ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΗΡΑΚΛΕΙΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του σπουδαστή

ΣΩΤΗΡΗ ΧΟΛΕΒΑ

ΜΕ ΘΕΜΑ:

Ρομποτική πλοήγηση και καταγραφή δεδομένων βασιζόμενη σε κάμερα

Επιβλέπων : Δρ. Καβουσανος Μανώλης
Καθηγητής του Τμήματος

Μηχανολογίας του Τ.Ε.Ι. Ηρακλείου

Κρήτη, Ιούνιος 2010

Περιεχόμενα

Εισαγωγή

Τα τελευταία χρόνια ο κλάδος της ρομποτικής γνωρίζει αρκετά μεγάλη άνθηση. Σε αυτή την εργασία θα εξετάσουμε κάποιες βασικές και έπειτα θα έναν συγκεκριμένο τομέα του, αυτόν της ρομποτικής πλοήγησης.

Συγκεκριμένα στο πρώτο κεφάλαιο θα ασχοληθούμε με κάποιες γενικές έννοιες της ρομποτικής και τις κυριότερες κατηγορίες ρομπότ που χρησιμοποιούνται σήμερα. Έπειτα στο δεύτερο κεφάλαιο θα αναλυθεί η ρομποτική πλοήγηση και οι κυριότεροι τομείς της, επικεντρωνόμενοι στο τρίτο κεφάλαιο στο πρόβλημα του SLAM. Στο τέταρτο κεφάλαιο θα αναλύσουμε τους αισθητήρες που χρησιμοποιούν τα ρομπότ εκτός από τα οπτικά αισθητήρια. Στο πέμπτο κεφάλαιο θα αναλυθούν διάφορες τεχνικές που χρησιμοποιούνται για την ρομποτική πλοήγηση με κάμερα αναλύοντας και τα οπτικά αισθητήρια και θα καταλήξουμε στο τελευταίο κεφάλαιο με την ανάλυση αλγορίθμων σύγκρισης εικόνων που χρησιμοποιούνται συνήθως στο χώρο της ρομποτικής πλοήγησης.

Ρομποτική

Ρομποτική είναι η επιστήμη που ασχολείται με την κατασκευή και την λειτουργία των ρομπότ. Αν και από τα αρχαία χρόνια η ανθρώπινη φαντασία είχε συλλάβει κάποιες μηχανές με μεγαλύτερο παράδειγμα τον Τάλω, έναν χάλκινο γίγαντα που φύλαγε την Κρήτη τα χρόνια του Μίνωα μόνο τα τελευταία χρόνια η ρομποτική έχει αρχίσει να αναπτύσσεται ραγδαία.

Πλεονεκτήματα χρήσης ρομπότ

Ένα ρομπότ είναι μία μηχανή που χρησιμοποιείται για να αντικαταστήσει τον άνθρωπο σε κάποιες εργασίες. Συνήθως η χρήση ρομπότ για κάποια εργασία προσφέρει μεγαλύτερη απόδοση από την χρήση ανθρώπων. Στην βιομηχανία τα ρομπότ μπορούν να κάνουν μία εργασία αρκετά πιο γρήγορα από ότι οι εργάτες και με πολύ μεγαλύτερη ακρίβεια, χωρίς να παραπονιούνται για την συνεχή και μονότονη επανάληψή της. Επίσης η χρήση τους είναι πολύ πιο οικονομική από την χρήση εργατών, ενώ οι δυνατότητές τους είναι πολύ μεγαλύτερες, πχ μπορούν να σηκώσουν αρκετά μεγαλύτερο βάρος.

Ένας άλλος λόγος που επιβάλλει την χρήση ρομπότ είναι κάποιες επικίνδυνες εργασίες ή οι πολύ δύσκολες συνθήκες που μπορεί να επικρατούν σε κάποιο μέρος. Συνήθως χρησιμοποιούνται στην βιομηχανία, πχ σε χυτήρια σιδήρου ή σε ερευνητικά εργαστήρια, αλλά υπάρχουν και άλλες ακραίες περιπτώσεις όπως είναι η εξερεύνηση του διαστήματος ή των βυθών των θαλασσών.

Κατηγορίες ρομπότ

Κατά την εξέλιξη της ρομποτικής έχουν προκύψει διάφορες κατηγορίες ρομπότ, ανάλογα με το πόσο προηγμένη ήταν η τεχνολογία κάθε εποχή αλλά και με το τι έπρεπε να κάνει κάθε ρομπότ. Κατά συνέπεια υπάρχουν διάφορες κατηγορίες και είδη ρομπότ, τα βασικότερα από τα οποία θα αναφέρουμε παρακάτω.

Ελευθερία κίνησης

Ανάλογα με το πώς και κατά πόσο μπορεί να κινηθεί ένα ρομπότ μπορεί να μπει σε μία συγκεκριμένη κατηγορία.

Ρομπότ σταθερής βάσης

Παρόλο που οι περισσότεροι άνθρωποι στο άκουσμα της λέξης ρομπότ σκέφτονται ένα κατασκεύασμα που τους μοιάζει ο κανόνας είναι πως τα περισσότερα ρομπότ που κατασκευάζονται και αναπτύσσονται σήμερα δεν έχουν καμία σχέση στην εμφάνισή τους και στις δυνατότητές τους με τους ανθρώπους και στις περισσότερες περιπτώσεις απλά μιμούνται μόνο ένα ή και κανένα μέλος του ανθρώπινου σώματος. Το πιο χαρακτηριστικό παράδειγμα είναι οι ρομποτικοί βραχίονες που χρησιμοποιούνται στις γραμμές παραγωγής στην βαριά βιομηχανία, οι οποίοι

μιμούνται ένα ανθρώπινο χέρι. Αυτά τα ρομπότ ονομάζονται *ρομπότ σταθερής βάσης*, και πρόκειται για μηχανές οι οποίες δεν μπορούν να μετακινηθούν στο χώρο. Συνήθως αυτά κουνάνε ένα συγκεκριμένο μέλος τους ώστε να κάνουν κάποιες εργασίες.

Κινούμενα ρομπότ

Μία άλλη μεγάλη κατηγορία ρομπότ είναι τα *κινούμενα ρομπότ*, δηλαδή ρομπότ που μπορούν να μετακινηθούν. Πρόκειται για μία αρκετά ευρεία κατηγορία γιατί η ελευθερία και ο τρόπος κίνησης τους διαφέρει κατά πολύ. Για παράδειγμα μπορεί να πρόκειται για ρομποτικά αεροπλάνα που κινούνται νε την χρήση κινητήρων jet, ρομποτικά πλοία και υποβρύχια που κινούνται με προπέλες ή τροχοφόρα ρομπότ. Υπάρχουν ακόμη και κάποιες άλλες ειδικές κατηγορίες που αναφέρουμε, συγκεκριμένα:

Αυτόματα καθοδηγούμενα οχήματα: Τα αυτόματα καθηγούμενα οχήματα (AGV, Automated Guided Vehicles) είναι τα ρομπότ που μπορούν να κινηθούν μόνο μέσα σε μία συγκεκριμένη πορεία που μπορεί να την δείχνουν κάποιες σιδηροτροχιές, καλώδια ή πομποί. Ένα ρομποτικό τρένο είναι ένα τέτοιο παράδειγμα, όπου δεν αποφασίζει που θα πάει αλλά μόνο κατά πόσο θα επιταχύνει ή θα φρενάρει.

Βαδίζοντα ρομπότ: Πρόκειται για τα ρομπότ που μοιάζουν περισσότερο στον άνθρωπο (αν έχουν δύο πόδια) ή και σε κάποιους άλλους οργανισμούς, αφού υπάρχουν ρομπότ με 4 ή και 6 πόδια. Αν και τα πόδια τους δίνουν την δυνατότητα να κινηθούν σε ανώμαλο έδαφος ή σκαλοπάτια, ο συγχρονισμός των ποδιών ώστε να ισορροπεί το ρομπότ και να προχωράει χωρίς να σταματάει στις ανωμαλίες του εδάφους είναι πάρα πολύ δύσκολος.

Βαθμός αυτονομίας

Ένα ρομπότ δεν είναι απαραίτητο ότι μπορεί να σκέφτεται και να παίρνει αποφάσεις σχετικά με το τι πρέπει να κάνει. Ανάλογα με το πόσες και ποιες πρωτοβουλίες πρέπει να πάρει είναι διακρίνεται σε διάφορες κατηγορίες.

Μία μηχανή που προγραμματίζεται για να κάνει μία δουλειά και μετά την κάνει χωρίς καμία τροποποίηση δεν παίρνει πρακτικά καμία πρωτοβουλία. Το ίδιο ισχύει και για τα ρομπότ τα οποία καθοδηγούνται πλήρως από έναν άνθρωπο που τα χειρίζεται.

Όμως μπορεί να υπάρχουν διάφορες μορφές αυτονομίας. Μία μηχανή μπορεί να παίρνει γενικές οδηγίες από έναν άνθρωπο για να εκτελέσει μία συγκεκριμένη εργασία και μετά την εκτελεί μόνος του χωρίς να πρέπει να του δοθούν οδηγίες για κάθε μικρή λεπτομέρεια. Ένα παράδειγμα είναι ένας προηγμένος ρομποτικός βραχίονας σε ένα εργοστάσιο. Το ρομπότ χρειάζεται να μπορεί να καταλάβει που ακριβώς βρίσκεται το αντικείμενο της δουλειάς του σε σχέση με το άκρο του

βραχίονα και να τον μετακινήσει κατάλληλα ώστε να έρθει ακριβώς στο κατάλληλο σημείο που έχει προγραμματιστεί ώστε πχ να βιδώσει μία βίδα ή να ανοίξει μία τρύπα, ενώ κάποια προηγμένα μοντέλα αναγνωρίζουν το εξάρτημα το οποίο έχουν μπροστά τους και ανάλογα εκτελούν την προγραμματισμένη εργασία.

Επίσης τα ρομποτικά αεροπλάνα είναι μία κατηγορία ρομπότ τα οποία γενικά παίρνουν κάποιες πρωτοβουλίες. Συγκεκριμένα μπορεί να τους δοθεί μία οδηγία να μεταβούν από το σημείο που βρίσκονται σε ένα άλλο, χωρίς περαιτέρω λεπτομέρειες. Το ρομπότ πρέπει να πάρει την κατάλληλη πορεία λαμβάνοντας υπόψη του αρκετές παραμέτρους όπως το ύψος που πρέπει να ταξιδέψει ή της καιρικές συνθήκες οι οποίες μπορεί να το κάνουν να αλλάξει πορεία. Σε αυτή την περίπτωση το ρομπότ παίρνει μία γενική οδηγία και έπειτα πρέπει να αποφασίσει ποιος είναι ο καλύτερος τρόπος για αν την εκτελέσει.

Αυτόνομα ρομπότ

Μία ειδική κατηγορία ρομπότ σε αυτό τον τομέα είναι τα αυτόνομα ρομπότ. Πρόκειται για ρομπότ τα οποία, όπως υπονοεί και το όνομά τους χρειάζονται όσο το δυνατόν λιγότερες οδηγίες για να επιτελέσουν μία εργασία. Καθώς η ρομποτική πλοήγηση σχετίζεται κυρίως με τα αυτόνομα ρομπότ, θα αναλύσουμε διεξοδικά τα κριτήρια που πρέπει να έχει ένα ρομπότ για να ανήκει σε αυτή την κατηγορία. Κάποια χαρακτηριστικά ενός αυτόνομου ρομπότ είναι τα παρακάτω:

Αυτοσυντήρηση: Το ρομπότ πρέπει να μπορεί να προσέξει τον εαυτό του. Εδώ η έννοια της συντήρησης αναφέρεται στο να μπορεί να επιτελέσει κάποιες εργασίες ώστε να συνεχίσει να λειτουργεί, για παράδειγμα να αντιλαμβάνεται αν δεν έχει αρκετή ενέργεια και να φορτίσει μόνο του τους συσσωρευτές του. Επίσης χρειάζεται να μπορεί να αντιλαμβάνεται πια μέρη είναι επικίνδυνα και να τα αποφεύγει. Κάτι τέτοιο μπορεί να γίνει με την χρήση αισθητήρων θερμοκρασίας, πίεσης ή υγρασίας ώστε να μην πλησιάζει τυχόν επικίνδυνα μέρη.

Αίσθηση του περιβάλλοντος: για να μπορεί ένα ρομπότ να λειτουργήσει πρέπει να μπορεί να αντιλαμβάνεται το περιβάλλον του. Σε αντίθεση με την προηγούμενη περίπτωση όπου το ρομπότ χρειαζόταν να έχει αισθητήρες για διάφορα ερεθίσματα στον εαυτό του, τώρα πρέπει να έχει αισθητήρες για το περιβάλλον του. Περισσότερα για αυτό θα δούμε παρακάτω, όπου θα αναλύσουμε διεξοδικά τους αισθητήρες που πρέπει να έχει ένα ρομπότ.

Επιτέλεση εργασιών: Για να είναι χρήσιμο ένα ρομπότ πρέπει να μπορεί να επιτελέσει και κάποιες εργασίες που του έχουν δοθεί. Αν και ένα χαρακτηριστικό παράδειγμα αυτόνομου ρομπότ είναι τα ρομπότ-ηλεκτρικές σκούπες πού ήδη πωλούνται εμπορικά είναι εξαιρετικά χρήσιμη η δυνατότητα επιτέλεσης εργασιών αν πληρούνται κάποιες συγκεκριμένες

συνθήκες, για παράδειγμα ένα ρομπότ-καθαριστής να καθαρίζει μόνο τις επιφάνειες που είναι ήδη βρώμικες.

Εύρεση θέσης και πλοήγηση στο χώρο: Για να μπορεί να είναι πραγματικά αυτόνομο ένα ρομπότ πρέπει να μπορεί να κινηθεί αποτελεσματικά στον χώρο ώστε να επιτελέσει τις εργασίες του. Περισσότερα για αυτό το θέμα θα δούμε παρακάτω, όπου και θα αναλύσουμε διεξοδικά το θέμα της ρομποτικής πλοήγησης.

*Εύρεση ενέργεια*ς: Ένα ρομπότ πρέπει να είναι σε θέση να βρει μόνο του ενέργεια όταν κινδυνεύει να απενεργοποιηθεί. Αυτό δεν είναι πάντα εύκολο, ειδικά για ρομπότ που λειτουργούν σε εξωτερικούς χώρους και μακριά από εξειδικευμένες εγκαταστάσεις.

Ρομποτική πλοήγηση

Γενικά

Ο τομέας της ρομποτικής πλοήγησης ασχολείται με το πρόβλημα του να καταφέρει ένα ρομπότ να κινηθεί με επιτυχία στον χώρο. Με επιτυχία εννοούμε ότι το ρομπότ κατ' αρχήν θα καταφέρει φτάσει στον προορισμό του, αλλά στην πορεία δεν θα έχει προσκρούσει πάνω σε εμπόδια και δεν θα έχει κάνει ζημιά στο περιβάλλον. Επίσης θέλουμε το ρομπότ να αποφύγει διάφορους κινδύνους που μπορεί να υπάρχουν στο περιβάλλον του και να διαλέξει την πιο σύντομη διαδρομή, λαμβάνοντας υπόψη του τα παραπάνω.

Για να μπορεί ένα ρομπότ να κινηθεί αποτελεσματικά στον χώρο πρέπει να έχει τρεις βασικές ικανότητες:

Αυτοτοποθέτηση
Σχεδιασμός πορείας
Κατασκευή ή και ερμηνεία ενός χάρτη του χώρου.

Αυτές οι τρεις ικανότητες είναι απαραίτητες για την επιτυχή πλοήγηση ενός ρομπότ στο περιβάλλον του, αν και οι ανάγκες διαφέρουν ανάλογα με την εφαρμογή. Παρακάτω θα τις αναλύσουμε περισσότερο.

Αξίζει να σημειωθεί ότι μιλώντας για ρομποτική πλοήγηση αναφερόμαστε σε ένα αυτόνομο και συνήθως κινούμενο ρομπότ. Το ρομπότ είναι αυτόνομο αφού πρέπει να πάρει μόνο του αποφάσεις για την πορεία του και φυσικά κινούμενο επειδή πρόκειται να κινηθεί, αν και υπάρχουν κάποιες εξαιρέσεις,

Πηγές πληροφορίας

Το ρομπότ έχει στη διάθεσή του δύο πηγές πληροφορίας με σκοπό να πλοηγηθεί στον χώρο. Πρόκειται για τις πληροφορίες που παίρνει σχετικά με τα δικά του εξαρτήματα, παραδείγματος χάρη το πόσα βήματα έχει κάνει ή πόσες φορές έχουν γυρίσει οι ρόδες του με σκοπό να βρει της απόσταση που έχει διανύσει.

Επιπλέον βάση των αισθητήρων που έχει παίρνει και πληροφορία για την μορφή του χώρου. Μπορεί να πρόκειται για ένα μεγάλο εύρος πληροφοριών από πολλούς αισθητήρες, όπως για παράδειγμα μπορεί να έχει οπτική πληροφορία από μία κάμερα, μία απεικόνιση του χώρου από ένα σονάρ και άλλα.

