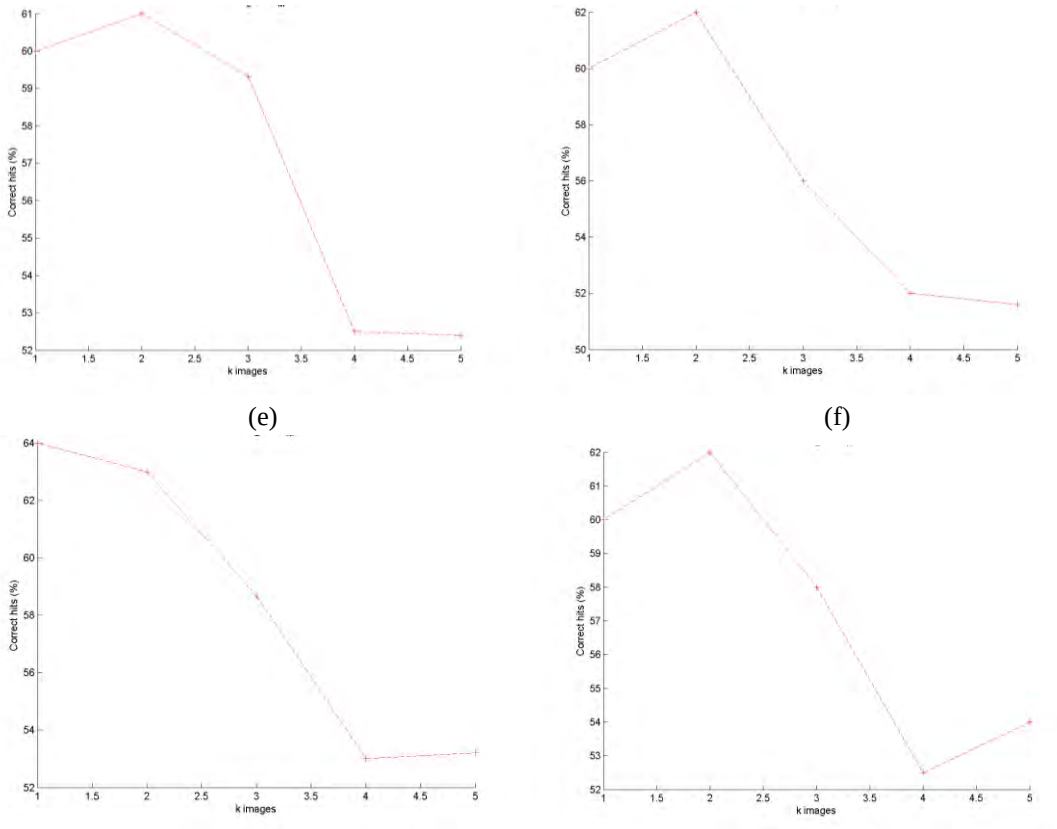
(b)



(c)



(d)

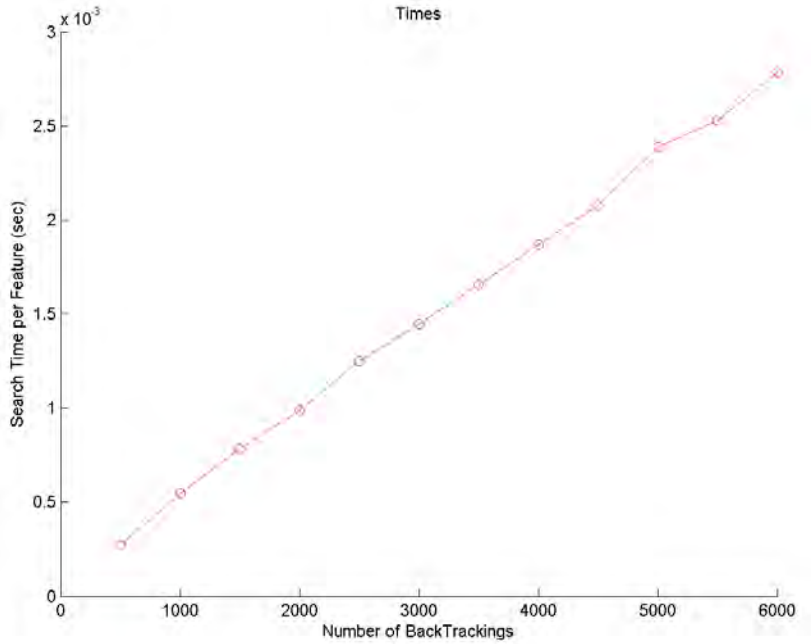


(e) (f)

(g) (h)

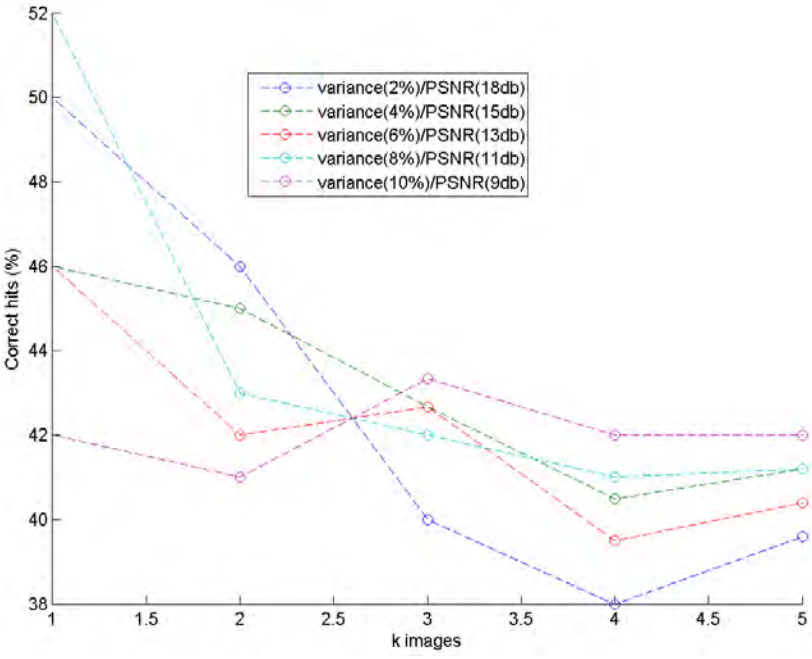
Εικόνα 6.10 Limited Fast NN Search for k images οι (a)-(h) γραφικές παραστάσεις είναι η κάθε μια για διαφορετικό αριθμό *backtracking*.

Από τις γραφικές παραστάσεις παρατηρούμε ότι όλοι οι μέθοδοι δίνουν περίπου τα ίδια αποτελέσματα. Ο κάθετος άξονας μας δείχνει το ποσοστό των εικόνων που βρίσκονται, να είναι της ίδια κατηγορίας με την *query* και ο οριζόντιος άξονας είναι ο αριθμός των εικόνων που αποφασίζουμε να επιστρέψουμε στο χρήστη. Τα αποτελέσματα έδειξαν ότι επιστρέφοντας τις εικόνες με το μεγαλύτερο *repeatability* και χωρίς καμία επιπλέον επεξεργασία της εικόνας, περίπου οι μισές ανήκαν στην ίδια κατηγορία με την *query* εικόνα. Οι χρόνοι αναζήτησης ενός διανύσματος με DOS Fast NN και Complete είναι 0.0185 millisecond και 58 millisecond, αντίστοιχα. Οι χρόνοι αναζήτησης για τη Limited μέθοδο φαίνονται στην Εικόνα 6.11 για διαφορετικό αριθμό *backtracking*.

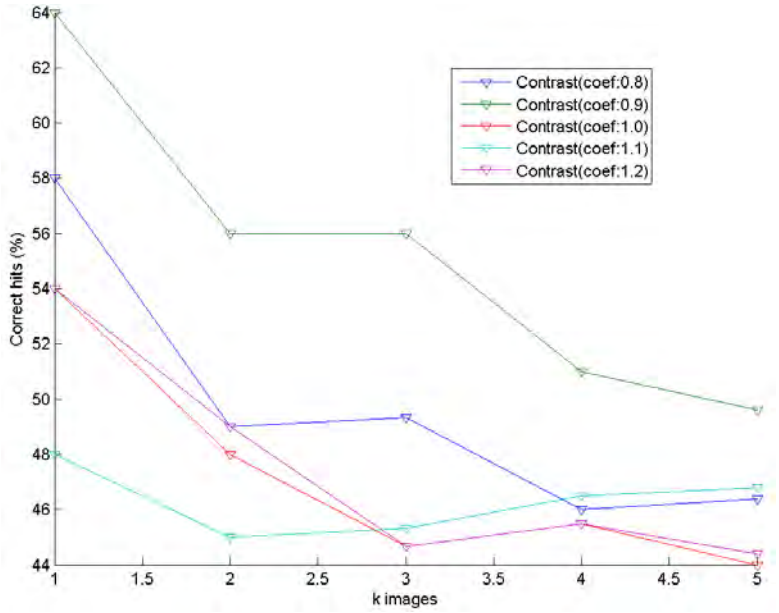


Εικόνα 6.11 Times for Limited Fast NN Search for 500-6000 *backtracking*

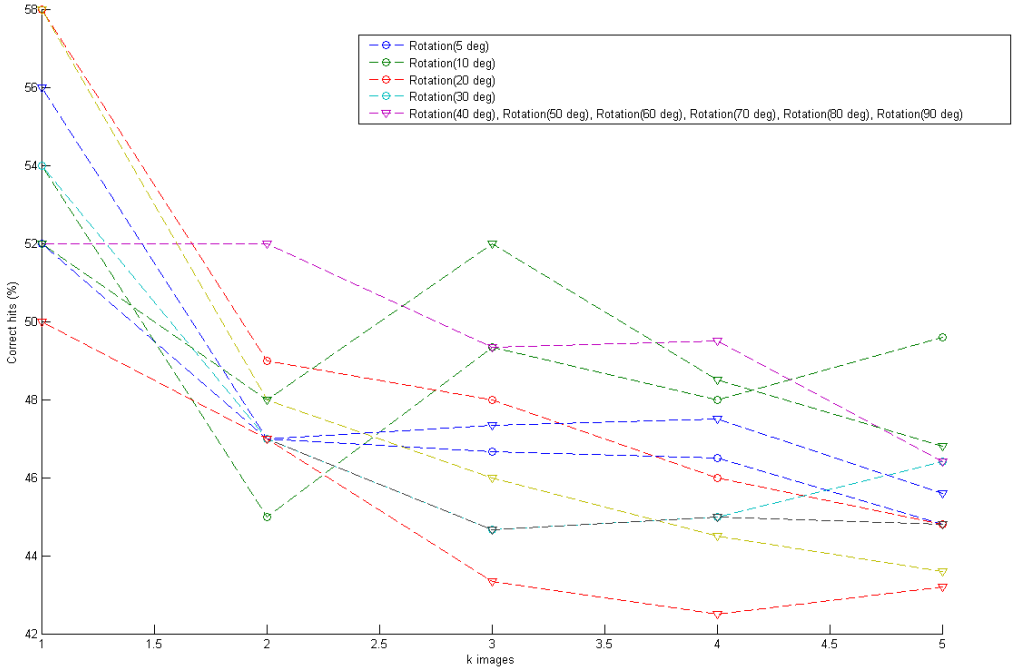
Στην περιγραφή του πειράματος αναφέρουμε, ότι θέλοντας να επεκτείνουμε το 2ο πείραμα πήραμε τις εικόνες από το σύνολο *Query2* και τις μετασχηματίσαμε σε αυτές του *Query3*. Τα αποτελέσματα που πήραμε παρουσιάζονται στις επόμενες γραφικές παραστάσεις, Εικόνες 6.12-6.20. Σχετικά με τους χρόνους αναζήτησης δεν φαίνεται να υπάρχει διαφορά συγκρίνοντάς τους με αυτούς του πρώτου πειράματος. Τα αποτελέσματα παρουσιάζονται στις παρακάτω γραφικές παραστάσεις, Εικόνες 6.18-6.21.