Καμία από τις δύο παραπάνω πηγές δεν αρκεί για να δώσει ακριβή πληροφορία σχετικά με το που βρίσκεται ανά πάσα στιγμή. Οι τεχνικές που χρησιμοποιούνται για να βρει την απόσταση που έχει διανύσει μπορεί να περιέχουν σφάλματα ενώ πχ μία εικόνα του χώρου δεν είναι αρκετή για να δώσει την ακριβή θέση του, καθώς

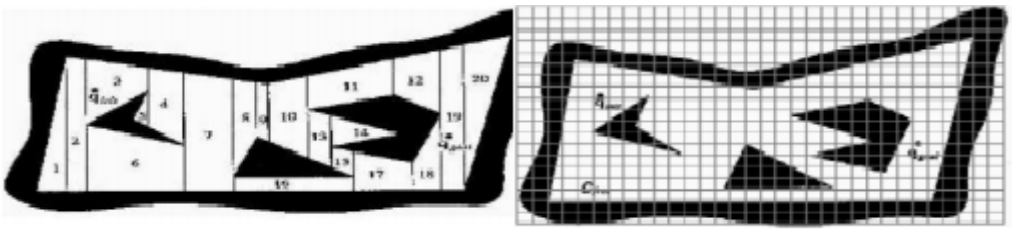
μπορεί δύο μέρη να είναι όμοια, όπως οι διάδρομοι σε ένα κτήριο γραφείων που μοιάζουν αρκετά μεταξύ τους.

Κατασκευή και ερμηνεία χάρτη του χώρου

Αυτό το πρόβλημα συνδέεται άμεσα με τον χώρο της χαρτογραφίας, δηλαδή της κατασκευής χαρτών. Ο σκοπός είναι να μπορεί ένα ρομπότ να κατασκευάσει έναν χάρτη του χώρου και να τον ερμηνεύσει. Προσοχή ότι η κατασκευή ενός χάρτη δεν είναι απαραίτητη και μπορεί να δίνεται κάποιος έτοιμος χάρτης, αλλά ακόμη και σε αυτή την περίπτωση το ρομπότ πρέπει να είναι σε θέση να τον ερμηνεύσει.

Η αναπαράσταση του χάρτη μπορεί να είναι είτε μετρική ή τοπολογική.

> Η *μετρική αναπαράσταση* είναι αυτή που χρησιμοποιούν οι άνθρωποι στους χάρτες. Ο χώρος θεωρείται ότι είναι μία επιφάνεια και πάνω σε αυτή τοποθετούνται τα αντικείμενα που περιέχει.
> Η *τοπολογική αναπαράσταση* λαμβάνει υπόψη της μόνο τα αντικείμενα και τη σχέση που έχουνε μεταξύ τους, συνήθως κρατώντας τις αποστάσεις ανάμεσά τους. Σε αυτή την περίπτωση ο χάρτης είναι ένας γράφος, όπου κάθε αντικείμενο είναι μία κορυφή και ενώνονται με ακμές που δίνουν τα μεταξύ τους μονοπάτια.



Εικόνα 1 Μία κάτοψη του περιβάλλοντος και η προσεγγιστική μετατροπή της σε κελιά

Ένας χάρτης δεν μπορεί να αναπαρασταθεί για ένα ρομπότ όπως για έναν άνθρωπο αλλά πρέπει να εκφραστεί με κάποιον μαθηματικό τρόπο. Υπάρχουν τρεις κύριοι μέθοδοι για την αναπαράσταση ενός χάρτη:

- *Χάρτες ελεύθερου χώρου*, οι οποίοι δεν αντιμετωπίζουν τον χώρο σαν μία επιφάνεια αλλά τα αντικείμενα μπορούν να είναι τοποθετημένα και σε διαφορετικό ύψος. Ανάλογα με τον τρόπο απεικόνησης μπορούν να χρησιμοποιούνται:
    - Γράφοι στον χώρο
    - Διαγράμματα Voronoi
    - Γενικευμένα διαγράμματα Voronoi
- *Χάρτες αντικειμένων*, που είναι οι τυπικοί χάρτες που χρησιμοποιούν οι άνθρωποι. Θεωρούν ότι ο χώρος είναι μία επιφάνεια χωρισμένη σε μικρά τετράγωνα και τοποθετούν σε αυτόν τα αντικείμενα που περιέχει.

- *Σύνθετοι χάρτες*, που πάλι θεωρούν ότι ο χώρος είναι μία επιφάνεια χωρισμένη σε τετράγωνα αλλά επιτρέπουν κάθε τετράγωνο να περιέχει λιγότερα ή περισσότερα τετράγωνα ανάλογα με τα αντικείμενα που υπάρχουν εκεί. Μπορεί να χρησιμοποιούνται:
    - Δίκτυα σημείων
    - Δίκτυα περιοχών
    - Τετραδικά δέντρα

Αυτοτοποθέτηση

Από τη στιγμή που το ρομπότ έχει έναν χάρτη του χώρου πρέπει να μπορεί να βρει που βρίσκεται σε αυτόν τον χώρο. Αυτό δεν είναι τόσο απλό όσο φαίνεται από την στιγμή που το ρομπότ αρχίσει να κινείται, καθώς σίγουρα δεν μπορεί να βρει πόση απόσταση έχει διανύσει και προς πια κατεύθυνση χωρίς σφάλματα. Σε αυτή την περίπτωση μπορεί να προσανατολιστεί εξετάζοντας πάλι τον χώρο.

Σε αυτόν τον τομέα χρησιμοποιείται η θεωρεία πιθανοτήτων και συγκεκριμένα φίλτρα Bayes ώστε να υπολογιστεί που είναι πιθανόν να βρίσκεται κάθε φορά το ρομπότ μετά από μία κίνηση. Η βασική ιδέα είναι ότι δεδομένων των σφαλμάτων στον υπολογισμό της θέσης του ρομπότ κάθε κίνηση κάνει λιγότερο πιθανό να ξέρει το ρομπότ την ακριβή του θέση αλλά χρησιμοποιώντας πιθανότητες μπορεί να υπολογιστεί πόσο περίπου απέχει από την υπολογιζόμενη θέση. Φυσικά αυτή η απόκλιση αυξάνεται συνεχώς, οπότε όταν ξεπεράσει ένα συγκεκριμένο όριο τότε πρέπει πάλι να προσπαθήσει να προσανατολιστεί.

Σχεδιασμός πορείας

Εφόσον ένα ρομπότ έχει μία εικόνα του περιβάλλοντός του και ξέρει που βρίσκεται πρέπει να βρει πια είναι η κατάλληλη διαδρομή που πρέπει να πάρει για να πάει από ένα σημείο σε ένα άλλο. Κάτι τέτοιο δεν είναι αυτονόητο για τις μηχανές όπως είναι για τους ανθρώπους. Ο συντομότερος δρόμος ανάμεσα σε δύο σημεία είναι η ευθεία, όμως αυτό δεν είναι πάντα η λύση στο πρόβλημά μας. Αρχικά ένας χώρος μπορεί να περιέχει μία πλειάδα εμποδίων ανάμεσα στα οποία πρέπει να κινηθεί το ρομπότ για να φτάσει στον προορισμό του. Επιπλέον μπορεί να υπάρχουν και επιμέρους παράμετροι που το ρομπότ πρέπει να λάβει υπόψη του, όπως το ίδιο του το μέγεθος, δηλαδή μπορεί να μην χωράει να περάσει ανάμεσα από δύο εμπόδια ή να μην μπορεί να στρίψει αρκετά σε ένα συγκεκριμένο σημείο.

Πέρα από αυτά μπορεί να υπάρχουν περιπτώσεις που να μην ισχύουν απόλυτες απαγορεύσεις, όπως το μέγεθος, αλλά σχετικές. Κατ' αρχήν πάντα θέλουμε το ρομπότ να ακολουθεί την συντομότερη διαδρομή και να μην κάνει άσκοπους κύκλους. Επίσης μπορεί να έχουν δοθεί οδηγίες στο ρομπότ όπως «προσπάθησε να αποφύγεις τα πολύ ζεστά μέρη» αλλά και ταυτόχρονα «μην καθυστερήσεις πάνω από ένα ορισμένο χρονικό διάστημα» (σε ένα ρομπότ διάσωσης για παράδειγμα).

Σε αυτή την περίπτωση το ρομπότ πρέπει να περάσει από πολύ ζεστά μέρη μόνο όσο χρειάζεται ώστε να φτάσει στον προορισμό του στον επιθυμητό χρόνο.

Το πρόβλημα του να βρεθεί μία διαδρομή που να ικανοποιεί κάποιες προϋποθέσεις είναι ένα κλασσικό πρόβλημα των μαθηματικών, το πρόβλημα της ελάχιστης διαδρομής. Μία τυπική λύση είναι η μετατροπή των πιθανών σημείων της διαδρομής σε κορυφές ενός γράφου, ενώ ενώνονται κατάλληλα με ακμές οι οποίες δείχνουν πια από αυτά επικοινωνούν. Αν θέλουμε να εισάγουμε επιπλέον περιορισμούς τότε αυξάνουμε ή μειώνουμε ανάλογα τα βάρη κάθε μονοπατιού ώστε να αντικατοπτρίζει το πόσο επιθυμητή ή όχι είναι η συγκεκριμένη διαδρομή, ενώ αν έχουμε και έναν χρονικό περιορισμό μπορούμε να βάλουμε και ένα μέγιστο μέγεθος για το τελικό μονοπάτι. Έπειτα χρησιμοποιούμε έναν αλγόριθμο ώστε να βρούμε την ελάχιστη διαδρομή που να ικανοποιεί αυτές τις προϋποθέσεις. Υπάρχουν αρκετοί αλγόριθμοι που μπορούν να υπολογίσουν το ελάχιστο μονοπάτι, με τον καθένα να έχει διαφορετικές απαιτήσεις στο χρόνο που χρειάζεται για να ολοκληρωθεί και στο πόσο καλά είναι τα αποτελέσματα που βγάζει.

Κάποιοι από τους τυπικούς αλγόριθμους εύρεσης ελάχιστου μονοπατιού είναι:

Ο αλγόριθμος Dijkstra.
Ο αλγόριθμος Bellman-Ford.
Ο αλγόριθμος αναζήτησης A^*.
Ο αλγόριθμος Floyd-Marshall.
Ο αλγόριθμος Johnson.
Η θεωρεία κλονισμού τροχιάς.

Αλγόριθμοι αποφυγής εμποδίων

Αφού βρεθεί το μονοπάτι το πρόβλημα που μένει είναι πως το κινούμενο ρομπότ θα αποφύγει εμπόδια, χωρίς να προσκρούσει πάνω τους. Υπάρχουν διάφοροι αλγόριθμοι που το κάνουν αυτό.

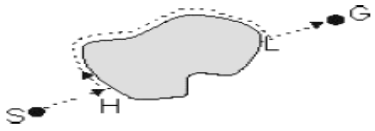
Αλγόριθμοι Bug

Μια λύση, η οποία μάλιστα είναι από τις πρώτες που προτάθηκαν, είναι γνωστή ως «bug algorithm», καθώς μιμείται τον τρόπο που τα έντομα βρίσκουν το δρόμο τους.

Εικόνα 2 Η διαδρομή που ακολουθείται από το ρομπότ βάση του αλγορίθμου Bug

Σε αυτού του τύπου τους αλγορίθμους το κινούμενο ρομπότ έχει προσχεδιασμένο μόνο τον προορισμό αλλά όχι κάποιο συγκεκριμένο μονοπάτι. Κάθε φορά που κάποιο εμπόδιο εισέρχεται στο βελησνεκές των αισθητήρων του, το ρομπότ κινείται περιμετρικά από το εμπόδιο μέχρι να επανέλθει στην αρχική θέση. Τότε υπολογίζει το σημείο του εμποδίου με την μικρότερη διαδρομή μέχρι τον προορισμό του και στη συνέχεια αφήνει την περίμετρο του εμποδίου από αυτό το σημείο. Προφανώς αυτός ο αλγόριθμος δεν είναι αποδοτικός, όμως είναι πολύ απλός στον υπολογισμό.

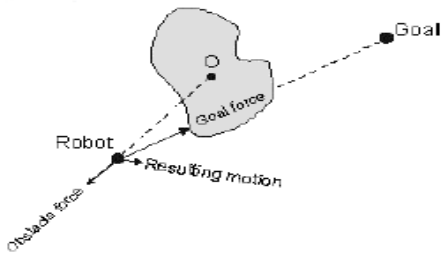
Λόγω της μη αποδοτικότητάς του έχουν προταθεί διάφορες βελτιώσεις, όπως ο αλγόριθμος “bug 2”, όπου το ρομπότ και πάλι αρχίζει να κινείται στην περίμετρο του εμποδίου, όμως όταν φτάσει στη νοητή γραμμή που ενώνει την αρχική θέση με τον προορισμό, εγκαταλείπει την περίμετρο του εμποδίου και συνεχίζει την πορεία του. Αν και αυτή η λύση είναι πιο γρήγορη και δεν απαιτεί από το ρομπότ να διανύσει ολόκληρο τον κύκλο του εμποδίου, έχει όλα τα υπόλοιπα μειονεκτήματα του αρχικού bug αλγόριθμου και είναι και αυτός πολύ αργός σε σχέση με άλλους αλγορίθμους.



Εικόνα 3 Η διαδρομή που ακολουθείται από το ρομπότ βάση του αλγορίθμου Bug 2

Potential Field

Αντίθετα με τον αλγόριθμο Bug και τις παραλλαγές του, σε αυτή την πρόταση το πρόβλημα της αποφυγής εμποδίων είναι ένα υποπρόβλημα του γενικότερου προβλήματος της εύρεσης μονοπατιού από το ένα σημείο στο άλλο.



Εικόνα 4 Η διαδρομή που ακολουθείται από το ρομπότ βάση του αλγορίθμου Potential Field

Στο potential field ο αλγόριθμος υποθέτει πως υπάρχουν εικονικές δυνάμεις που είτε έλκουν το κινούμενο ρομπότ προς τον προορισμό του ή το απωθούν από τα εμπόδια. Το τελικό μονοπάτι είναι το αποτέλεσμα του συγκερασμού αυτών των δυνάμεων. Παρόλο όμως που είναι μια σαφής βελτίωση σε σχέση με τους bug

αλγόριθμους, δεν είναι σε θέση να προσπεράσει όλα τα εμπόδια ή να χωρέσει σε σχετικά στενά περάσματα, έτσι η πρακτική του αξία μειώνεται.

Elastic bands και bubbles

Μια άλλη, σαφώς βελτιωμένη λύση, είναι ο αλγόριθμος elastic bands που βασίζεται με τη σειρά του στη λύση των Bubbles. Σε αυτή τη μέθοδο, ορίζεται ένα "bubble" γύρω από το ρομπότ, με σχήμα παρόμοιο, αλλά απλούστερο, με αυτό του ρομπότ. Στο κέντρο του bubble είναι το ρομπότ. Το bubble αυτό καταλαμβάνει το μέγιστο δυνατό χώρο γύρω από το ρομπότ και τα εμπόδια, ώστε να μπορεί να κινηθεί προς οποιαδήποτε κατεύθυνση χωρίς να χτυπήσει κάπου. Το μέγεθος αυτού του χώρου οριοθετείται από τους αισθητήρες και εξαρτάται και από το βελησνεκές τους. Στην απλούστερη περίπτωση το bubble έχει σχήμα σφαίρας, ώστε να είναι λιγότερο πολύπλοκο στον υπολογισμό των αποστάσεων, με κέντρο το ρομπότ και διάμετρο τη μικρότερη απόσταση από ένα πιθανό εμπόδιο.

Φτιάχνοντας μια αλληλουχία από τέτοιες περιοχές, δημιουργείται ένα «band» από bubbles, το οποίο χρησιμοποιείται για τη δημιουργία ενός μονοπατιού ανάμεσα στο αρχικό σημείο της κίνησης, μέχρι τον προορισμό.

Κάθε φορά που κινείται το ρομπότ, νέα δεδομένα από τους αισθητήρες του λαμβάνονται υπόψη κι έτσι, συνυπολογίζοντας και τα μέχρι τότε στοιχεία από το υπάρχον μονοπάτι, μπορεί να αποφύγει ακόμα και κινούμενα εμπόδια, αυξομειώνοντας το μέγεθος των bubbles. Με αυτό τον τρόπο το band των bubbles γίνεται ελαστικό, από όπου αντλεί το όνομά του ο αλγόριθμος.