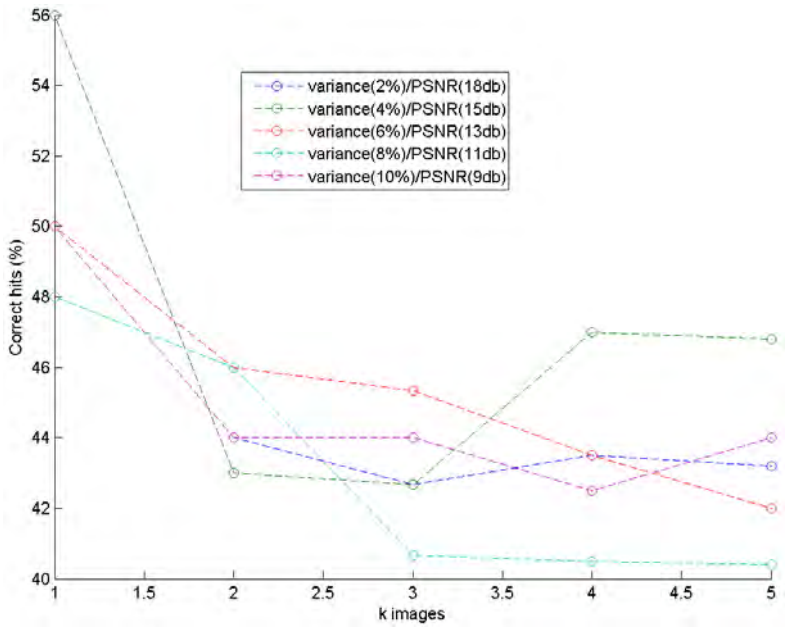
Εικόνα 6.12 Depth only Search Noise



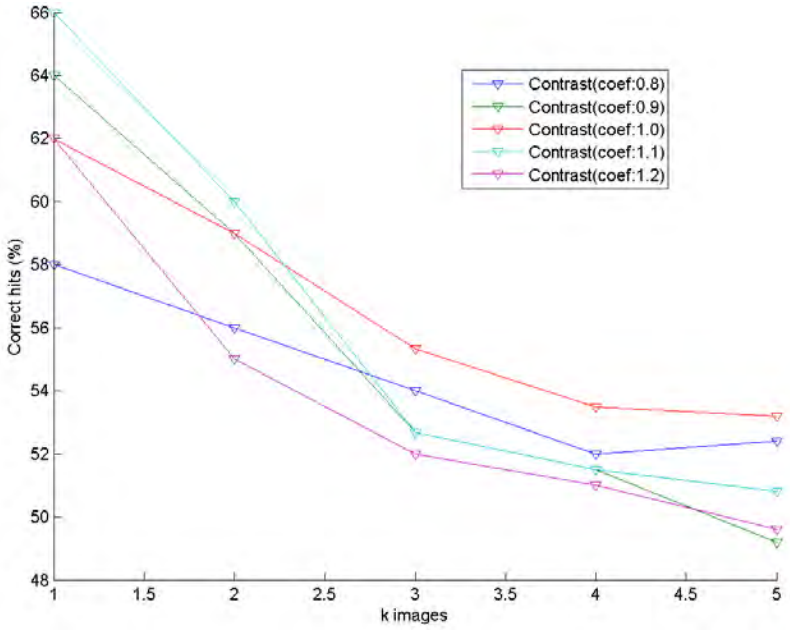
Εικόνα 6.13 Depth only Search Contrast Stretching



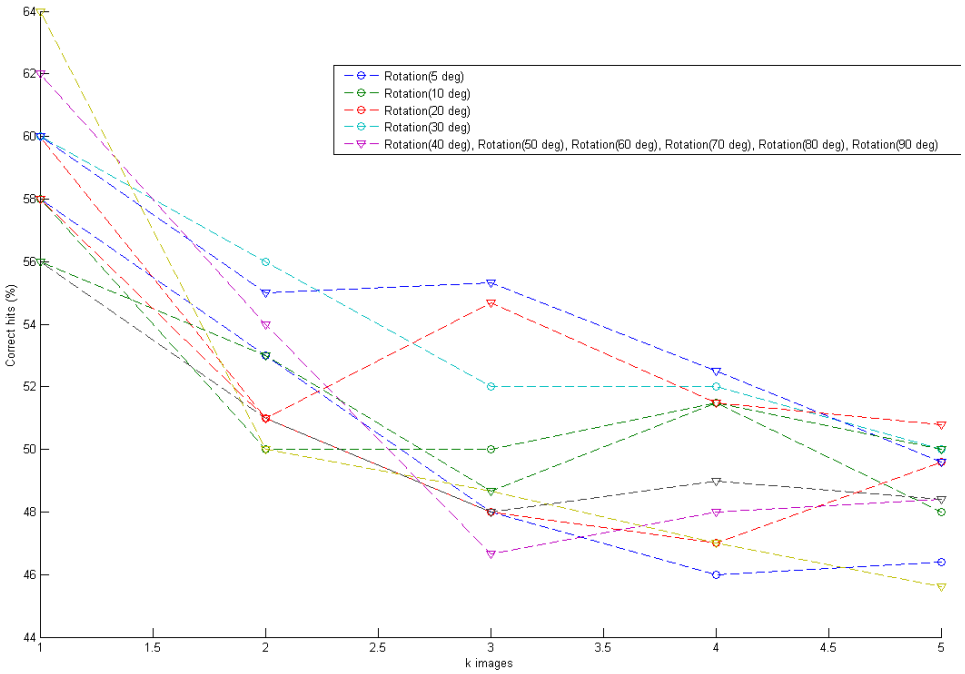
Εικόνα 6.14 Depth only Search Rotation



Εικόνα 6.15 Complete Fast NN Noise

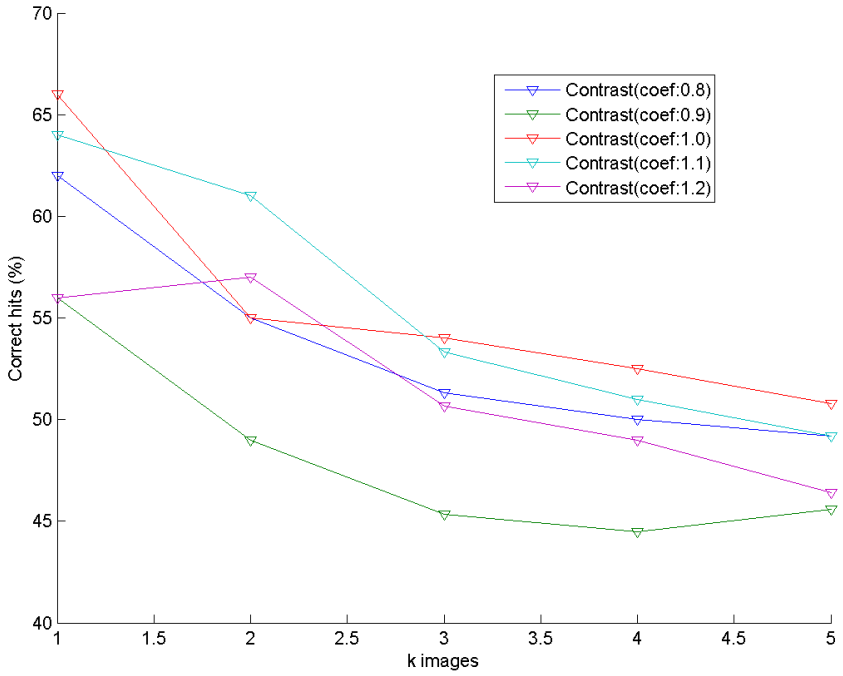


Εικόνα 6.16 Complete Fast NN Contrast Stretching

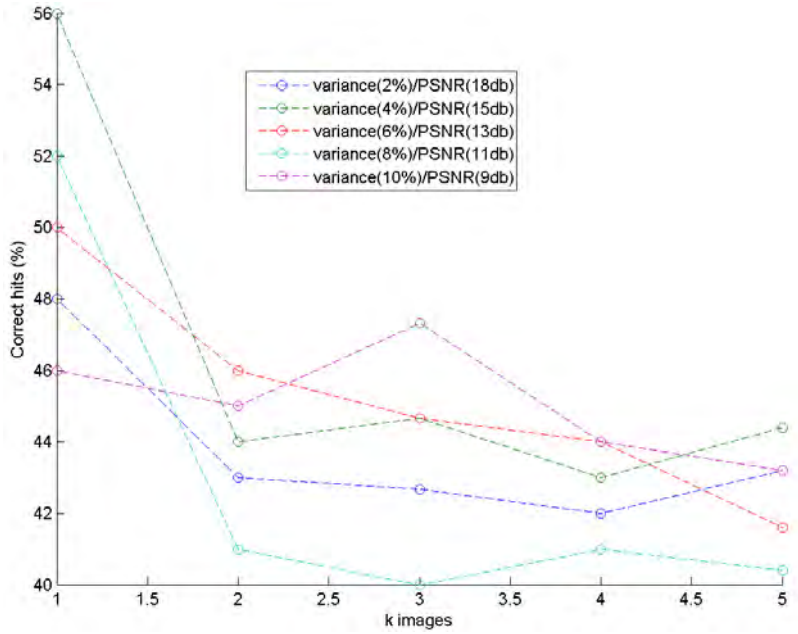


Εικόνα 6.17 Complete Fast NN Rotation

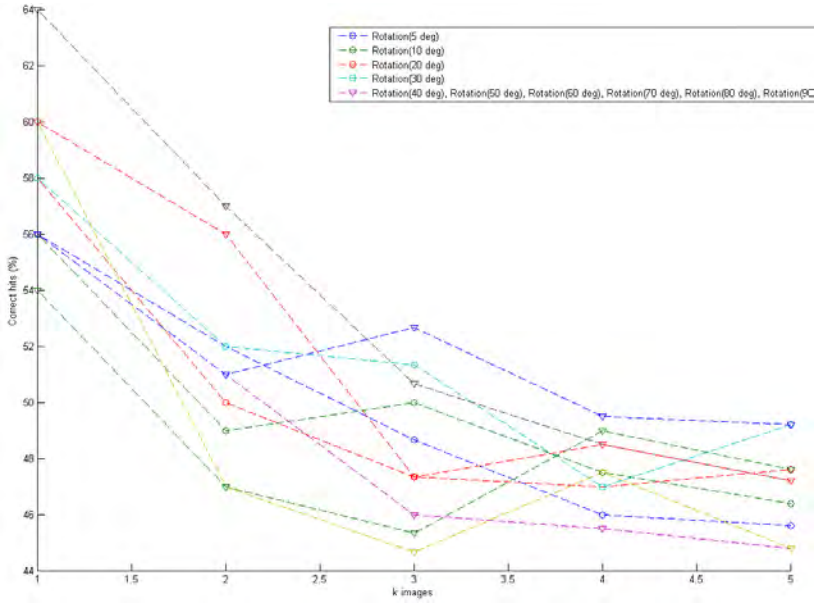
Τα αποτελέσματα ήταν σχεδόν ίδια για όλες τις αλλαγές που έγιναν στα *backtracking*. Έτσι, στην Εικόνα 6.18 παρουσιάζονται ενδεικτικά οι γραφικές παραστάσεις για 4000 *backtracking.*