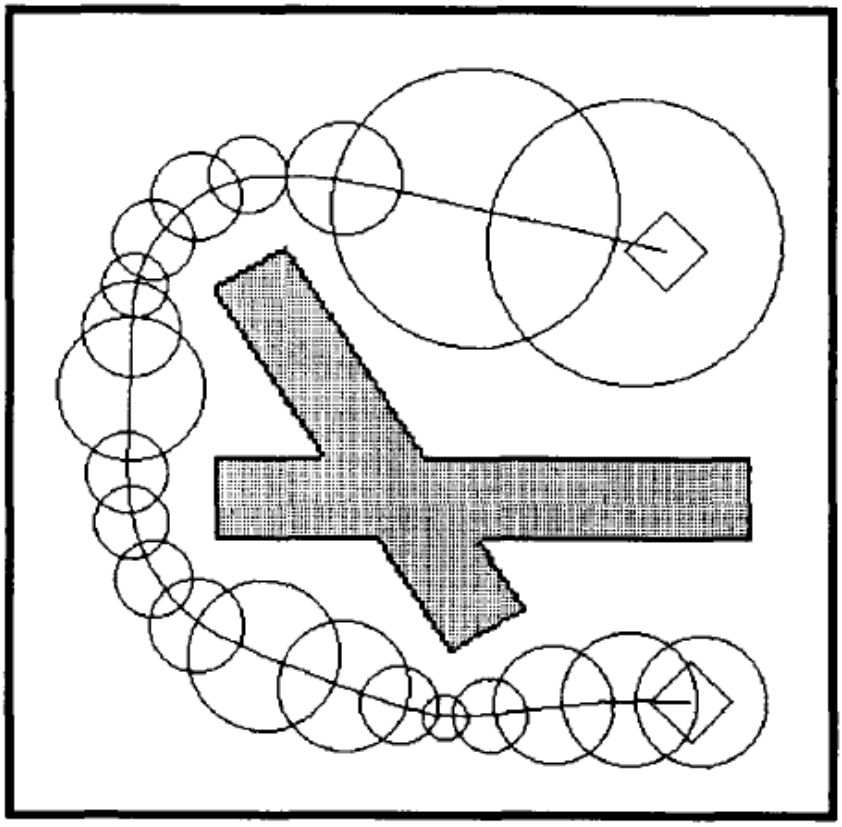
Οι γενικότεροι μαθηματικοί κανόνες που πρέπει να ισχύουν για να λειτουργήσει ο αλγόριθμος είναι οι εξής: σε έναν χώρο M υπάρχουν κάποια εμπόδια O. Ο χώρος που καταλαμβάνεται από τα εμπόδια είναι ένα σύνολο που αποτελεί υποσύνολο του χώρου, δηλαδή $M \supseteq O$. Για το σύνολο A του χώρου το σύνολο A αντιπροσωπεύει τον ελεύθερο χώρο και το B ένα στοιχείο του A, το οποίο είναι οποιαδήποτε θέση θα μπορούσε να είναι ένα Bubble, και P το κέντρο του κάθε Bubble. Για αυτά τα σύνολα θα πρέπει να ισχύει:

- $A \supset B$ (δηλαδή το σύνολο B είναι υποσύνολο του A)
- $P \in B$ (δηλαδή τα σημεία του P ανήκουν στο B)
- $\forall Q \in B \; \exists$ *ένα βέλτιστο μονοπάτι* $\neg (P,Q) \subset$ *Bσε συνδιασμό με το* P *και το* Q
  (ότι δηλαδή το κάθε σημείο του μονοπατιού είναι προσβάσιμο από το ρομπότ)

Για το μονοπάτι θα πρέπει να ισχύει το εξής:

Υπάρχει ένα αρχικό σημείο $Po \in M$, ένα τελικό σημείο $P1 \in M$, και ένα πεπερασμένο σύνολο από bubbles $\{B_i, i=1, \ldots, n\}$ τέτοιο ώστε:

- Όλα τα bubbles ανήκουν στον ελεύθερο χώρο M-O.
- $Bi \cap Bi+1 \neq 0 \; \forall \; i = 0, \ldots, n-1$ που σημαίνει ότι το κάθε bubble έχει κοινά στοιχεία με το αμέσως προηγούμενο και το αμέσως επόμενο.
- Και τέλος ότι P_0 είναι το κέντρο του B_1 και P_i είναι το κέντρο του B_n.

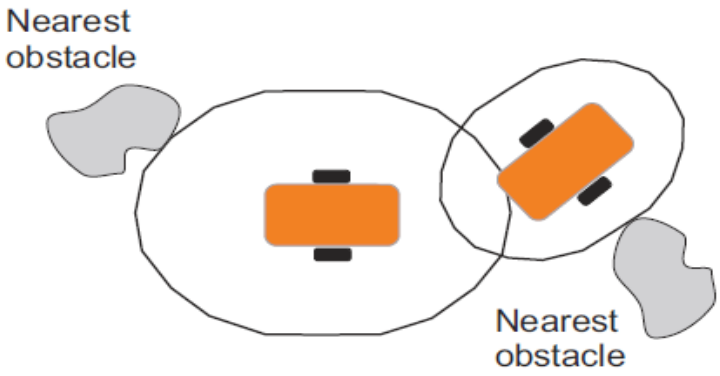


Εικόνα 5 Το μονοπάτι και τα bubbles που δημιουργούνται κατά την εκτέλεση του αλγορίθμου elastic bands

Ανάλογα με το σχήμα του ρομπότ αλλάζει και το σχήμα των Bubbles. Ένα απλό σχήμα, όπως ο κύκλος, είναι πιο εύκολο να υπολογιστούν και χρειάζονται μικρότερη υπολογιστική ισχύ από πιο πολύπλοκα σχήματα. Όμως το σχήμα αυτό εξαρτάται από το σχήμα του ρομπότ και όσο πιο κοντά είναι στο κανονικό σχήμα, τόσο πιο ακριβές γίνεται το μονοπάτι. Έτσι χρειάζεται να γίνει κάποια παραχώρηση και να έχουμε όσο το δυνατόν πιο απλοποιημένο σχήμα.

Όσο πιο μικρά είναι τα bubbles σε κάποιο σημείο, τόσο πιο ακριβές είναι το μονοπάτι. Επίσης η κάθε περιοχή θα πρέπει να υπερκαλύπτει μέρος της επόμενης.

Σε κάθε νέο υπολογισμό του μονοπατιού λοιπόν αφαιρούνται τα πλεονάζοντα bubbles ή αυτά που δεν ισχύουν πλέον, αλλά και ενδεχομένως να δημιουργούνται νέα. Τα bubbles επιλέγονται βάση μιας εσωτερικής ώθησης το μονοπάτι να είναι όσο πιο συμπαγές γίνεται και ταυτόχρονα μιας εξωτερικής ώθησης να αποφύγει τα εμπόδια.



Εικόνα 6 Δημιουργία bubbles κατά την κίνηση του ρομπότ

Για να καταλήξει στην επιλογή των σωστών κάθε φορά bubbles ο αλγόριθμος μετακινεί κάθε περιοχή προς τον προορισμό. Πολύ για τις μεγάλες περιοχές, λίγο για τις μικρές. Αν η νέα περιοχή δεν συναντά κάποιο εμπόδιο, αλλά ταυτόχρονα παραμένει πάνω στο μονοπάτι, δηλαδή επικαλύπτει το χώρο των διπλανών του bubbles, τότε η νέα περιοχή προστίθεται στο μονοπάτι και παίρνει τη θέση της μετακινημένης. Σε περίπτωση που δεν επικαλύπτεται από άλλες υπάρχουσες θέσεις, τότε δημιουργείται μια νέα περιοχή. Αν ούτε αυτή ανήκει στο μονοπάτι (δεν επικαλύπτεται από τις γειτονικές) τότε η νέα θέση θεωρείται άκυρη και το bubble παραμένει στην αρχική του θέση και μέγεθος. Κάνοντας αυτή τη διαδικασία για κάθε νέα θέση το ρομπότ είναι σε θέση να αποφύγει ακόμα και εμπόδια που δεν είχαν υπολογιστεί στο αρχικό μονοπάτι.

Παραλλαγές αυτού του αλγορίθμου χρησιμοποιούν «αστέρια» αντί για σφαίρες ως το βασικό τους σχήμα. Τα αστέρια αυτά έχουν τις ίδιες ιδιότητες με τις σφαίρες που είδαμε παραπάνω και αποτελούν τα bubbles. Η διαφορά βρίσκεται στην αποτελεσματικότητα και στην πολυπλοκότητα. Από πειραματικά δεδομένα αποδεικνύεται ότι σε ακραίες συνθήκες οι δύο λύσεις έχουν την ίδια πολυπλοκότητα και παράγουν τα ίδια αποτελέσματα, ενώ στις περισσότερες περιπτώσεις τα αστέρια είναι λιγότερο πολύπλοκα και κατά συνέπεια καλύτερα.

Η μέθοδος των elastic bands, με όλες τις παραλλαγές, εξαρτάται βέβαια από τους αισθητήρες, οι οποίοι όσο πιο ακριβείς είναι τόσο πιο ακριβές είναι και το μονοπάτι. Σε πειραματικό επίπεδο έχουν δοκιμαστεί ψηφιακές κάμερες σε συνδυασμό με τρισδιάστατους Laser σαρωτές.

Simultaneous localization and mapping

Γενικά

Όπως αναφέρθηκε παραπάνω τα ρομπότ χρησιμοποιούν έναν χάρτη για να πλοηγηθούν στο περιβάλλον. Αυτός ο χάρτης μπορεί να δίνεται έτοιμος στο ρομπότ ή μπορεί το ίδιο να προσπαθήσει να τον κατασκευάσει παρατηρώντας το περιβάλλον του χρησιμοποιώντας τους αισθητήρες με τους οποίους είναι εξοπλισμένο και χρησιμοποιείται για να αναπαραστήσει το περιβάλλον ώστε το ρομπότ να μπορέσει να σχεδιάσει την πορεία του και γενικά όσο πιο λεπτομερής είναι ο χάρτης τόσο πιο μεγάλη βοήθεια προσφέρει στις αποφάσεις που παίρνει το ρομπότ.

Τεχνικά προβλήματα

Το μεγάλο πρόβλημα είναι η ακρίβεια των αισθητήρων του ρομπότ, καθώς ακόμη και αν έχει ήδη έτοιμο τον χάρτη δεν μπορεί να ξέρει ακριβώς που βρίσκεται. Ένας λόγος είναι ο μηχανισμός με τον οποίο το ρομπότ μετράει την απόσταση που έχει διανύσει από το αρχικό του σημείο, για παράδειγμα το πόσες φορές έχουν γυρίσει οι ρόδες του αν το ρομπότ είναι τροχοφόρο ή το πόσα βήματα έχει κάνει αν πρόκειται για ένα βαδίζον ρομπότ. Οι μετρήσεις που επιστρέφουν αυτές περιέχουν ένα μικρό ή μεγάλο ποσοστό λάθους. Όμως σε αυτή την περίπτωση όσο περισσότερο απομακρύνεται το ρομπότ από την αρχική του θέση τόσο πιο πολύ μεγαλώνει το απόλυτο λάθος. Μπορεί αρχικά το λάθος να είναι πάρα πολύ μικρό και να είναι ανεκτό, αλλά όσο κινείται το ρομπότ τόσο περισσότερο μεγαλώνει το λάθος και μοιραία από κάποιο σημείο και μετά θα γίνει τόσο μεγάλο που θα κάνει την σωστή πλοήγηση στον χώρο αδύνατη.

Το πρόβλημα τονίζεται ακόμη περισσότερο στην περίπτωση που το ρομπότ χρειάζεται να κατασκευάσει μόνο του έναν χάρτη του χώρου στον οποίο πρέπει να κινηθεί. Σε αυτή την περίπτωση ο ίδιος ο χάρτης έχει πολύ μεγαλύτερη πιθανότητα να είναι ελλιπής ή λαθεμένος.

Κατ’ αρχήν δεν είναι πάντα δυνατόν για το ρομπότ να έχει μία πλήρη εικόνα του χώρου από την αρχική του θέση. Μπορεί ένα μεγάλο εμπόδιο να κρύβει το τι βρίσκεται πίσω από αυτό. Ακόμη χειρότερα μπορεί ο τελικός προορισμός στον οποίο πρέπει να πάει το ρομπότ να μην είναι ορατός και να πρέπει να κινηθεί στον χώρο για να τον ανακαλύψει εισάγοντας τα λάθη μέτρησης που αναφέραμε προηγουμένως για τον εντοπισμό του που βρίσκεται στον χώρο. Ακόμη χειρότερα ο χώρος στον οποίο κινείται μπορεί να μεταβάλλεται με τέτοιο τρόπο ώστε κάποια εμπόδια να μετακινούνται και να ανοίγουν κάποιες διαδρομές κλείνοντας κάποιες άλλες.

Επίσης αν πρέπει να κατασκευάσει έναν χάρτη βασιζόμενο μόνο στις δικές του αισθήσεις χωρίς να έχει δοθεί μία έτοιμη άποψη του χώρου πρέπει να χρησιμοποιήσει αποκλειστικά τα αισθητήρια του. Και σε αυτή την περίπτωση, άσχετα με το πόσο ακριβή είναι τα αισθητήρια του εισάγεται αυτόματα μέσα στον χάρτη ένα ποσοστό λάθους. Επίσης στον πραγματικό κόσμο είναι αδύνατον να γίνει μία μέτρηση χωρίς να υπάρχει ένα μικρότερο ή μεγαλύτερο επίπεδο θορύβου που μειώνει την ποιότητά της.

Η λύση στα παραπάνω προβλήματα δεν είναι προφανής, και για την ακρίβεια οι προφανείς λύσεις αποτυγχάνουν.

Μπορούμε να θέσουμε διάφορους περιορισμούς στον χώρο στον οποίο χρησιμοποιείται το ρομπότ, ώστε να εγγυηθούμε ότι το περιβάλλον δεν θα μεταβάλλεται (δηλαδή τα εμπόδια θα είναι σταθερά) και το επίπεδο του θορύβου θα είναι σίγουρα ανεκτό. Αν και ένα τέτοιο ιδανικό περιβάλλον μπορεί με κάποια προσπάθεια να δημιουργηθεί στο εργαστήριο, είναι τεχνιτό και ένα ρομπότ που είναι σχεδιασμένο με αυτό τον τρόπο δεν θα είναι ιδιαίτερα χρήσιμο στον πραγματικό κόσμο.

Ένας άλλος τρόπος να περιορίσουμε το πρόβλημα είναι να δώσουμε από την αρχή έναν αρκετά λεπτομερή χάρτη του κόσμου στο ρομπότ ώστε να μην έχουμε το πρόβλημα των λαθών που αυτό μπορεί να κάνει όταν σχεδιάζει τον δικό του χάρτη, αλλά και αυτή η λύση δεν έχει ιδιαίτερο νόημα γιατί όπως και πριν δεν επιθυμούμε πάντα την λειτουργία του ρομπότ μέσα σε έναν τεχνητό κόσμο αλλά πολλές φορές θέλουμε να μπορεί να λειτουργήσει αυτόνομα σε ένα νέο και άγνωστο για αυτό περιβάλλον.

Τέλος θα μπορούσαμε να χρησιμοποιήσουμε αρκετά ποιοτικούς αισθητήρες ώστε το ποσοστό λάθους που εισάγεται από αυτούς να είναι ελάχιστο και να μην έχει πραγματική επίπτωση στην λειτουργία του ρομπότ, ή ακόμη και να του δίνουμε κάθε στιγμή την ακριβή του θέση στέλνοντας ένα εσωτερικό σήμα με κάποιο τρόπο. Όμως σε αυτή την περίπτωση το ρομπότ δεν θα ήταν πραγματικά αυτόνομο.

Η χρήση αισθητήρων μεγάλης ακρίβειας πιθανότητα δεν είναι πάντα εφικτή γιατί πιθανότατα θα ανέβαζε το κόστος του ρομπότ σε πολύ μεγάλα επίπεδα ώστε η χρήση του να είναι απαγορευτική. Ακόμη και στην περίπτωση που το κόστος είναι ανεκτό υπάρχουν περιπτώσεις όπου η τεχνολογία απλά δεν μπορεί να παράγει καλύτερους αισθητήρες ώστε να χρησιμοποιηθούν. Η θωράκιση των αισθητήρων ώστε να μειώσουν το επίπεδο του θορύβου σε ελάχιστο επίπεδο και πάλι δεν είναι πάντα δυνατή για τον ίδιο λόγο.

Ταυτόχρονη χαρτογράφηση και τοποθέτηση στον χώρο

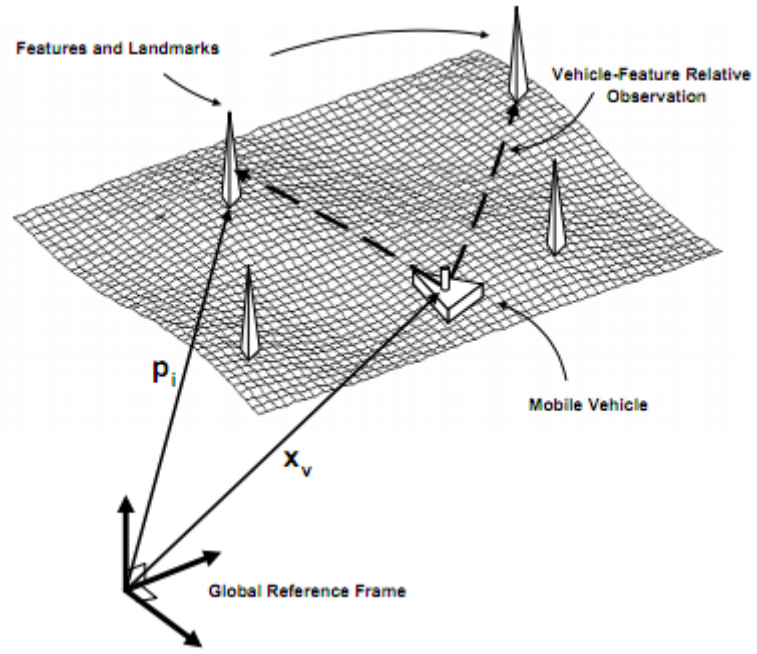
Η λύση στα παραπάνω προβλήματα είναι η τεχνική ταυτόχρονης χαρτογράφησης και τοποθέτησης στον χώρο ή στα αγγλικά Simultaneous localization and mapping

(SLAM). Δεδομένου ότι το ρομπότ έχει πρόβλημα να δημιουργήσει έναν ακριβή χάρτη και να κινηθεί με ακρίβεια μέσα σε αυτόν ξέροντας κάθε στιγμή την θέση του η λύση είναι η συνεχής επανεξέταση και των δύο ερωτημάτων.