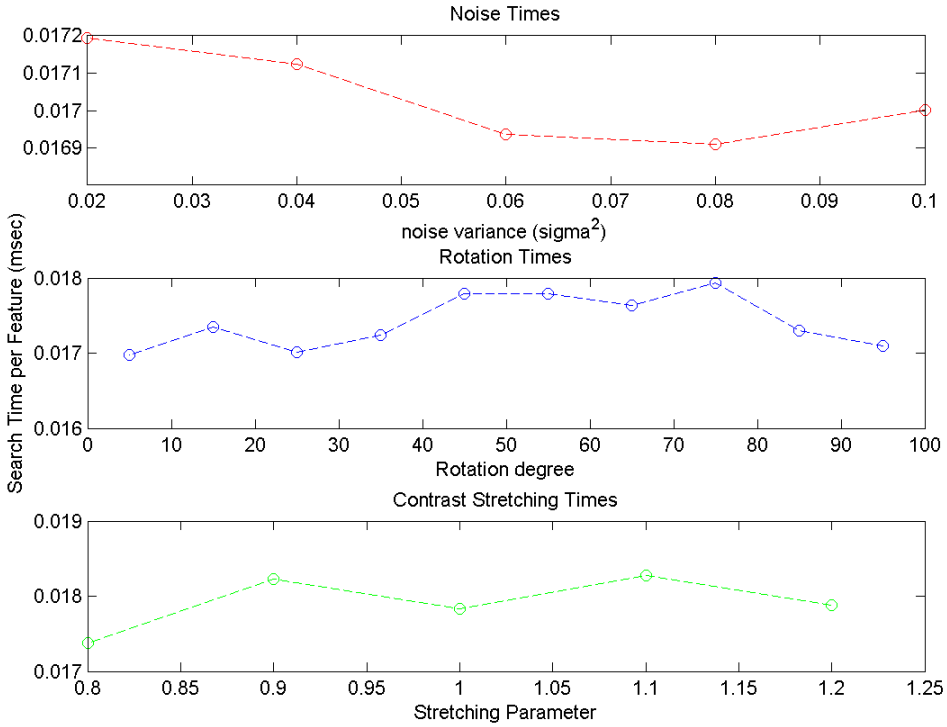
Εικόνα 6.18 Limited Fast NN Contrast Stretching for 4000 *Backtracking*



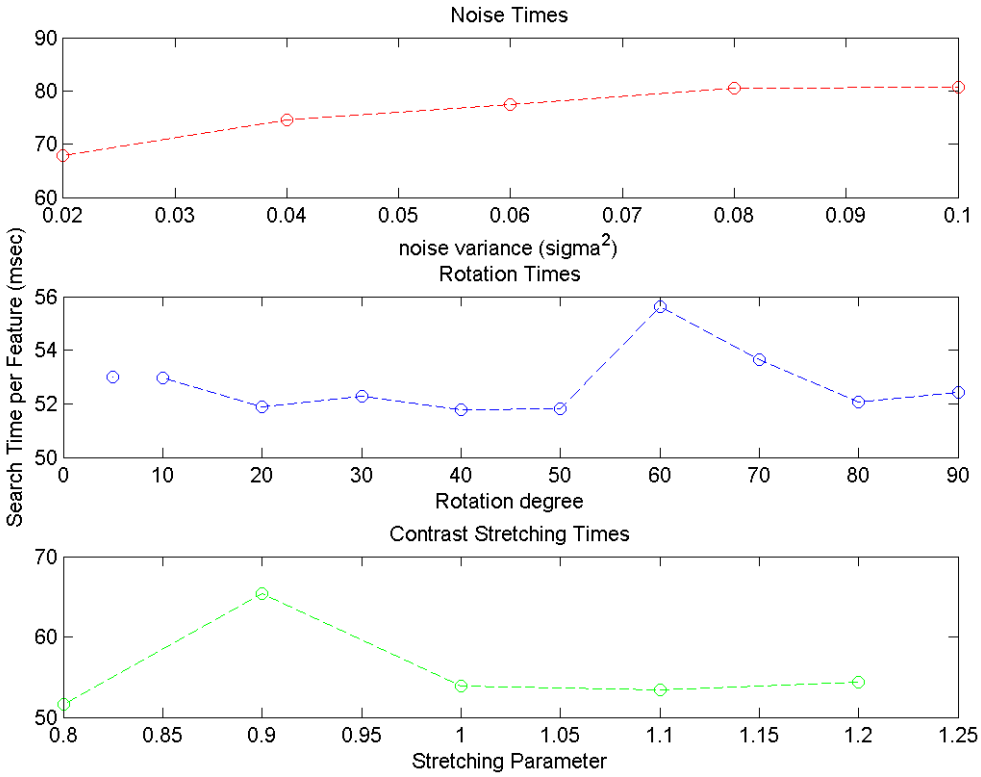
Εικόνα 6.19 Limited Fast NN Noise for 4000 *Backtracking*



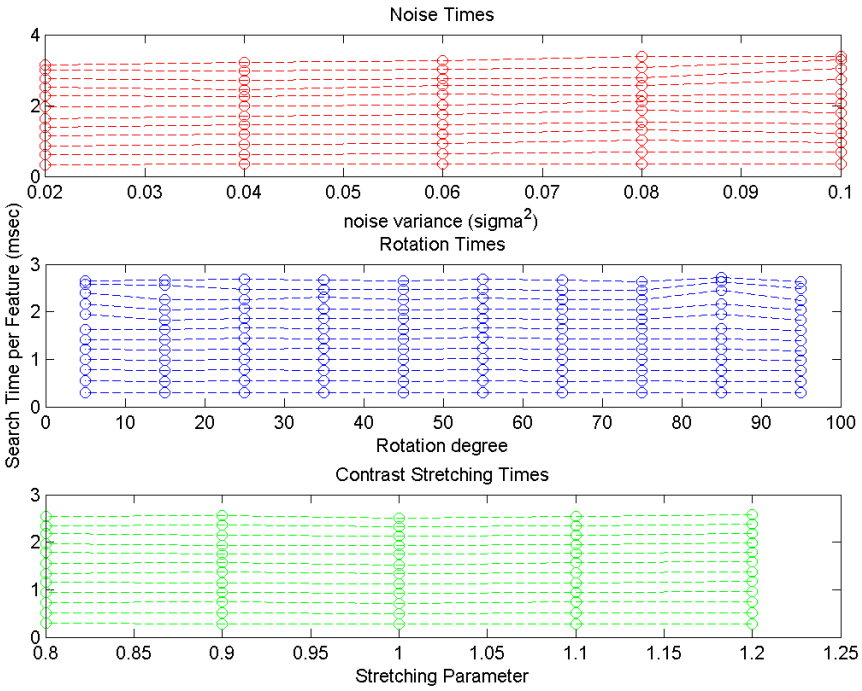
Εικόνα 6.20 Limited Fast NN Rotation for 4000 *Backtracking*



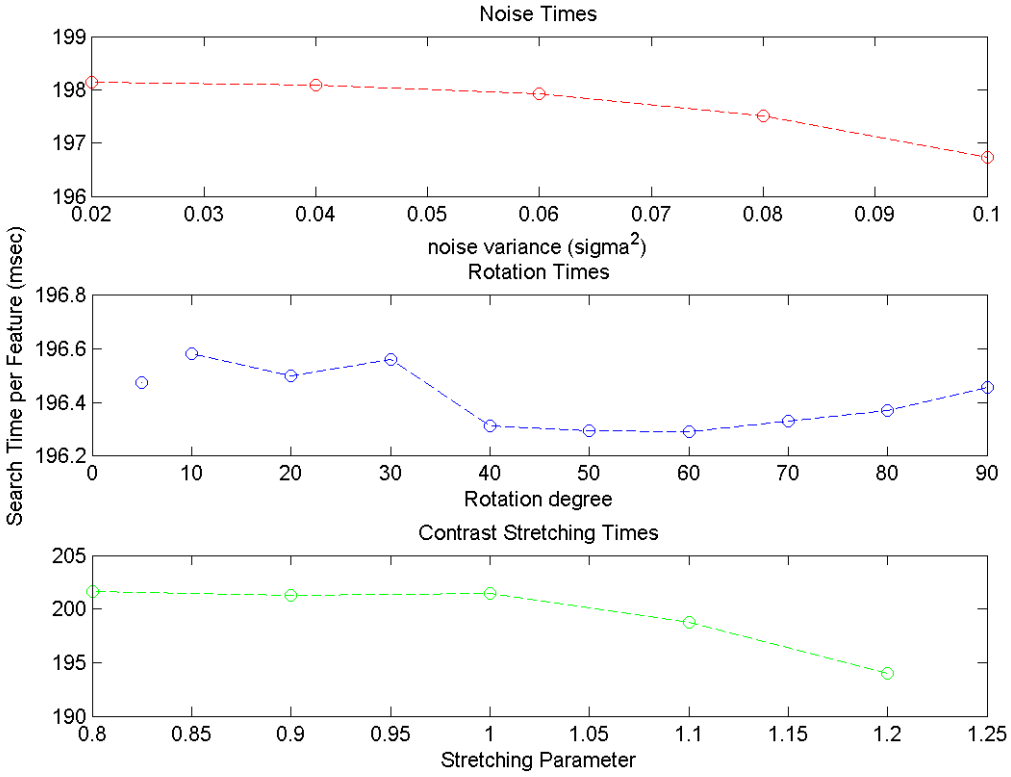
Εικόνα 6.21 Times Depth only Fast NN



Εικόνα 6.22 Times Complete Fast NN



Εικόνα 6.23 Times Limited Fast NN



Εικόνα 6.24 Times Full NN Search

Όπως είναι φυσικό, τα ποσοστά των σωστών επιστροφών μειώθηκαν διότι είναι πιο δύσκολη η αναγνώριση των περιεχομένων όταν οι εικόνες έχουν υποστεί μετασχηματισμούς. Επίσης, παρατηρώντας τους χρόνους βλέπουμε μια μικρή αύξηση στο χρόνο αναζήτησης με τη μέθοδο Complete Fast NN. Εδώ πρέπει να σημειωθεί, ότι μόνο με τη χρήση της αναζήτησης του Κοντινότερου Γείτονα και χωρίς καμία άλλη επεξεργασία της εικόνας πήραμε τα παραπάνω αποτελέσματα.

6.1.3 Πείραμα 3ο

Στο εν λόγο πείραμα χρησιμοποιήσαμε τις εικόνες του συνόλου *Query4*. Σκοπός του πειράματος ήταν να δείξει ότι ο χρόνος αναζήτησης αυξάνεται, όταν γίνει η αναζήτηση μίας εικόνας διαφορετικής κατηγορίας από αυτές που είναι στη βάση δεδομένων. Τα αποτελέσματα για τις μεθόδους DOS και Limited είναι τα ίδια με προηγούμενες γραφικές παραστάσεις χρόνων, Εικόνες 6.21, 6.23. Η αναζήτηση με την DOS έχει κατά μέσο όρο τον ίδιο χρόνο για όλες τις περιπτώσεις, αφού πάντα η αναζήτηση τερματίζει όταν φτάσουμε σε φύλλο. Επίσης, με τη Limited μέθοδο θέτουμε έναν συγκεκριμένο αριθμό *backtracking* και έτσι πάλι περιορίζεται ο χρόνος αναζήτησης. Με βάση τα παραπάνω, μπορούμε να εξετάσουμε την συμπεριφορά μόνο της Complete Fast NN μεθόδου, η οποία βρίσκει τον επακριβή κοντινότερο γείτονα. Στον Πίνακα 6.6 αναφέρεται ο μέσος όρος των *backtracking* που χρειάζεται να εκτελέσει ο Complete Fast NN για κάθε μετασχηματισμό και δίπλα ο μέσος χρόνος αναζήτησης ενός διανύσματος. Τα τρία πρώτα δεδομένα αναζήτησης του Πίνακα 6.6 είναι όταν αναζητήσαμε εικόνες από το *Query1* σύνολο στη βάση δεδομένων. Το τελευταίο είναι όταν αναζητήσαμε τις εικόνες του συνόλου *Query4*.