Πολλές φορές στην επιστημονική κοινότητα το SLAM αναφέρεται σαν το πρόβλημα της κότας με το αυγό. Για να μπορέσει να τοποθετήσει τον εαυτό του στον χώρο χρειάζεται να έχει έναν έγκυρο χάρτη. Αλλά για να κατασκευάσει ένα έγκυρο χάρτη πρέπει να ξέρει που ακριβώς βρίσκεται.

Αρχικά το ρομπότ χρησιμοποιώντας τα αισθητήριά του χαρτογραφεί το περιβάλλον του. Το ότι σε αυτόν τον χάρτη θα υπάρχουν σφάλματα είναι γνωστό αλλά θεωρούμε πώς χωρίς ιδιαίτερη προσπάθεια ο χάρτης θα είναι επαρκώς ακριβής ώστε να μπορέσει να διανύσει χωρίς ιδιαίτερο πρόβλημα μικρές αποστάσεις μέσα στον χώρο. Έπειτα επιλέγει μία πορεία και κινείται προς αυτή. Εξαιτίας της κίνησής του στον χώρο η εικόνα του περιβάλλοντος που δίνεται από τα αισθητήριά του αλλάζει, με αποτέλεσμα να έχει μία διαφορετική εικόνα του περιβάλλοντος και να μπορεί να χαρτογραφήσει ακόμη μεγαλύτερο μέρος του χώρου.

Φυσικά για να το κάνει αυτό πρέπει να ξέρει πόσο κινήθηκε. Αν αρχικά είχε συναντήσει δύο εμπόδια και προχωρώντας συναντήσει ένα τρίτο θα προσπαθήσει να το τοποθετήσει στον χάρτη. Σε αυτή την περίπτωση η θέση που θα το βάλει σε σχέση με τα προηγούμενα δύο εμπόδια εξαρτάται μόνο από την κατεύθυνση στην οποία έχει κινηθεί και την διεύθυνση της κίνησης.



Εικόνα 7 Ένα ρομπότ που κάνει σχετικές μετρήσεις για την απόσταση των αντικειμένων από αυτό.

Μία μέτρηση που αρχικά είναι λάθος ίσως να μην δημιουργήσει ιδιαίτερο πρόβλημα αλλά όπως αναφέραμε παραπάνω οι ανακριβείς μετρήσεις με τον χρόνο συσσωρεύονται και καταλήγουν να κάνουν την ακριβή πλοήγηση αδύνατη. Για αυτό τον λόγο υπάρχουν διάφορες τεχνικές που χρησιμοποιούνται ώστε να αποτελέσματα να διορθώνονται.

Θεωρητική εκτίμηση

Η πιο δημοφιλής προσέγγιση για την μείωση των σφαλμάτων μέτρησης βασίζεται στην θεωρεία πιθανοτήτων. Η κύρια λογική είναι πως μπορεί τα σφάλματα σε κάθε μέτρηση να μην είναι μετρήσιμα, αλλά το όριο σφάλματος είναι γνωστό εκ των προτέρων. Κατά συνέπεια αν υπολογιστούν κατάλληλα οι πιθανές αποκλίσεις κάθε μέτρησης τότε μπορεί με μία σχετική βεβαιότητα να εξαχθεί η πιθανότερη σωστή τιμή.

Κάτι τέτοιο είναι αρκετά απαιτητικό σε υπολογιστικές απαιτήσεις. Ο λόγος είναι πως ανά πάσα στιγμή το ρομπότ κάνει εκατοντάδες μετρήσεις με τους αισθητήρες του προκειμένου να χαρτογραφήσει τον χώρο και τα πιθανά σφάλματα πρέπει να υπολογιστούν για κάθε μία από αυτές. Ακόμη η κίνηση του ρομπότ, δηλαδή το πόσο θέλησε να κινηθεί και το πόσο τελικά κινήθηκε προσθέτουν ακόμη περισσότερες μετρήσεις το ποσοστό λάθους των οποίων πρέπει και αυτό να υπολογιστεί. Καθώς τα περισσότερα ρομπότ είναι μικρά σε μέγεθος δεν είναι πάντα εφικτό και εύκολο να έχουν ενσωματωμένους τόσο δυνατούς επεξεργαστές ώστε να κάνουν τις απαραίτητες πράξεις και κατά συνέπεια είναι αρκετά αργά στις κινήσεις και τις αντιδράσεις τους.

Οι σημαντικότεροι αλγόριθμοι που χρησιμοποιούνται από αυτή την προσέγγιση είναι:

- *Φίλτρα Kalman*, μία μαθηματική μέθοδος που αντιμετωπίσει ακριβώς το πρόβλημα της εξουδετέρωσης σφαλμάτων από τα πειραματικά δεδομένα.
- *Φίλτρα σωματιδίων*, ένα μοντέλο τεχνικών εκτίμησης βασισμένα στην εξομοίωση συστημάτων.
- *Μέθοδοι Monte Carlo*, μία ομάδα υπολογιστικών αλγορίθμων τα οποία βασίζονται στην εκτίμηση τυχαίων δειγμάτων ώστε να υπολογίσουν την κατάσταση ενός συστήματος.

Σχετική θέση χαρακτηριστικών σημείων

Μια δεύτερη προσέγγιση περιορίζει κατά πολύ την θεωρητική προσέγγιση και βασίζεται σε μία απλούστερη στατιστική προσέγγιση των μετρήσεων.

Αυτός ο τρόπος υπολογισμού έχει το πλεονέκτημα ότι δεν χρειάζεται πολύπλοκους υπολογισμούς όπως η θεωρητική προσέγγιση αλλά ο προσδιορισμός της απόστασης ανάμεσα σε δύο (ή και περισσότερα) σημεία είναι υπολογιστικά απλή,

επιτρέποντας και την απλοποίηση των υπολογιστικών συστημάτων στα ρομπότ επιτυγχάνοντας την μείωση του κόστους τους.

Κάποιες συγκεκριμένες εφαρμογές αυτής της προσέγγισης είναι:

- *Διαχωρισμός των αισθητηρίων του ρομπότ* σε διαφορετικές αυτόνομες ομάδες, κάθε μία εκ των οποίων κάνει τους υπολογισμούς της ανεξάρτητα από την άλλη. Ένα κεντρικό υπολογιστικό σύστημα είναι υπεύθυνο για την συγκέντρωση των διαφορετικών μετρήσεων και την εξαγωγή των τελικών μετρήσεων.
- *Πολλαπλές μετρήσεις κατά την πορεία.* Η διαφορά αυτού του αλγόριθμου από τον προηγούμενο είναι ότι το ρομπότ δεν είναι απαραίτητο να περιέχει πολλές διαφορετικές ομάδες αισθητηρίων αλλά αντίθετα κάνει συνέχεια μετρήσεις του περιβάλλοντος ενώ προχωράει.
- *Αναγνώριση και εξαντλητική μέτρηση των σχετικών σημείων.* Με αυτόν τον τρόπο το ρομπότ προσπαθεί να διακρίνει ένα χαρακτηριστικό σημείο του περιβάλλοντος και έπειτα να το χαρτογραφήσει εξαντλητικά από αρκετές διαφορετικές γωνίες ώστε με το πλήθος των μετρήσεων να βγάλει ένα ακριβές μοντέλο. Έπειτα διαλέγει ένα άλλο χαρακτηριστικό σημείο του περιβάλλοντος και επαναλαμβάνει αυτή την διαδικασία. Με τις συνεχείς μετρήσεις τα πιθανά σφάλματα εξαφανίζονται. Με αυτόν τον τρόπο μπορούν να δημιουργηθούν ακριβείς χάρτες του περιβάλλοντος αλλά είναι αρκετά χρονοβόρα.

Υπολογιστική προσέγγιση

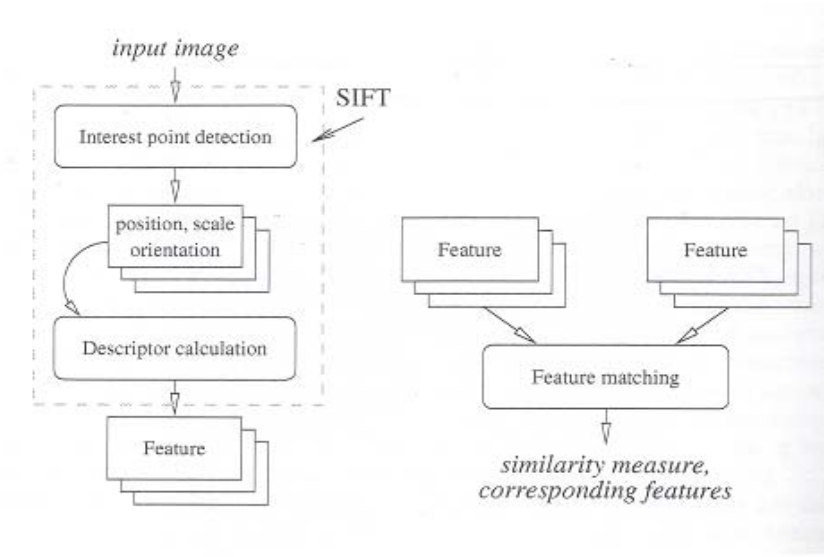
Μία Τρίτη σχολή σκέψης είναι τελείως διαφορετική από την θεωρητική εκτίμηση και βασίζεται στον υπολογισμό των θέσεων των αντικειμένων βάση αρκετών μετρήσεων των μεταξύ τους θέσεων. Με αυτόν τον τρόπο δεν υπολογίζεται καθόλου το που περίπου πρέπει να είναι ένα αντικείμενο αλλά το αποτέλεσμα βασίζεται μόνο σε πολλές μετρήσεις των ιδίων αποστάσεων από διαφορετική άποψη.

Αν και αυτός ο τρόπος είναι αρκετά απλός στην υλοποίηση του χρειάζεται πάλι την επεξεργασία αρκετών ενδείξεων των αισθητηρίων του ρομπότ, με αποτέλεσμα όπως και η θεωρητική εκτίμηση έχει αρκετά μεγάλες απαιτήσεις από τον επεξεργαστή του ρομπότ.

Κάποια χαρακτηριστικά παραδείγματα πάνω στην έρευνα σε αυτόν τον τομέα είναι τα παρακάτω:

- *Ταύτιση χαρακτηριστικών σημείων*: Το ρομπότ αναγνωρίζει κάποια χαρακτηριστικά σημεία του περιβάλλοντος και σε όλες τις μετρήσεις υπολογίζει τις αποστάσεις όλων των αντικειμένων από αυτά.

- *Εύρεση ορίων περιοχών*: Το ρομπότ προσπαθεί να βρει κάποιες χαρακτηριστικές περιοχές όπως για παράδειγμα οροσειρές και δεν κάνει μετρήσεις για να βρει που ακριβώς βρίσκονται αυτές αλλά για να υπολογίσει κάποια χαρακτηριστικά όρια σε αυτές ώστε να τις αποφύγει. Προφανώς αυτός ο αλγόριθμος έχει εφαρμογές σε ρομπότ που λειτουργούν σε πολύ μεγάλες περιοχές όπως για παράδειγμα σε ρομποτικά αεροσκάφη.

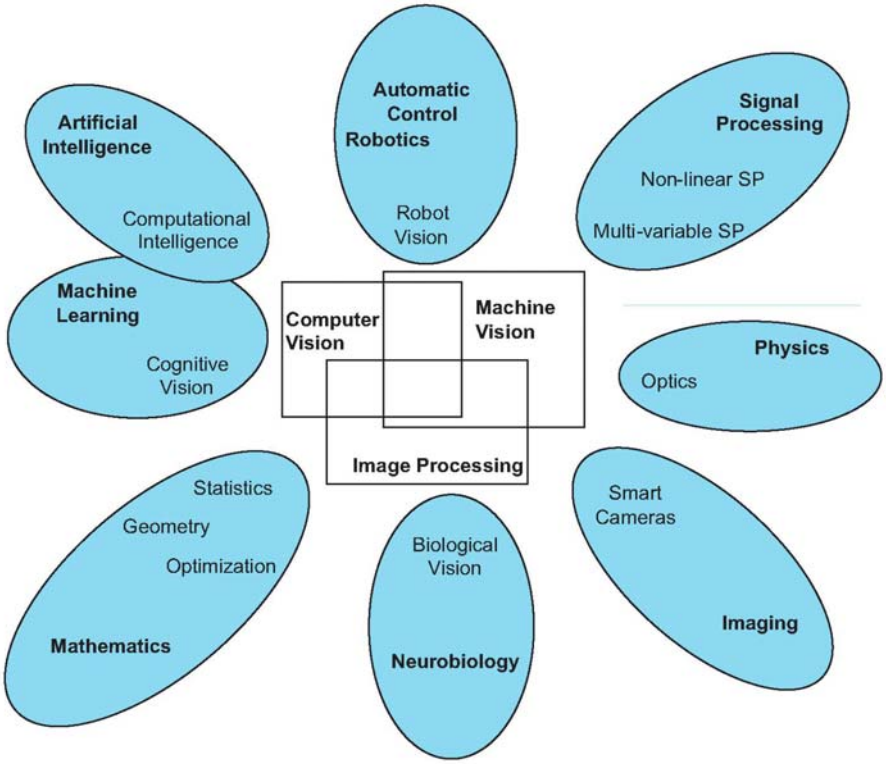


Εικόνα 8 Αριστερά :Σχετική θέση χαρακτηριστικών σημείων Δεξιά: Υπολογιστική προσέγγιση

Μηχανική όραση

Εισαγωγή

Η όραση μηχανών (Machine Vision) είναι η εφαρμογή της όρασης υπολογιστών στη βιομηχανία και τις κατασκευές. Ενώ η όραση υπολογιστών στρέφεται κυρίως στη επεξεργασία εικόνας βασισμένη σε μηχανήματα, η όραση μηχανών συχνότερα απαιτεί επίσης τις ψηφιακές συσκευές εισόδου-εξόδου και δίκτυα υπολογιστών για να ελέγξει άλλους κατασκευαστικούς εξοπλισμούς όπως οι ρομποτικοί βραχίονες. Η όραση μηχανών είναι ένας τομέας της εφαρμοσμένης μηχανικής που καλύπτει την πληροφορική, την οπτική, τη μηχανική, και τη βιομηχανική αυτοματοποίηση. Μια από τις πιο κοινές εφαρμογές της όρασης μηχανών είναι η επιθεώρηση κατασκευασμένων προϊόντων όπως τα τσιπ, τα αυτοκίνητα, τα τρόφιμα και τα φαρμακευτικά είδη. Ακριβώς όπως οι επιθεωρητές που εργάζονται στη γραμμή παραγωγής επιθεωρούν οπτικά τα διάφορα στάδια παραγωγής για να κρίνουν την ποιότητα της εργασίας, έτσι τα συστήματα όρασης μηχανών χρησιμοποιούν τις ψηφιακές κάμερες, τις έξυπνες κάμερες και το λογισμικό επεξεργασίας εικόνας για να εκτελέσουν τις παρόμοιες επιθεωρήσεις.



Εικόνα 9 Σχέση της υπολογιστικής όρασης με άλλα επιστημονικά πεδία

Τα συστήματα όρασης μηχανών είναι προγραμματισμένα να εκτελούν λεπτομερώς καθορισμένες εργασίες όπως η μέτρηση αντικειμένων σε έναν μεταφορικό ιμάντα, η ανάγνωση των αυξόντων αριθμών, και η αναζήτηση ατελειών στις επιφάνειες των προϊόντων. Οι κατασκευαστές προτιμούν τα συστήματα όρασης μηχανών για τις οπτικές επιθεωρήσεις οι οποίες απαιτούν μεγάλη ταχύτητα, μεγάλη μεγέθυνση της εικόνας, εικοσιτετράωρη λειτουργία, και συχνή επανάληψη των μετρήσεων. Αυτές οι εργασίες επεκτείνουν τους ρόλους που ανατίθενται στις μηχανές και που παραδοσιακά καταλαμβάνονταν από ανθρώπους, των οποίων ο βαθμός αποτυχίας είναι υψηλός λόγω απόσπασης της προσοχής, ασθένειας και συνθηκών εργασίας. Οι άνθρωποι βέβαια επιδεικνύουν μεγαλύτερη αντίληψη κατά τη διάρκεια της μικρής χρονικής περιόδου καθώς και μεγαλύτερη ευελιξία στην ταξινόμηση και προσαρμογή στις νέα σφάλματα και δυσλειτουργίες στη γραμμή παραγωγής και τις πολιτικές εξασφάλισης ποιότητας.

Οι υπολογιστές «δεν βλέπουν» με τον ίδιο τρόπο που οι άνθρωποι είναι σε θέση να δουν. Οι κάμερες δεν είναι ισοδύναμες με την ανθρώπινη οπτική αίσθηση και ενώ οι άνθρωποι μπορούν να στηριχθούν σε υποθέσεις και να οδηγηθούν σε συμπεράσματα οι μηχανές «βλέπουν» εξετάζοντας μεμονωμένα στοιχεία εικόνας (pixel) των εικόνων, και προσπαθούν να καταλήξουν σε συμπεράσματα με τη βοήθεια κατάλληλων βάσεων δεδομένων και αναγνώρισης μοτίβων. Αν και μερικοί αλγόριθμοι όρασης μηχανών έχουν αναπτυχθεί για να μιμηθούν την ανθρώπινη οπτική αντίληψη, διάφορες αξιόλογες μέθοδοι επεξεργασίας έχουν αναπτυχθεί για την επεξεργασία εικόνων και να προσδιορίσουν τα σχετικά χαρακτηριστικά γνωρίσματα εικόνας κατά τρόπο αποτελεσματικό και συνεπή. Τα συστήματα όρασης μηχανών όπως και τα συστήματα όρασης υπολογιστών είναι ικανά να επεξεργαστούν εικόνες με συνέπεια.

Δυστυχώς τα βασισμένα σε υπολογιστές συστήματα επεξεργασίας εικόνας σχεδιάζονται ώστε να εκτελούν ενιαίες, επαναλαμβανόμενες εργασίες, και παρά τις σημαντικές βελτιώσεις στον τομέα, κανένα σύστημα όρασης μηχανών ή όρασης υπολογιστών δεν μπορεί ακόμα να προσεγγίσει μερικές ικανότητες της ανθρώπινης όρασης όπως η ανοχή στις αλλαγές φωτισμού, την υποβάθμιση εικόνας ή της μεταβλητότητας των αντικειμένων και του περιβάλλοντος.

Συστατικά ενός συστήματος όρασης μηχανών

Ένα χαρακτηριστικό σύστημα όρασης μηχανών θα αποτελείται από αρκετά συστατικά μέρος των οποίων είναι τα ακόλουθα:

Μία ή περισσότερες ψηφιακές ή αναλογικές κάμερες (ασπρόμαυρες ή έγχρωμες).

Διεπαφή interface κάμερας για την ψηφιοποίηση των εικόνων (ευρέως γνωστή και ως frame grabber).

Ένας επεξεργαστής (συχνά ένας Η/Υ ή ένας ενσωματωμένος επεξεργαστής, όπως ένα DSP).

Υλικό εισόδου-εξόδου (π.χ. ψηφιακή Input/Output) ή συνδέσεις επικοινωνίας (π.χ. σύνδεση δικτύου ή rs-232) για να αναφέρουν τα αποτελέσματα.

Φακοί για να προσδιοριστεί το επιθυμητό οπτικό πεδίο στον αισθητήρα εικόνας.

Κατάλληλες, συχνά πολύ εξειδικευμένες, πηγές φωτός (LED φωτιστικά , λαμπτήρες φθορισμού ή αλογόνου)/

Κατάλληλο λογισμικό για την επεξεργασία εικόνων και ανίχνευσης σχετικών χαρακτηριστικών γνωρισμάτων.

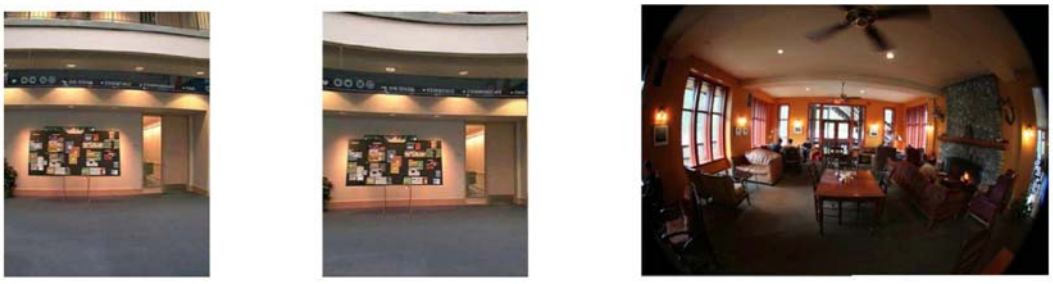
Όλα τα παραπάνω στοιχεία εκτός από το τελευταίο αποτελούν συνήθως μέρος της ειδικής κάμερας που είναι ενσωματωμένη σε ένα ρομπότ, ενώ το λογισμικό επεξεργασίας εικόνας είναι ενσωματωμένο στο λογισμικό του ρομπότ που είναι υπεύθυνο για τις λειτουργίες του (τον «εγκέφαλο» του ρομπότ).

Αισθητήρες όρασης (Cameras)

Η δουλειά μίας κάμερας είναι να συλλάβει μία (ή και περισσότερες) εικόνες και να τις μετατρέψει σε ψηφιακή πληροφορία, την οποία το ρομπότ μπορεί να καταλάβει και να επεξεργαστεί. Πριν προχωρήσουμε στην αναλυτική παρουσίαση του της όρασης των ρομπότ θεωρούμε σκόπιμο να παρουσιάσουμε σύντομα τους βασικούς κανόνες που διέπουν το σχηματισμό μιας ψηφιακής εικόνας μέσα από έναν κοινό αισθητήρα όρασης (κάμερα) και πώς αυτή γίνεται αντιληπτή από ένα υπολογιστικό σύστημα. Η σκοπιμότητα αυτή δικαιολογείται από το γεγονός ότι η κατανόηση του σχηματισμού εικόνας είναι απαραίτητη για την κατανόηση των μεθόδων, σύμφωνα με τις οποίες ανακτούμε πληροφορίες από τις εικόνες.

Φακός

Όλα τα συστήματα απεικόνισης χρησιμοποιούν συγκεκριμένους φακούς, αναλόγως το σκοπό της χρήσης του αισθητήρα. Ωστόσο, κάθε είδος φακού, προκαλεί ένα είδος αλλοίωσης του αντικειμένου που προβάλλεται στην εικόνα, παραμορφώνοντας το σχήμα του.



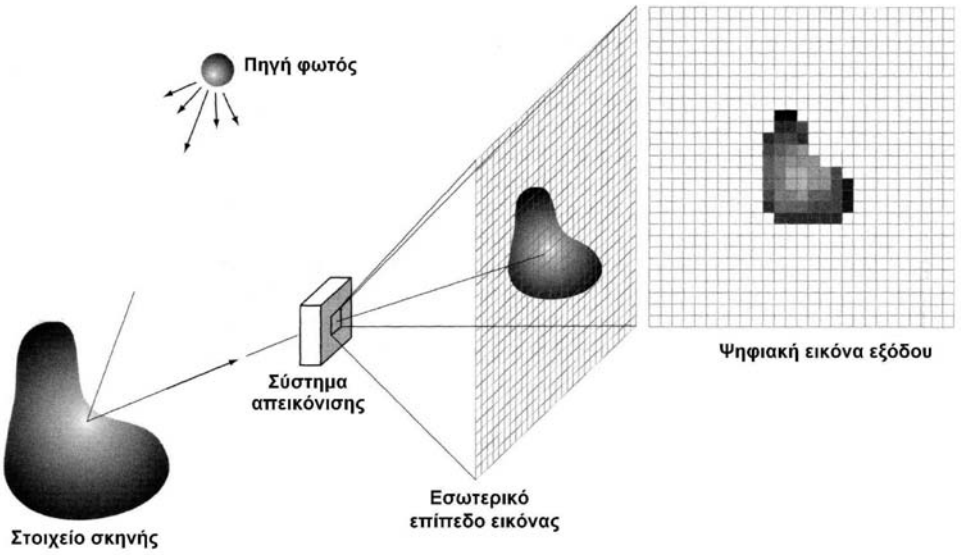
(a) Barrel (b) Pincushion (c) Fisheye

Εικόνα 10 Τυπικά σχήματα παραμόρφωσης φακού

Ένα ορατό, με γυμνό μάτι, χαρακτηριστικό γνώρισμα των κοινών φακών είναι η καμπύλωση των ευθειών. Ευθείες γραμμές προβάλλονται σαν καμπύλες, η καμπύλωση των οποίων γίνεται εντονότερη στα άκρα του πεδίου εστίασης του αισθητήρα. Οι παραμορφώσεις των φακών εξαρτώνται από το είδος και την κυρτότητα του φακού και αντιμετωπίζονται συνήθως σαν ένα είδος ακτινικής παραμόρφωσης (radial distortion). Τυπικά σχήματα παραμόρφωσης είναι οι παραμορφώσεις τύπου barrel, pincushion και fisheye, με την τρίτη να είναι και η εντονότερη. Στο παραπάνω σχήμα φαίνονται παραδείγματα τέτοιων παραμορφώσεων. Για να αντιμετωπιστούν τέτοιου είδους παραμορφώσεις η στρατηγική που ακολουθείται σε αυτές τις περιπτώσεις είναι είτε η ανακατασκευή των εικόνων αντισταθμίζοντας την ακτινική παραμόρφωση και κατόπιν η αντιστοίχιση τους, είτε η απευθείας ενσωμάτωση των μοντέλων αυτών στη διαδικασία της αντιστοίχισης.

Ψηφιοποίηση Εικόνας

Ο φακός μίας κάμερας συλλαμβάνει την εικόνα σαν αναλογική πληροφορία. Για να μπορέσει μία μηχανή να επεξεργαστεί αυτή την πληροφορία πρέπει να την μετατρέψει σε ψηφιακή πληροφορία. Ο πλέον διαδεδομένος τρόπος απεικόνισης εικόνων, ο οποίος χρησιμοποιείται και στις ψηφιακές κάμερες, βασίζεται στη χρήση πλέγματος αισθητήρων.



Εικόνα 11 Σχηματισμός ψηφιακής εικόνας

Ένας τυπικός αισθητήρας για αυτές τις κάμερες είναι ένα CCD (Charge-Couple Device) πλέγμα, το οποίο ουσιαστικά αποτελεί ένα πλέγμα φωτοευαίσθητων κυττάρων και θα αναλυθεί διεξοδικά παρακάτω. Η εισερχόμενη ενέργεια μετατρέπεται σε τάση συνδυάζοντας την ηλεκτρική ισχύ της εισόδου με το υλικό

του αισθητήρα, το οποίο είναι αρκετά ευαίσθητο σε τέτοιου είδους μορφές ενέργειας. Η κυματομορφή της τάσης εξόδου είναι η απόκριση του αισθητήρα, η οποία είναι ανάλογη της συνολικής ενέργειας φωτός που προσπίπτει πάνω στην επιφάνειά του.

Η πρώτη λειτουργία που υλοποιείται από το σύστημα απεικόνισης, είναι η συλλογή της εισερχόμενης ενέργειας και η συγκέντρωσή της πάνω σε ένα επίπεδο εικόνας. Το πλέγμα παράγει έξοδο ανάλογη με το σύνολο του φωτός που λαμβάνεται από κάθε αισθητήρα. Ψηφιακά και αναλογικά κυκλώματα, σαρώνουν τις εξόδους και τις μετατρέπουν σε σήμα video, το οποίο ψηφιοποιείται στη συνέχεια από ένα άλλο τμήμα του συστήματος καταγραφής εικόνας. Η έξοδος τελικά είναι μια ψηφιακή εικόνα.

Επεξεργασία εικόνας

Από τη στιγμή που η μηχανή έχει στην διάθεσή της μία ψηφιακή εικόνα πρέπει να την επεξεργαστεί ώστε να καταλάβει που βρίσκονται διάφορα αντικείμενα και σημεία ενδιαφέροντος μέσα σε αυτή.

Βελτίωση εικόνας

Όπως και σε κάθε αναλογική μέτρηση που μετατρέπεται σε ψηφιακή έτσι μία ψηφιακής εικόνας περιέχει θόρυβο, ο οποίος μπορεί να αλλοιώσει τα αποτελέσματα της επεξεργασίας, οπότε πρώτο βήμα στην διαδικασία είναι η αφαίρεση του θορύβου. Η πρώτη και απλούστερη προσέγγιση είναι η εφαρμογή μαθηματικών φίλτρων, τα οποία μπορούν να καταλάβουν πια εικονοστοιχεία δεν ανήκουν στην εικόνα και να τα αντικαταστήσουν με αυτά που θεωρούν ότι θα έπρεπε να βρίσκονται στην θέση τους.

Εικόνα 12 Χαρακτηριστικό παράδειγμα αφαίρεσης θορύβου

Μία πιο σύνθετη προσέγγιση είναι η σύγκριση κάθε εικόνας με την προηγούμενη, ώστε να αναγνωριστούν τα κοινά σημεία και να διορθωθούν. Για να γίνει κάτι τέτοιο πρέπει η εικόνα να αναλυθεί και να χωριστεί στα απαραίτητα αντικείμενα, τα οποία έπειτα συγκρίνονται με τις προηγούμενες εικόνες τους. Σε αυτή την περίπτωση το αποτέλεσμα είναι συνήθως αρκετά βελτιωμένο σε σχέση με την απλή επεξεργασία εικόνας αλλά χρειάζεται μία αρκετά πιο πολύπλοκη διαδικασία. Επίσης η τυχόν λάθος αναγνώριση ή αντιστοίχηση ενός αντικειμένου με ένα άλλο που είχε προηγουμένως αναγνωριστεί μπορεί να δημιουργήσει πολύ μεγάλα σφάλματα.

Κατάτμηση εικόνας

Αφού η μηχανή έχει στην διάθεση της μία εικόνα χρειάζεται να καταλάβει τι αντικείμενα περιέχει αυτή η εικόνα. Αυτό γίνεται χωρίζοντας την εικόνα στα επιμέρους αντικείμενα βάση κάποιον κανόνων, πρόβλημα που ονομάζεται κατάτμηση εικόνας (image segmentation). Ο στόχος είναι ο χωρισμός της εικόνας σε επιμέρους αντικείμενα, ώστε να δημιουργηθούν ομάδες κοινών εικονοστοιχείων τα οποία μπορεί να ανήκουν σε κοινά αντικείμενα.

Εικόνα 13 Παράδειγμα κατάτμησης εικόνας

Υπάρχουν διάφορες τεχνικές με τις οποίες μπορεί να χωριστεί μία εικόνα σε τμήματα. Παρακάτω θα δούμε κάποιες τεχνικές που συνήθως χρησιμοποιούνται.

Μέθοδος συμπίεσης

Αυτή η μέθοδος βασίζεται στην αρχή ότι η καλύτερη κατάτμηση είναι αυτή που μειώνει όσο το δυνατόν περισσότερο τον όγκο των τελικών δεδομένων. Λέγοντας συμπίεση δεν εννοούμε την μείωση της ανάλυσης της εικόνας ή την μείωση του όγκου που καταλαμβάνει στην μνήμη βάση κάποιου κλασσικού αλγόριθμου συμπίεσης όπως αυτούς που χρησιμοποιούνται στα αρχεία εικόνας, πχ gif ή jpeg. Σε αντίθεση με αυτούς τους αλγόριθμους οι οποίοι προσπαθούν να μειώσουν το μέγεθος όλων των pixel, η κατάτμηση με χρήση συμπίεσης προσπαθεί να ομαδοποιήσει τα εικονοστοιχεία που είναι γειτονικά και χρωματικά συναφή ώστε κάθε αντικείμενο να περιγράφεται από ένα στοιχείο για το εσωτερικό του και από το όριο του (το περίγραμμά του).

Κάθε τμήμα λοιπόν κωδικοποιείται από την υφή του (το χρώμα του) και το σχήμα του περιγράμματός του. Τα τμήματα μοντελοποιούνται από μία συνάρτηση κατανομής πιθανοτήτων και το μέγεθος των δύο στοιχείων του μετά την κωδικοποίηση.

- Ο υπολογισμός του περιγράμματος γίνεται με βάση το γεγονός ότι στις περισσότερες εικόνες το περίγραμμα των σχημάτων είναι συνεχές και ομοιόμορφο. Με χρήση του κώδικα Huffman υπολογίζεται η αλυσίδα των πιθανών περιγραμμάτων και έπειτα επιλέγονται αυτά με το μικρότερο μήκος.
- Το εσωτερικό του σχήματος υπολογίζεται χρησιμοποιώντας διαδοχικές εφαρμογές ενός αλγορίθμου απωλεστικής συμπίεσης, αλλά το μέγεθος κάθε τμήματος υπολογίζεται σαν το γινόμενο του αριθμού των εφαρμογών επί το μέγεθος της εντροπίας της εικόνα.