Πίνακα 6.6		
Transformation	**Backtracking**	**Times**
Noise	~172K	~71msec
Rotation	~72K	~25msec
Contrast Stretching	~39K	~13msec
No Transformation		
Query4	~150K	~63msec

Παρατηρούμε, ότι όταν έχουμε θόρυβο παρόλο που η εικόνα υπάρχει μέσα στη βάση δεδομένων τα *backtracking* που χρειάζεται η Complete μέθοδος για να βρει τον επακριβή κοντινότερο γείτονα είναι πολύ περισσότερα απ’ ότι για τους άλλους μετασχηματισμούς.

Αυτή η μεγάλη διαφορά στον αριθμό των *backtracking* οφείλεται στο γεγονός ότι ο θόρυβος που έχουμε προσθέσει στην εικόνα είναι πολύ μεγάλος και έτσι η εικόνα γίνεται σχεδόν αγνώριστη(PSNR:18db($\sigma^2 = 0.08$)-9db($\sigma^2 = 0.1$)). Στο πείραμα 1 όμως φάνηκε ότι μπορούμε ακόμη και έτσι να την αναγνωρίσουμε. Οι εικόνες του *Query4* συνόλου δεν περιέχουν θόρυβο. Βλέπουμε ότι ο αριθμός των *backtracking* που χρειάζεται η Complete για αναζήτηση ενός διανύσματος των εικόνων του *Query4* συνόλου, είναι μικρότερος από ότι για τα διανύσματα του *Query1*, στις οποίες έχει προστεθεί θόρυβος. Επομένως, μπορούμε να υποθέσουμε ότι η Complete μέθοδος, δεν έχει τόσο καλή απόδοση από άποψη ταχύτητας παρουσία τόσο πολύ θορύβου.

Συγκρίνοντας όμως τα αποτελέσματα για τις εικόνες του *Query1* που δεν έχουν θόρυβο (*Rotation, Contrast Stretching*) και την αναζήτηση των εικόνων του *Query4*, βλέπουμε ότι τα *backtracking* που χρειάζονται για να βρεθεί ο επακριβής κοντινότερος γείτονας είναι πολύ λιγότερα όταν η εικόνα υπάρχει στη βάση δεδομένων.

Παρόμοια αποτελέσματα έχουν ήδη παρατηρηθεί στο πρόβλημα της αναγνώρισης χειρονομιών (*gesture recognition*), όπου βρέθηκε ότι δεδομένα χειρονομιών με προσθετικό θόρυβο ή κατηγορίας εκτός λεξιλογίου (*out-of-vocabulary*) προκαλούν μεγαλύτερο αριθμό *backtracking* σε σχέση με "καθαρά" δεδομένα εισόδου [3].

Κεφάλαιο 7

Συμπεράσματα και μελλοντική εργασία

Στην παρούσα εργασία μελετήθηκε και υλοποιήθηκε ένα σύστημα αναζήτησης και ανάκτησης εικόνων. Κάνοντας την υλοποίηση του αλγορίθμου εξαγωγής χαρακτηριστικών SIFT σε γλώσσα προγραμματισμού C, μελετήθηκαν και κατανοήθηκαν οι παράμετροι που διέπουν έναν τέτοιο πολύπλοκο αλγόριθμο και τον κάνουν τόσο διαδομένο σε εφαρμογές υπολογιστικής όρασης.

Η κατανόηση και η χρήση του αλγορίθμου Fast NN [2, 3] μας έδειξε τη λειτουργία ενός αλγορίθμου αναζήτησης κοντινότερου γείτονα (Nearest Neighbor). Επίσης, σημαντικό είναι να αναφερθεί ότι πρώτη φορά μελετήθηκε η συμπεριφορά του για αναζήτηση διανυσμάτων άνω των 64-διαστάσεων και έγινε χρήση του σε ένα πρόβλημα ανάκτησης εικόνας. Οι χρόνοι για μία exact Fast NN αναζήτηση διανυσμάτων 128-διαστάσεων δεν ήταν πολύ ταχύτεροι από αυτούς της exact Full NN. Οι προσεγγιστικές μέθοδοι, όμως, είναι ένα ελκυστικό μέρος αφού έκτος από ταχύτητα παρουσιάζουν και ακρίβεια.

Η μέθοδος αξιοποίησης των αποτελεσμάτων της αναζήτησης NN που προτείνεται, μας έδωσε υψηλά ποσοστά σωστών ανακτήσεων όταν μετασχηματίσαμε μία εικόνα της βάσης δεδομένων και στη συνέχεια την αναζητήσαμε. Τα αποτελέσματα όμως για το πρόβλημα κατηγοριοποίησης δεν ήταν τόσο ικανοποιητικά. Αυτό οφείλεται σε μεγάλο βαθμό στις εικόνες που χρησιμοποιήθηκαν, αφού πολλές δεν περιείχαν αποκλειστικά μία κατηγορία αντικειμένου.

Συνοψίζοντας, κρίσιμό σημείο για το χρόνο ανάκτησης μίας εικόνα είναι η εξαγωγή των χαρακτηριστικών αλλά και η αναζήτησή της μέσα στη βάση δεδομένων. Δεδομένης μίας *query* εικόνας διαστάσεων 256x256 και μίας βάσης δεδομένων 250 εικόνων, ο χρόνος από τη στιγμή από θα μπει στο σύστημα μέχρι να ανακτηθεί είναι περίπου 1.0 second με τη μέθοδο DOS. Στο χρόνο περιλαμβάνεται όλη η διαδικασία εξαγωγή, αναζήτηση και ανάκτηση.