Για κάθε δοθείσα κατάτμηση της εικόνας υπολογίζεται, βάση των παραπάνω, το μέγεθος που απαιτείται για την συμπίεσή της, με στόχο να βρεθεί το ελάχιστο.

Μέθοδος βασισμένη σε ιστογράμματα

Αυτή η μέθοδος βασίζεται στην ομαδοποίηση των εικονοστοιχείων βάση του χρώματος ή της φωτεινότητάς τους. Αντικείμενα θεωρούνται αρχικά τα εικονοστοιχεία που διαφέρουν περισσότερο μεταξύ τους και σε αυτά προστίθενται οι ομάδες εικονοστοιχείων που έχουν μεγαλύτερη ομοιότητα με αυτά.

Αν και αυτή η μέθοδος είναι αρκετά γρήγορη επειδή χρειάζεται πολύ μικρή υπολογιστική ισχύς είναι προβληματική στον διαχωρισμό αντικειμένων που μοιάζουν χρωματικά και στις περιπτώσεις που μία εικόνα είναι σκοτεινή και κατά συνέπεια τα χρώματα των διαφόρων αντικειμένων μοιάζουν.

Εύρεση ακμών

Η εύρεση ακμών προσπαθεί να διακρίνει τα ξεχωριστά αντικείμενα ανιχνεύοντας τις ακμές τους βασιζόμενη στην παραδοχή ότι τα διαφορετικά αντικείμενα έχουν μεγάλες διαφορές στις άκρες τους με τα γειτονικά τους αντικείμενα. Μία μεταβολή στην φωτεινότητα μπορεί να υποδεικνύει μία ασυνέχεια στο βάθος, στον προσανατολισμό της επιφάνειας, στο υλικό της επιφάνειας ή στο φωτισμό της εικόνας.

Στην ιδανική περίπτωση η εφαρμογή ενός αλγόριθμου εύρεσης ακμών θα δώσει ένα σύνολο ακμών που θα συμπίπτουν με τα περιγράμματα των αντικειμένων που υπάρχουν σε μία εικόνα , επιτρέποντας να βρούμε τα αντικείμενα που περιέχονται μέσα στα σχήματα που αυτές σχηματίζουν. Σε αυτή την περίπτωση τα σχήματα που προκύπτουν δεν είναι απαραίτητο να έχουν όλα το ίδιο χρώμα, επιτρέποντας την ανίχνευση αντικειμένων που έχουν πολλά χρώματα ή διαφορετική φωτεινότητα.

Παρόλα αυτά σε περισσότερο πολύπλοκες εικόνες οι ακμές που προκύπτουν δεν σχηματίζουν κλειστά σχήματα και κατά συνέπεια δεν προσφέρουν μία πλήρη κατάτμηση, ενώ μπορεί να περιέχουν και ψεύτικες ακμές, οι οποίες δυσκολεύουν την ανίχνευση. Κάποιες από τις κυριότερες δυσκολίες μπορεί να είναι θολά σημεία της εικόνας, τα οποία προκύπτουν είναι από το εστιακό βάθος του φακού είτε από αντανακλάσεις, αλλαγές στην φωτοσκίαση του ιδίου αντικειμένου ή αντικείμενα με ακανόνιστο σχήμα, ιδιαίτερα όταν υπερκαλύπτουν τμήματα άλλων αντικειμένων. Ιδιαίτερα απαιτητικές είναι οι επιφάνειες που αλλάζουν ομαλά χρώματα. Μία τέτοια επιφάνεια μπορεί να υποδηλώνει είτε αλλαγή στο χρώμα ενός αντικειμένου, είτε αλλαγή στην σκιά που μπορεί να πέφτει επάνω του ή και πολλά αντικείμενα με παραπλήσια χρώματα.

Τεχνικές εύρεσης ακμών

Οι περισσότερες μέθοδοι ευρέσεις ακμών μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες, *συνολικής αναζήτησης* και *εναλλαγής πρόσημου*. Οι μέθοδοι συνολικής αναζήτησης αρχικά υπολογίζουν ένα γενικό μέγεθος πρώτου βαθμού για την εικόνα (πχ το μέσο χρώμα) και έπειτα αναλύουν τα εικονοστοιχεία ώστε να βρουν ελάχιστα και μέγιστα βάση αυτού του μέγεθος. Οι μέθοδοι εναλλαγής πρόσημου λειτουργούν ερευνώντας τα σημεία όπου η παράγωγος ενός μεγέθους αλλάζει πρόσημο.

Στις περισσότερες περιπτώσεις πριν την επεξεργασία χρησιμοποιείται ένα φίλτρο λείανσης με σκοπό την μείωση του θορύβου και διαφέρουν μεταξύ τους στο είδος των φίλτρων και τον τρόπο με τον οποίο αυτό εφαρμόζεται.

Εικόνα 8 Η ποιότητα της εικόνας και του φωτισμού επιρεάζει τα αποτελέσματα της εύρεσης ακμών

Εύρεση ακμών Canny

Αυτή η μέθοδος πήρε το όνομα της από τον John Canny, ο οποίος μελέτησε το πρόβλημα της δημιουργίας ενός βελτιστοποιημένου φίλτρου λείανσης (smoothing filter) με κριτήρια αξιολόγησης την ανίχνευση ακμών, την εύρεση ακμών σε τοπικά

όρια αντικειμένων που διαφέρουν ελάχιστα σε σχέση με τις διαφορές που παρουσιάζονται σε ολόκληρη την εικόνα και την εξάλειψη της ανίχνευσης πολλών ακμών σε περιοχές που υπάρχει μόνο μία.

Ο Canny έδειξε ότι το φίλτρο που πληρούσε καλύτερα αυτά τα κριτήρια είναι το άθροισμα τεσσάρων εκθετικών όρων, καθώς επίσης ότι αυτό μπορεί να προσεγγιστεί με ικανοποιητική ακρίβεια εξετάζοντας τις παραγώγους πρώτου βαθμού γκαουσσιανών συναρτήσεων. Επίσης εισήγαγε την έννοια της μη μέγιστης καταστολής που σημαίνει ότι δεδομένης της εφαρμογής των αρχικών φίλτρων λείανσης τα σημεία που αποτελούν μία ακμή ορίζονται σαν τα σημεία όπου η τάξη της χρωματικής διαφοράς παίρνει ένα τοπικό μέγιστο.

Εύρεση ορίων και ένωση σημείων ακμών

Μετά τον υπολογισμό ενός μεγέθους για το πόσο διαφέρουν τα σημεία της εικόνας μεταξύ τους, που βασίζεται συνήθως στην χρωματική διαφορά, το επόμενο βήμα είναι η επιβολή ενός ορίου βάση του οποίου θα επιλεγούν όσα σημεία το ξεπερνάνε σαν ακμές. Το όριο αυτό επηρεάζει το πλήθος των ακμών που θα ανιχνευτούν και το πόσο ευαίσθητα θα είναι τα αποτελέσματα στον θόρυβο. Αν είναι χαμηλό θα βρεθούν περισσότερες ακμές αλλά πολλές από αυτές θα είναι αποτελέσματα του θορύβου και όχι πραγματικά όρια ανάμεσα στα αντικείμενα. Αντίθετα ένα υψηλό όριο σημαίνει ότι θα υπάρχουν λιγότερες ακμές με κίνδυνο την παράληψη κάποιων πραγματικών ακμών ή την ανίχνευση διακεκομμένων ακμών αλλά θα μειωθεί αρκετά το πόσο επηρεάζεται το αποτέλεσμα από τον θόρυβο.

Αν το όριο είναι ανάλογο κάποιου συνολικού μεγέθους για την εικόνα, πχ το μέσο χρώμα, θα υπάρχουν πολλές παχιές ακμές κάνοντας απαραίτητη την λέπτυνσή τους με κάποιον επιπλέον αλγόριθμό που θα εφαρμοστεί μετά την ανίχνευση. Αντίθετα για ακμές που ανιχνεύονται χρησιμοποιώντας σαν την μη μέγιστη καταστολή το αποτέλεσμα είναι αρκετά λεπτές ακμές που έπειτα πρέπει να επεκταθούν για να συμπεριλάβουν και τα γειτονικά τους στοιχεία.

Μέθοδοι αναπτυσσόμενων περιοχών

Με αυτή τη μέθοδο χρειάζεται αρχικά να επιλεχθούν κάποια αρχικά σημεία της εικόνας από τα οποία θα ξεκινήσει η ανάλυση. Αυτά τα σημεία μπορεί να δίνονται αρχικά σαν είσοδος στον αλγόριθμο, να επιλέγονται τυχαία ή να επιλέγονται βάση κάποιων κριτηρίων. Κάθε σημείο είναι μία περιοχή της εικόνας και θα αντιστοιχεί σε ένα διαφορετικό αντικείμενο του αποτελέσματος.

Σε κάθε βήμα της μεθόδου επιλέγεται ένα νέο σημείο της εικόνας. Αρχικά υπολογίζεται η απόσταση όλων των σημείων που γειτονεύουν με τα υπάρχοντα σχήματα από τα σχήματα αυτά. Η απόσταση βρίσκεται υπολογίζοντας την διαφορά της έντασης του σημείου με την μέση ένταση των στοιχείων που περιλαμβάνονται ήδη σε μία περιοχή. Έπειτα το σημείο με την ελάχιστη απόσταση προστίθεται στην

γειτονική του περιοχή και η διαδικασία επαναλαμβάνεται έως ότου όλα τα σημεία ανήκουν σε κάποια περιοχή.

Η ποιότητα των αποτελεσμάτων εξαρτάται σε μεγάλο βαθμό με από τα αρχικά σημεία, γύρω από τα οποία αναπτύσσονται οι περιοχές, ενώ ο θόρυβος μπορεί να προκαλέσει αρκετά μεγάλα προβλήματα στην τοποθέτηση των αρχικών σημείων και στην επέκταση των περιοχών.

Μία παραλλαγή που αναπτύχθηκε για να καταπολεμήσει αυτό το πρόβλημα δεν χρειάζεται κάποια αρχικά σημεία αλλά ξεκινάει από ένα τυχαίο σημείο, το οποίο αποδεικνύεται ότι δεν επηρεάζει και το αποτέλεσμα. Ο αλγόριθμος σε κάθε βήμα υπολογίζει την απόσταση των γειτονικών σημείων από την αρχική περιοχή όπως πριν αλλά για να προσθέσει ένα νέο σημείο σε αυτή πρέπει η απόστασή του να μην ξεπερνάει μία τιμή κατωφλίου. Αν την ξεπερνάει τότε θεωρεί ότι το νέο σημείο δεν ανήκει στην υπάρχουσα περιοχή και δημιουργεί μία νέα που το περιέχει. Η τιμή κατωφλίου μπορεί να δίνεται αρχικά ή να εξάγεται από όλη την εικόνα, πχ το μέσο χρώμα.

Κατάτμηση βασιζόμενη σε μοντέλα

Πρόκειται για μία διαφορετική προσέγγιση του προβλήματος η οποία ξεκινάει με την μηχανή να έχει μία βάση δεδομένων με αντικείμενα που υπάρχει περίπτωση να συναντήσει και αναλύει την εικόνα προσπαθώντας να βρει μέσα σε αύτη κάποιο από αυτά. Η αναζήτηση γίνεται αναλύοντας τα αντικείμενα της εικόνας και έπειτα με την χρήση ενός μοντέλου πιθανοτήτων να εκτιμηθεί κατά είναι πιθανό πόσο ένα σχήμα μπορεί να είναι μία παραλλαγή ενός υπάρχοντος μοντέλου.

Στερεοσκοπική Αντιστοίχιση

Η στερεοσκοπική όραση είναι ένα πεδίο το οποίο στοχεύει στην ικανότητα υπολογιστικών συστημάτων ή ρομπότ να αντιλαμβάνονται την τρίτη διάσταση, δηλαδή το βάθος της σκηνής. Αυτό επιτυγχάνεται με την εκτίμηση της απόστασης κάθε σημείου της σκηνής ως προς το σημείο παρατήρησης. Ο συνήθης αλγόριθμος, τον οποίο ακολουθεί και ο ανθρώπινος εγκέφαλος, είναι ο συνδυασμός της πληροφορίας εικόνων από διαφορετικά σημεία λήψης, με γνωστή τη σχετική τους απόσταση. Στις μέρες μας υπάρχουν αισθητήρες που έχουν τη δυνατότητα αποτύπωσης εικόνων των οποίων η τιμή έντασης είναι μόνο συνάρτηση της απόστασης (βάθους) του αντίστοιχου σημείου της σκηνής και αναφέρονται ως ακτινικές εικόνες (range images). Από αυτές τις εικόνες είναι προφανές ότι μπορούμε κατευθείαν να υπολογίσουμε το βάθος της σκηνής ενδιαφέροντος. Δυστυχώς το κόστος τέτοιων αισθητήρων-συστημάτων είναι αρκετά υψηλό και επομένως το πρόβλημα της εκτίμησης της πληροφορίας του βάθους από διαφορετικές προβολές έντασης φωτεινότητας της ίδιας σκηνής εξακολουθεί να έχει ενδιαφέρον.

(a) Αριστερή εικόνα (b) Δεξιά εικόνα (c) Χάρτης ανομοιότητας

Εικόνα 15 Δημιουργία χάρτη ανομοιότητας

Το κύριο πρόβλημα βέβαια είναι αυτό της αντιστοίχισης εικόνων, η επίλυση του οποίου απαιτεί πολύ μεγαλύτερο υπολογιστικό κόστος. Η προεπεξεργασία που λαμβάνει χώρα μαζί με τη βαθμονόμηση αισθητήρων, αφορά τυχόν ανακατασκευή εικόνων, έτσι ώστε το πρόβλημα της αντιστοίχισης να απλοποιείται αρκετά, όπως θα δούμε. Το πρόβλημα της βαθμονόμησης αισθητήρων μπορεί να αγνοηθεί, αν τα χαρακτηριστικά του στερεοσκοπικού συστήματος που χρησιμοποιείται είναι γνωστά.

Η αντιστοίχιση συνδέεται με την ανάκτηση της πληροφορίας βάθους, αφού αν γνωρίζουμε τις μετατοπίσεις των σημείων της σκηνής από τη μία εικόνα στην άλλη, μπορούμε να έχουμε μια εκτίμηση του διατακτικού βάθους, δηλαδή ποιο αντικείμενο βρίσκεται πιο κοντά στο σύστημα λήψης (μικρότερο βάθος) σχετικά με κάποιο άλλο. Επιπλέον, αν είναι γνωστά τα χαρακτηριστικά του αισθητήρα όρασης και συγκεκριμένα η εστιακή απόσταση και η απόσταση μεταξύ των αισθητήρων, τότε μπορούμε να υπολογίσουμε ακριβώς το βάθος ενός σημείου ενδιαφέροντος της σκηνής.

Στην περίπτωση αυτή τα δύο προφίλ συνθέτουν δύο εικόνες μιας σκηνής, οι οποίες έχουν ληφθεί από ένα στερεοσκοπικό σύστημα λήψης.

Αλγόριθμοι Σύγκρισης Εικόνων

Γενικά

Στα προηγούμενα κεφάλαια αναφερθήκαμε εκτενώς στο ότι ένα ρομπότ για να καταφέρει να χαρτογραφήσει τον χώρο στον οποίο κινείται κάνει συνεχώς μετρήσεις της θέσης και της απόστασης των αντικειμένων του χώρου. Καθώς μας ενδιαφέρουν ιδιαίτερα οι εφαρμογές της ρομποτικής πλοήγησης που βασίζεται σε οπτικά αισθητήρια άσχετα από την υλοποίησή τους θα εξετάσουμε τον τρόπο με τον οποίο μία εικόνα διασπάται από το ρομπότ σε αντικείμενα και πως μπορεί να διακρίνει το ίδιο αντικείμενο στις διαδοχικές εικόνες ενώ κινείται, όπου το αντικείμενο αλλάζει εμφάνιση εξαιτίας της μεταβολής της απόστασης από αυτό και της γωνίας από την οποία φαίνεται.

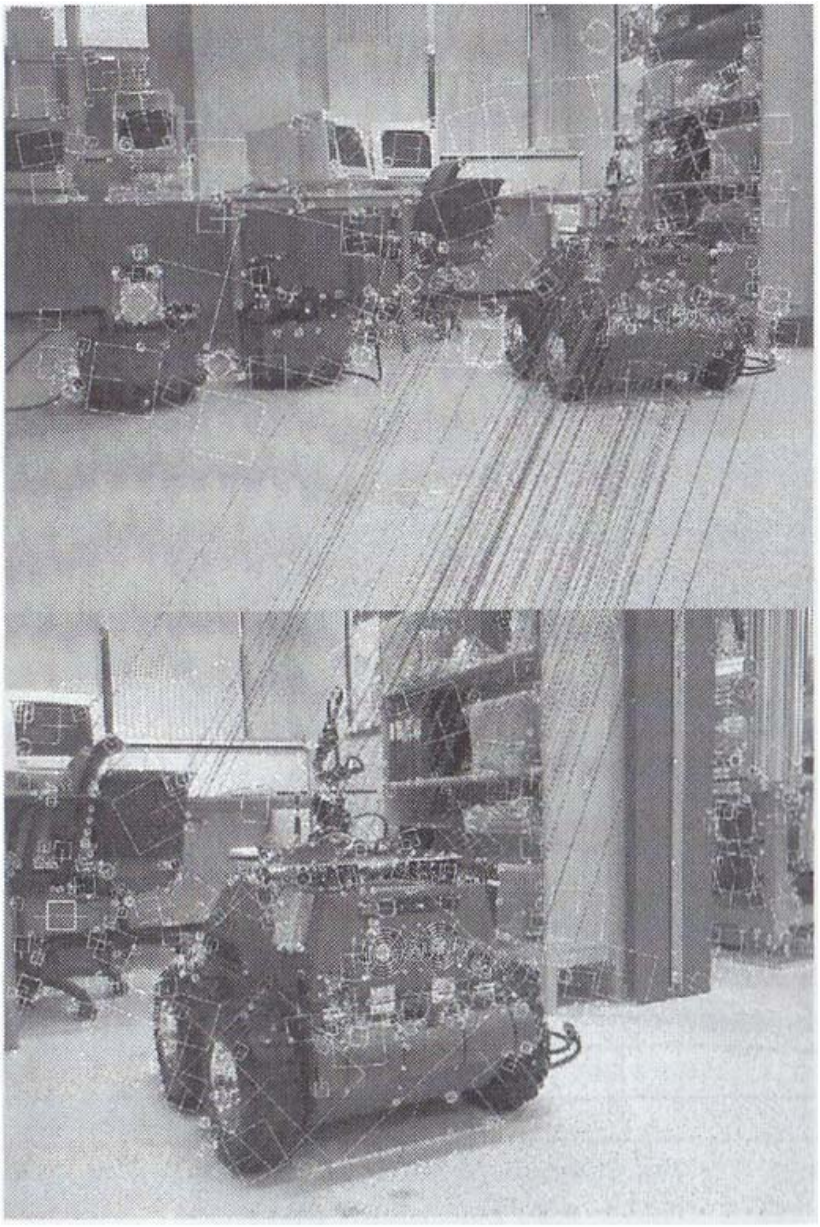
Πριν γίνει οποιαδήποτε σύγκριση σε μία εικόνα είναι απαραίτητη η μετατροπή της σε μία μορφή που είναι περισσότερο κατανοητή από το ρομπότ. Αρχικά επιλέγεται ένας περιγραφές(descriptor), δηλαδή ένα συγκεκριμένο μέγεθος ή μία ομάδα μεγεθών βάση των οποίων η εικόνα μετατρέπεται σε μία άλλη μορφή. Η νέα μορφή, που ονομάζεται και περιγραφή πρέπει να είναι μοναδική και σε αυτή να περιέχεται όλη η πληροφορία που περιέχει η αρχική εικόνα σχετικά με τα χρώματα, την αντίθεση των χρωμάτων και την φωτεινότητα του περιεχομένου.

Το πλεονέκτημα της μετατροπής είναι ότι είναι πολύ πιο εύκολη η υπολογιστική επεξεργασία των περιεχομένων και η μεταξύ τους ομαδοποίηση και σύγκριση χρησιμοποιώντας μαθηματικά μεγέθη. Ανάλογα με το είδος των περιγραφέων που επιλέγονται μία περιγραφή μπορεί να στηρίζεται είτε σε *τοπικά χαρακτηριστικά* είτε σε *ολικά χαρακτηριστικά*.

Τα *σφαιρικά ή ολικά χαρακτηριστικά* συμπυκνώνουν πληροφορίες από όλη την εικόνα σε ένα μόνο περιγραφέα. Συνεπώς για κάθε εικόνα μόνο ένας περιγραφέας δημιουργείται για κάθε γενικό χαρακτηριστικό, ένα παράδειγμα γενικού χαρακτηριστικού είναι το ομαλοποιημένο ιστόγραμμα χρωμάτων NHC – Normalised Colour Histogram το οποίο χρησιμοποιείται για σύγκριση απόδοσης σε εναλλακτικές μεθόδους από αυτές που θα εξετάσουμε.

Τα *τοπικά χαρακτηριστικά* εν αντιθέσει με τα σφαιρικά υπολογίζονται από διαφορικές σημαντικές υποπεριοχές της εικόνας. Για κάθε υποπεριοχή λαμβάνεται υπόψη ένας περιγραφέας, αυτό φαίνεται καθαρά στην παρακάτω εικόνα όπου απεικονίζει τη σύγκριση δύο εικόνων με τα αντικείμενα σε αυτές σε διαφορετική απόσταση και σε ελάχιστα διαφορετική κλίση. Το να συγκρίνουμε δύο εικόνες χρησιμοποιώντας τοπικά χαρακτηριστικά είναι τυπικά περισσότερο χρονοβόρο από το να συγκρίνουμε ένα καθολικό περιγραφέα, αφού κάθε τοπικό χαρακτηριστικό

πρέπει να συγκριθεί με τα πολλά τοπικά χαρακτηριστικά που έχουμε εξάγει από την άλλη εικόνα εκ των δύο που συγκρίνονται κάθε φορά. Η διαφορά αυτή στην επίδοση προκύπτει από το γεγονός ότι πολλά τοπικά χαρακτηριστικά υπολογίζονται με βάση όλη την εικόνα. Εάν παρεμποδίσουμε μέρη της εικόνας το καθολικό χαρακτηριστικό θα επηρεαστεί αρκετά. Με τα τοπικά χαρακτηριστικά πολλά χαρακτηριστικά σημεία με τους αντίστοιχους περιγραφείς εξάγονται τόσο στις ευανάγνωστες επιφάνειες όσο και σε εκείνες που μερικώς αποκρύπτονται. Η διαφορά έγκειται στο γεγονός ότι πολλά τοπικά χαρακτηριστικά θα παραμείνουν όμοια για τους περιγραφείς για τους οποίους η σημαντικότερη υποπεριοχή τους δεν επηρεάζεται ως εκ τούτου τα τοπικά χαρακτηριστικά τυπικά δείχνουν πολύ καλύτερη λειτουργικότητα και αυτοδυναμία σε σχέση με τα καθολικά στα δυναμικά περιβάλλοντα.



Εικόνα 16 Σύγκριση δύο εικόνων βασισμένη σε τοπικά χαρακτηριστικά.

Επίσης όπως φαίνεται και στην παραπάνω εικόνα η απόσταση μεταξύ δύο όμοιων χαρακτηριστικών είναι μερικά μέτρα οι προσεγγιστικές μέθοδοι τοπικών χαρακτηριστικών στις οποίες λαμβάνεται υπόψη η κλίμακα δεν είναι τόσο ευαίσθητες στις αλλαγές της οπτικής γωνίας της κάμερας. Τα τοπικά χαρακτηριστικά είναι επίσης κατάλληλα στη διαχείριση και σύνδεση διασταύρωσης δεδομένων , δηλαδή στο να καθορίζουν ποια δεδομένα αισθητήρων (μέρη των εικόνων που παίρνουμε από την κάμερα) επικαλύπτουν την ίδια φυσική υποπεριοχή.

Παρακάτω θα ασχοληθούμε με τον τρόπο εξαγωγής των τοπικών χαρακτηριστικών και την ταυτοποίηση εικόνων σύμφωνα με τους υπολογισμούς ομοιότητας.

Μετασχηματισμός Χαρακτηριστικών Αμετάβλητα ως προς την Κλίμακα

Η μέθοδος S.I.F.T (Scale Invariant Feature Transform) αναπτύχθηκε από τον Lowe το 1999 και είναι μια μέθοδος εξαγωγής τοπικών χαρακτηριστικών αμετάβλητα στην μετατροπή της κλίμακας, την περιστροφή της εικόνας και μερικώς αμετάβλητα στις διακυμάνσεις του φωτισμού και την παραμόρφωση λόγω προοπτικής. Η εξαγωγή των χαρακτηριστικών της μεθόδου SIFT γίνεται ως εξής :

1) Δημιουργία της χωροκλίμακας, δηλαδή ανάλυση της εικόνας σε μία ομάδα υποεικόνων κάθε μία από τις οποίες χαρακτηρίζεται από μία παράμετρο η οποία συνήθως είναι ο περιγραφέας της εικόνας. Αυτή η διαδικασία γίνεται εξομαλύνοντας συνεχώς την αρχική εικόνα με την μέθοδο Γκαουσιανού πυρήνα (Gaussian Kernel).
   Έχει σημασία πως η χρήση του περιγραφέα για τον χαρακτηρισμό των υποεικόνων αντανακλά το φυσικό μέγεθος που έχουμε επιλέξει για την περιγραφή των εικόνων, πχ την φωτεινότητα ή κάποια παραπλήσια χρώματα.
2) Ανίχνευση ακρότατων χωροκλίμακας. Με αυτόν τον τρόπο βρίσκονται οι ακραίες υποεικόνες και ταξινομούνται όλες οι υποεικόνες βάση των τιμών των ακρότατων που βρίσκονται. Αυτή η διαδικασία γίνεται χρησιμοποιώντας την μέθοδο διαφοράς Γκαουσιανών (difference of Gaussians).
3) Εντοπισμός των σημείων ενδιαφέροντος αξιολογώντας τα αποτελέσματα της προηγούμενης ταξινόμησης. Σε αυτό το βήμα τα ακρότατα που βρέθηκαν πριν αναλύονται σε επίπεδο εικονοστοιχείου και κρατούνται μόνο τα περισσότερο διαφορετικά από αυτά για να αποφευχθεί η διάσπαση ενός σημείου ενδιαφέροντος σε δύο.
4) Αποβολή των αδύναμων σημείων ενδιαφέροντος .Όλα τα ακρότατα τα οποία έχουν χαμηλή αντίθεση και βρίσκονται κατά μήκος των ακμών, δηλαδή γειτονεύουν με άλλα αρκετά διαφορετικά σημεία αποβάλλονται. Κρατάμε λοιπόν εκείνα που παρουσιάζουν τη μέγιστη σταθερότητα.

5) Στο σημείο αυτό υπολογίζεται για κάθε χαρακτηριστικό που επιλέχθηκε ένας ή και περισσότεροι προσανατολισμοί σύμφωνα με τις κλίσεις της εικόνας. Αυτό γίνεται για να αποκτήσουμε περιστροφική αμεταβλητότητα. Αυτό βέβαια γίνεται και σε συνάρτηση με τον προσανατολισμό των γειτονικών σημείων κάθε φορά .Το μέγεθος της γειτονιάς που λαμβάνεται υπόψη εξαρτάται από την κλίμακα που κάναμε τους υπολογισμούς.
6) Υπολογισμός του ιστογράμματος περιγραφέων. Δοθείσας της θέσης , της κλίμακας και του προσανατολισμού του κάθε ενδιαφέροντος σημείου, ένα κομμάτι εικόνας επιλέγεται όπου το μέγεθος και ο προσανατολισμός της κλίσης χρησιμοποιούνται για να δημιουργήσουν μια αναπαράσταση η οποία μας επιτρέπει μέχρι ενός ορίου αλλαγές στο φωτισμό και στους αιφνικούς υπολογισμούς.

Εικόνα 17 Τα σημεία ενδιαφέροντος όπως αρχικά ανιχνεύονται από τον αλγόριθμο S.I.F.T. και καθώς αυτά απορρίπτονται διαδοχικά από τα βήματα του αλγορίθμου

Τροποποιημένος Μετασχηματισμός Χαρακτηριστικών Αμετάβλητα ως προς την Κλιμακα

Η μέθοδος SIFT δημιουργεί χαρακτηριστικά τα οποία είναι αναλλοίωτα στη μετάφραση της εικόνας, την κλίμακα, την περιστροφή και μερικώς αμετάβλητα στη φωτεινότητα και την αιφινική τρισδιάστατη προβολή. Αυτές οι ιδιότητες κάνουν τη μέθοδο καταλληλότατη για τα κινούμενα ρομπότ επειδή τα ορόσημα στο χώρο μπορούν να παραλειφθούν από διαφορετικές γωνίες, αποστάσεις και φωτισμό. Ωστόσο στο θέμα του αυτοεντοπισμού δεν θέλουμε να μην υπάρχει καμία μεταβολή στη μετάφραση και στην κλίμακα. Θα θέλαμε η οπτική αντιπαραβολή να

είναι επιτυχής στον τομέα της περιοχής όπου η αυθεντική εικόνα καταγράφηκε στη βάση δεδομένων. Γι` αυτό και πήραν μέρος διάφορες αλλαγές οι οποίες οδήγησαν στη μέθοδο M.S.I.F.T.(Modified Scale Invariant Feature Transform) .Οι κυριότερες είναι :

Α) Η επιλογή των σημείων ενδιαφέροντος δεν γίνεται πλέον σε χωροκλίμακα. Βασίζεται σε ιδιοτιμές οι οποίες προκύπτουν από τη μέθοδο Hessian.

Β) Ο αριθμός των εξαγόμενων χαρακτηριστικών από τις εικόνες είναι σταθερός και μόνο τα κυρίαρχα χαρακτηριστικά χρησιμοποιούνται. Αυτό μας βοηθά στο να έχουμε συγκεκριμένο και σταθερό χρόνο υπολογισμού μεταξύ της σύγκρισης δύο διαδοχικών εικόνων το οποίο παίζει ιδιαίτερο ρόλο σε πραγματικό χρόνο.

Ταύτιση εικόνων με την χρήση των χαρακτηριστικών

Για κάθε εικόνα που αναλύεται με τις παραπάνω μεθόδους βρίσκουμε κάποιους περιγραφείς, και κάθε περιγραφέας είναι ένα χαρακτηριστικό. Αν έχουμε δύο εικόνες έχουμε και δύο ομάδες περιγραφέων, οπότε για να βρούμε ποια χαρακτηριστικά από την μία εικόνα είναι ίδια με κάποια χαρακτηριστικά από την δεύτερη πρέπει να τα συγκρίνουμε μεταξύ τους για να βγάλουμε ένα μέτρο ομοιότητας.

Για να γίνει αυτό υπολογίζουμε την ευκλείδεια απόσταση των χαρακτηριστικών μεταξύ τους. Όπως είπαμε στην αρχή του κεφαλαίου η εικόνα εκφράζεται με κάποιο μέγεθος που είναι κατανοητό από τον υπολογιστή, δηλαδή με κάποια νούμερα. Η ευκλείδεια απόσταση που θέλουμε να βρούμε είναι η τετραγωνική ρίζα του αθροίσματος των τετραγώνων των διαφορών των αντίστοιχων αριθμών, δηλαδή παρόμοιο με τον τύπο που βρίσκει την απόσταση δύο σημείων. Αλλά αντί για να έχουμε x και y για τις συντεταγμένες των δύο σημείων έχουμε τα μεγέθη που χρησιμοποιούμε για τους περιγραφείς, για παράδειγμα κάποια χρώματα.

Όπως φαίνεται καταλήγουμε με κάποιο νούμερο για κάθε δυνατό ζευγάρι χαρακτηριστικών και χρειάζεται να βρούμε το πιο ταιριαστό ζεύγος. Δεν αρκεί όμως μόνο να πάρουμε το μικρότερο νούμερο, γιατί κάποια νούμερα μπορεί να είναι πάρα πολύ κοντά για να βγει ένα σωστό συμπέρασμα. Ένα κριτήριο που έχει επιβεβαιωθεί εμπειρικά είναι πως ένα ζεύγος αντιστοιχεί στα ίδια χαρακτηριστικά αν η μικρότερη απόσταση είναι μικρότερη από το 60% της δεύτερης μικρότερης απόστασης. Αν υπάρχουν περισσότερα από ένα ζεύγη που πληρούν αυτό το κριτήριο τότε επιλέγεται το μικρότερο.

Όπως είναι φυσικό μπορούμε να αντιστοιχίσουμε ένα χαρακτηριστικό της πρώτης με ένα μόνο χαρακτηριστικό της δεύτερης. Τα αποτελέσματα αυτής της διαδικασίας εξαρτώνται λοιπόν από της σειρά με την οποία επιλέγονται τα ζεύγη, γιατί αν αρχικά αποφασίσουμε ότι ένα χαρακτηριστικό ταυτίζεται με ένα δεύτερο τότε το βγάζουμε από την διαδικασία ταύτισης και παύουμε να το συγκρίνουμε με τα

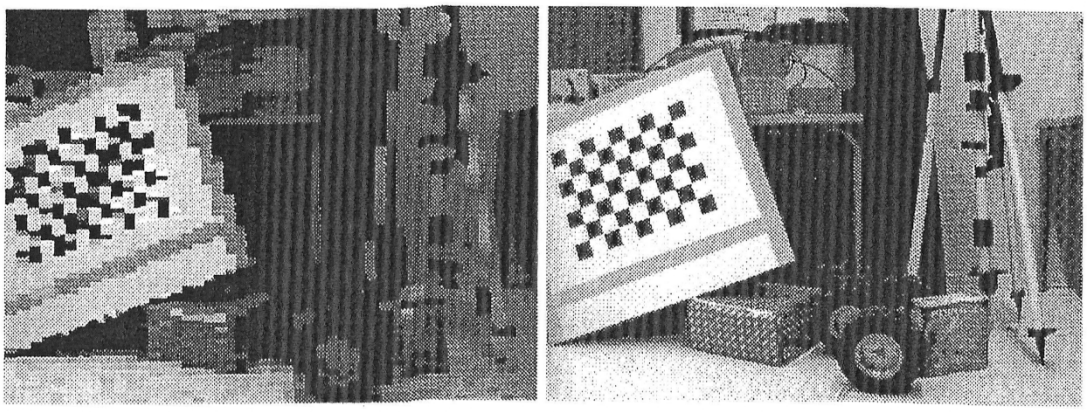
υπόλοιπα, χάνοντας ένα πιθανότερο καλύτερο ζεύγος. Για να αποφύγουμε τέτοια προβλήματα η διαδικασία σύγκρισης μπορεί να γίνει και ανάποδα, δηλαδή την πρώτη για να ταυτίσουμε όλα τα χαρακτηριστικά της πρώτης εικόνας με την δεύτερη και μετά για να ταυτίσουμε τα χαρακτηριστικά της δεύτερης εικόνας με την πρώτη. Στο τέλος επιλέγονται τα ζευγάρια που φαίνεται ότι ταυτίζονται και από τους δύο υπολογισμούς.