Μία πρόταση για επέκταση της παρούσας εργασίας, για ταχύτερη ανάκτηση, θα μπορούσε να είναι η χρήση εντολών παραλληλισμού και η συγγραφή πιο αποδοτικού κώδικα για την εξαγωγή των χαρακτηριστικών SIFT. Για μεγαλύτερη ακρίβεια αποτελεσμάτων, μπορεί να γίνει η χρήση και άλλων δεδομένων των SIFT χαρακτηριστικών, όπως η σχετική θέση που έχουν τα χαρακτηριστικά μεταξύ τους πάνω στην ίδια εικόνα.

Επίσης, για πετύχουμε καλύτερα αποτελέσματα κατηγοριοποίησης πρέπει να γίνουν περαιτέρω πειράματα και με άλλες κατηγορίες. Η μελέτη ενός Fast kNN αλγορίθμου στο

πρόβλημα της ανάκτησης, θα μπορούσε να αυξήσει τα ποσοστά επιτυχημένων επιστροφών.

Τέλος, θα μπορούσε να δημιουργηθεί μία βάση δεδομένων που θα αρχικοποιείται ξανά κάθε φορά που γίνεται η αναζήτηση μίας εικόνας πετυχαίνοντας έτσι τι δημιουργία ενός αυτοδιαχειριζόμενου συστήματος.

Κεφάλαιο 8

Βιβλιογραφία/References

[1] David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision*, 60, 2 (2004), pp. 91-110.

[2] I. Katsavounidis, C.-C.J. Kuo, and Zhen Zhang." **Fast tree-structured nearest neighbor encoding for vector quantization**". IEEE Trans. on Image Processing, 5(2):398-404, 1996.

[3] Stergios Poularakis, "Low complexity hand gesture recognition", Ph.D. Thesis, University of Thessaly, 2014.

[4] S. Lloyd. Least squares quantization in pcm. IEEE Trans. on Information Theory, 28(2):129{137, 1982.

[5] Arya, S., and Mount, D.M. 1993. Approximate nearest neighbor queries in fixed dimensions. In Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93), pp. 271-280.

[6] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., and Wu, A.Y. 1998. An optimal algorithm for approximate nearest neighbor searching. Journal of the ACM, 45:891-923.

[8] http://en.wikipedia.org/wiki/K-d_tree

[9] Jeffrey S. Beis and David G. Lowe , **"Shape indexing using approximate nearest-neighbour search in high-dimensional spaces,"** *Conference on Computer Vision and Pattern Recognition*, Puerto Rico (June 1997), pp.1000-1006.

[10] Friedman, J.H., Bentley, J.L. and Finkel, R.A. 1977. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, 3(3):209-226.

[11] http://en.wikipedia.org/wiki/Image_gradient

[12] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/index.html

[13] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In 47th Annual IEEE Symposium on Foundations of Computer Science, pages 459-468, 2006.

[14] T.B. Sebastian and B.B. Kimia. Metric-based shape retrieval in large databases. In 16th Int'l Conf. on Pattern Recognition, volume 3, pages 291-296, 2002.

[15] A. Guttman. R-trees: a dynamic index structure for spatial searching, volume 14.ACM, 1984.

[16] J. McNames. A fast Nearest-Neighbor algorithm based on a principal axis search tree. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 23(9):964-976, 2001.

[17] GONZALEZ, Rafael C.; WOODS, Richard E. Digital image processing, 3nd.*SL: Prentice Hall*, 2008, 3.

[18] Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce. "A sparse texture representation using affine-invariant regions." *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 2. IEEE, 2003.

[19] Kim, Sungho, Kuk-Jin Yoon, and In So Kweon. "Object recognition using a generalized robust invariant feature and Gestalt's law of proximity and similarity."*Pattern Recognition* 41.2 (2008): 726-741.

[20] M. Muja and D.G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 36(11):2227{2240, 2014.

[21] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In 47th Annual IEEE Symposium on Foundations of Computer Science, pages 459{468, 2006.

[22] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parametersensitive hashing. In IEEE Int'l Conf. on Computer Vision (ICCV), pages 750-757, 2003.

[23] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In IEEE Int'l Conf. on Computer Vision (ICCV), pages 2130-2137, 2009.

[24] J. He, W. Liu, and S. F. Chang. Scalable similarity search with optimized kernel hashing. In 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pages 1129-1138, 2010.

[25] http://en.wikipedia.org/wiki/Image_gradient

[26] Koenderink, J.J. 1984. The structure of images. Biological Cybernetics, 50:363-396.

[27] Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.

[28] Lowe, D.G. 1999. Object recognition from local scale-invariant features. In International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157.

[29] Mikolajczyk, K. 2002. Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France.

[30] Brown, M. and Lowe, D.G. 2002. Invariant features from interest point groups. In British Machine Vision Conference, Cardiff, Wales, pp. 656-665.

[31] http://en.wikipedia.org/wiki/Hessian_matrix

[32] http://www.libtiff.org/

[33] http://en.wikipedia.org/wiki/Grayscale

[34] http://www.cs.ubc.ca/~lowe/keypoints

[35] http://en.wikipedia.org/wiki/Canny_edge_detector

[36] http://en.wikipedia.org/wiki/Sobel_operator

[37] http://en.wikipedia.org/wiki/Corner_detection

[38] http://en.wikipedia.org/wiki/Features_from_accelerated_segment_test

[39] http://en.wikipedia.org/wiki/Blob_detection#The_Laplacian_of_Gaussian

[40] http://en.wikipedia.org/wiki/Difference_of_Gaussians

[41] http://en.wikipedia.org/wiki/Maximally_stable_extremal_regions

[42] http://en.wikipedia.org/wiki/Principal_curvature-based_region_detector

[43] http://en.wikipedia.org/wiki/Hough_transform

[44] http://en.wikipedia.org/wiki/Structure_tensor

[45] http://en.wikipedia.org/wiki/SURF