Από τα παραπάνω φαίνεται ότι η διαδικασία ταύτισης είναι αρκετά χρονοβόρα καθώς απαιτεί να υπολογίσουμε τις αποστάσεις όλων των πιθανών ζευγαριών χαρακτηριστικών, και ίσως να χρειαστεί να γίνει δύο φορές όπως είπαμε προηγουμένως. Για αυτό χρησιμοποιούνται περισσότερο αποδοτικές δομές αναζήτησης και προσεγγιστικής αναζήτησης, όπως για παράδειγμα η αναζήτηση της πρώτης καλύτερης ομάδας (Best Bin First, BBF).

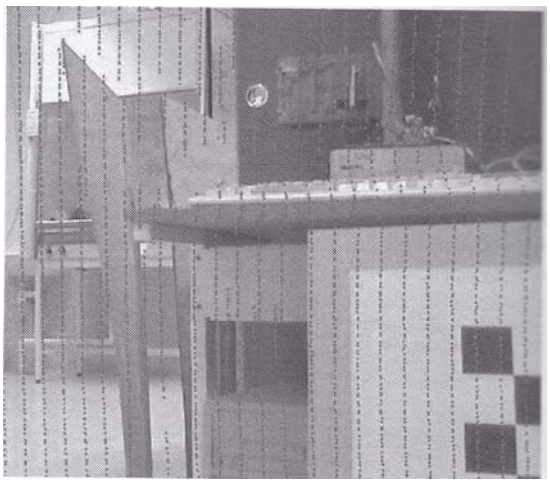
Τρισδιάστατη όραση

Γενικά

Μέχρι τώρα αναφερθήκαμε στο πως γίνεται να αναλύσουμε μία εικόνα ώστε να μπορέσει ένα ρομπότ να πλοηγηθεί σε αυτή, όμως μία εικόνα είναι δισδιάστατη και δεν δίνει καμία πληροφορία για το πόσο απέχουν τα αντικείμενα στην πραγματικότητα. Σε αυτή την ενότητα θα δούμε πως γίνεται να συνδυάσουμε έναν laser αισθητήρα μέτρησης αποστάσεων για να αποκτήσει το ρομπότ την αίσθηση του βάθους.



Εικόνα 18 Αριστερά φαίνεται η ανάλυση του τρισδιάστατου σαρρωτή laser και δεξιά η ανάλυση της κάμερας.



Εικόνα 19 Βλέπουμε μία εικόνα από την ψηφιακή κάμερα συνδυασμένη με τη σάρωση ενός laser scanner.

Το σύνηθες πρόβλημα που παρουσιάζεται σε αυτές τις μεθόδους είναι ότι οι περισσότεροι αισθητήρες laser παρέχουν ένα μοντέλο του χώρου που είναι αρκετά μικρότερης ανάλυσης σε σχέση με τις εικόνες που δημιουργούνται από τους αντίστοιχους οπτικούς αισθητήρες. Για παράδειγμα ένα αντικείμενο που φαίνεται σαν ένα τετράγωνο στο τρισδιάστατο μοντέλο μπορεί να είναι ένας κύλινδρος ή μία σφαίρα στην εικόνα, που επιπλέον είναι πιθανόν να μην έχει ένα ομοιόμορφο χρώμα αλλά πολλά διαφορετικά.

Παρεμβολή βασισμένη στην όραση

Αυτό που θέλουμε να πετύχουμε είναι να παρεμβάλουμε, δηλαδή να αντιστοιχίσουμε κάθε σημείο από το χαμηλής ανάλυσης μοντέλο που κατασκευάζει ο αισθητήρας laser με το αντίστοιχο σημείο ή τμήμα της εικόνας. Γενικά θεωρούμε ότι οι χρωματικές διαφορές και οι αλλαγές στην φωτεινότητα ανάμεσα στα αντικείμενα συμβαδίζουν με τις αντίστοιχες διαφορές που έχουν στην θέση τους στον χώρο.

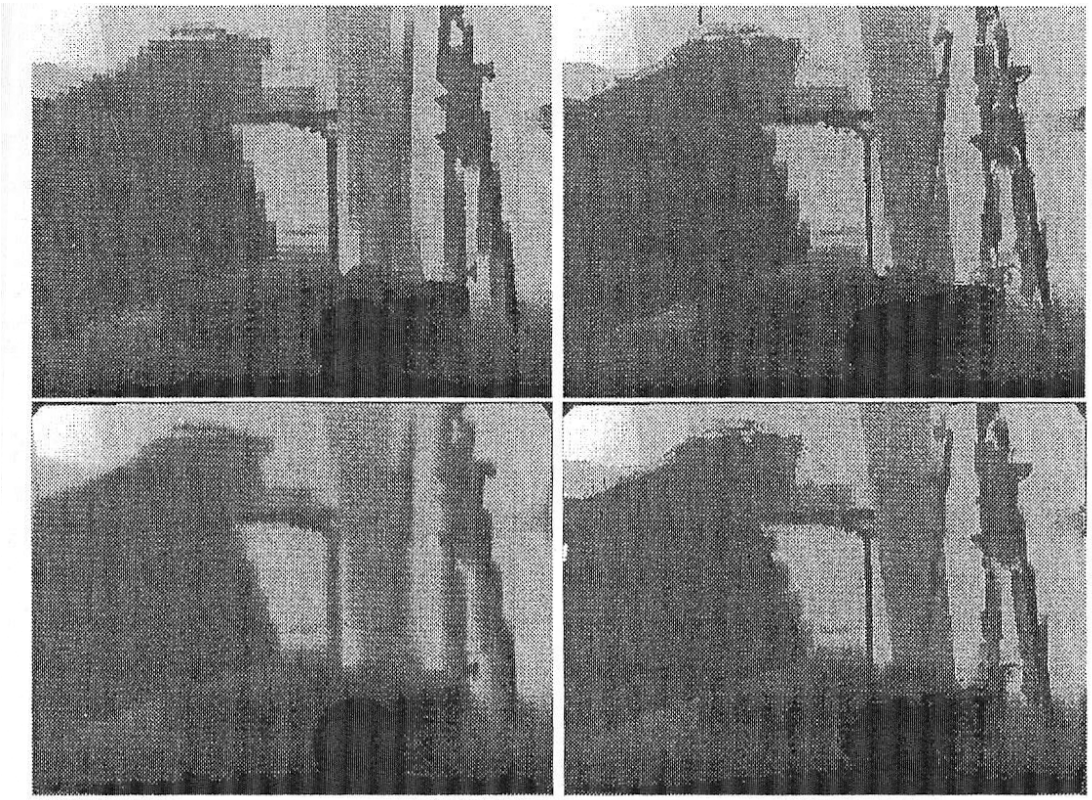
Στις περισσότερες μεθόδους που θα παρουσιάσουμε αρχικά δημιουργείται ένα μοντέλο με την χρήση του αισθητήρα laser και έπειτα προστίθενται επάνω στα αντικείμενα που ανιχνεύτηκαν τα κομμάτια της εικόνας που θεωρούμε ότι συμπίπτουν με αυτά, ώστε να δοθεί σε κάθε χαρακτηριστικό του μοντέλου ή δική του υφή, δηλαδή τα δικά του σχέδια και χρώματα.

Τα δεδομένα που έχουμε από τον αισθητήρα laser είναι ένα σύνολο μετρήσεων r_1, r_2, …, r_n. Κάθε μέτρηση περιέχει τρεις παραμέτρους, την κατακόρυφη γωνία της ακτίνας (tilt angle) θ_i, την οριζόντια γωνία (pan angle) π_i και την απόσταση r_i που αντιστοιχεί στην ευκλείδεια απόσταση του σημείου στον χώρο, άρα μπορούμε να γράψουμε ότι για κάθε μέτρηση ri = (θ_i, π_i, r_i) αντιστοιχεί ένα σημείο (x_i, y_i, z_i).

Η εικόνα αποτελείται από ένα σύνολο pixel P_j = (X_j, Y_j, C_j) όπου X_j και Y_j είναι οι συντεταγμένες ενός σημείου σε αυτή και C_j = (C^1_j, C^2_j, C^3_j) είναι τα τρία κανάλια χρώματος του pixel (πχ τα κανάλια μπορεί να είναι αυτά του συστήματος RGB – Red, Green, Blue). Χρησιμοποιώντας τριγωνομετρία μπορούμε συνδυάζοντας την απόσταση της μέτρησης r_i και την γωνία υπό την οποία αυτή πάρθηκε να προβάλουμε κάθε μέτρηση σε ένα σημείο της εικόνας, το οποίο θα χαρακτηρίζεται από την θέση του πάνω στην εικόνα, την απόστασή του και το χρώμα του, δηλαδή R_j = (X_j, Y_j, r^*_j, C_j).

Το πρόβλημα της παρεμβολής, βάση των παραπάνω, μπορεί να διατυπωθεί σαν ότι για ένα pixel P_i και για κάθε σύνολο προβολών R_i θέλουμε να υπολογίσουμε την απόσταση r^*_j με όσο το δυνατόν μεγαλύτερη ακρίβεια. Άρα αναζητούμε ένα σημείο $R^*_i = (X_i, Y_i, r_i, C_i)$.

- *Πλησιέστερη μέτρηση απόστασης*: Με την μέθοδο πλησιέστερης μέτρησης απόστασης (nearest range reading, NR) η παρεμβαλλόμενη μέτρηση r^*_j αντιστοιχίζεται στο πλησιέστερο σημείο R_i, υπολογίζοντας την κοντινότερη ευκλείδεια απόσταση σε pixel.
- *Πλησιέστερη μέτρηση απόστασης υπολογίζοντας και το χρώμα*: Αυτή η μέθοδος (nearest range reading considering color, NRC) είναι μία επέκταση της μεθόδου NR που λαμβάνει υπόψη της και την πληροφορία για το χρώμα. Δεδομένου ενός pixel P_j η παρεμβαλλόμενη μέτρηση r^*_j αντιστοιχίζεται στο πλησιέστερο σημείο R_i υπολογίζοντας όχι μόνο την ευκλείδεια απόσταση αλλά και την συνάφεια των χρωμάτων των δύο pixel.
- *Παρεμβολή πολλών γραμμών*: Με αυτή την μέθοδο (multi-lineal interpolation, MLI), με δεδομένα όλες τις προβολές R_1, R_2, …, R_n κατασκευάζεται ένα διάγραμμα Voronoi που περιέχει τις αντίστοιχες συντεταγμένες τους και τους γείτονες των αντίστοιχων pixel και έπειτα υπολογίζεται ο προσεγγιστικά κοντινότερος γείτονας.
- *Παρεμβολή πολλαπλών γραμμών υπολογίζοντας και το χρώμα*: Αυτή η μέθοδος (multi-lineal interpolation considering color, LIC), συνδυάζει την προηγούμενη αλλά λαμβάνει υπόψη της και το χρώμα στον υπολογισμό των βαρών για την κάθε περιοχή του διαγράμματος.
- *Παρεμβολή πολλών γραμμών υπολογίζοντας και το χρώμα χωρίς παραμέτρους.*



Εικόνα 9 Πάνω αριστερά: πλησιέστερη μέτρηση απόστασης. Πάνω δεξιά: *Πλησιέστερη μέτρηση απόστασης υπολογίζοντας και το χρώμα*. Κάτω αριστερά: *Παρεμβολή πολλών γραμμών*. *Κάτω δεξιά: Παρεμβολή πολλαπλών γραμμών υπολογίζοντας και το χρώμα*

Πειραματικά αποτελέσματα

Σε πειραματικό επίπεδο δοκιμάστηκαν όλες οι παραπάνω μέθοδοι με ένα ρομπότ εξοπλισμένο με μια κάμερα με αισθητήρα CCD ανάλυσης 1MP, σε συνδυασμό με έναν laser σαρωτή SICK LMS-200, τα οποία μπορούσαν να κινηθούν μαζί, περιστροφικά της βάσης-ρομπότ. Στο πείραμα το ρομπότ δεν ήταν αυτοκινούμενο, αλλά τηλεχειριζόμενο και οι λήψεις γίνονταν σε στάσεις που γίνονταν από το χειριστή του ρομπότ, σύμφωνα με τις ενδείξεις του εκάστοτε δοκιμαζόμενου αλγόριθμου.

Έγιναν τρεις μετρήσεις σε εσωτερικό χώρο του εργαστηρίου και τρεις σε εξωτερικό χώρο, όπου υπήρχε χιόνι κάνοντας τις διαβαθμίσεις των χρωμάτων αρκετά δύσκολο να μετρηθούν. Επίσης κάποιες παράμετροι αρχικοποιήθηκαν εμπειρικά, όπως ο χρωματικός θόρυβος του αισθητήρα και η πυκνότητα των pixel. Επιπλέον αυτών των μετρήσεων έγιναν και κάποιες εξομοιώσεις με δεδομένα που οι ερευνητές πήραν από τις πραγματικές μετρήσεις.

Οι μετρήσεις του εσωτερικού χώρου έδειξαν ότι οι μέθοδοι LIC, PLIC και MRF είχαν τα λιγότερα σφάλματα, με τις LIC και PLIC να έχουν τα λιγότερα, αλλά σημαντικότερα. Αυτό οφείλεται στις επιφάνειες του χώρου, που ήταν ενιαίου χρώματος χωρίς πολλές διακριτικές λεπτομέρειες.

Στις μετρήσεις εξωτερικού χώρου τα σφάλματα ήταν εμφανώς περισσότερα, κυρίως λόγω του χιονιού, που έχει ενιαίο χρώμα. Η μέθοδος MRF είχε σε αυτή την περίπτωση τα λιγότερα σφάλματα. Αυτό συμπεραίνεται εύκολα συγκρίνοντας τα αποτελέσματα της NR μεθόδου, που δεν λαμβάνει υπόψη τα χρώματα, με αυτά της MRF, που παρουσιάζουν μεγάλη ομοιότητα.

Το αποτέλεσμα είναι ότι, αν και οι μέθοδοι MRF και PLIC είχαν σχετικά περιορισμένο αριθμό σφαλμάτων, δεν υπάρχει κάποια μέθοδος που να είναι συνολικά καλύτερη από τις υπόλοιπες, αλλά η επιτυχία ή αποτυχία της μεθόδου εξαρτάται άμεσα από τις συνθήκες του περιβάλλοντος.

Αντιστοίχηση εικόνων

Η αντιστοίχηση εικόνας (image registration) είναι η διαδικασία με την οποία μετατρέπονται διαφορετικές ομάδες δεδομένων στο ίδιο σύστημα αναφοράς. Στην υπολογιστική όραση συνήθως αναφερόμαστε στην ανάλυση διαφορετικών εικόνων του ίδιου ή περίπου του ίδιου περιβάλλοντος από διαφορετική οπτική γωνία ώστε να εξάγουμε τη θέσει των ίδιων αντικειμένων σε κάθε μία από αυτές.

Συγκεκριμένα θα μελετήσουμε ειδικά την διαδικασία αντιστοίχησης όταν τα δεδομένα πέρα από τις δύο εικόνες, δηλαδή από χρωματική πληροφορία, συμπεριλαμβάνουν και τις μετρήσεις των αποστάσεων των αντικειμένων όπως αυτές δίνονται από έναν τρισδιάστατο σαρωτή laser.

Αυτή η διαδικασία μπορεί να προσφέρει πληροφορίες σχετικά με θέση των αντικειμένων στον χώρο και κυρίως σχετικά με το πώς μεταβάλλεται η θέση τους καθώς κινείται το ρομπότ, κάτι που είναι πάρα πολύ χρήσιμο για την τεχνική SLAM αφού μπορεί να βοηθήσει στον υπολογισμό της θέσης του ρομπότ.

Αξίζει να σημειωθεί ότι η πληροφορίες για την απόσταση των αντικειμένων που προσφέρει ένας σαρωτής laser δεν αρκούν για να βοηθήσουν ένα ρομπότ για να βρει το που βρίσκεται στον χώρο. Οι αποστάσεις από τα εμπόδια είναι χρήσιμες μόνο για να υπολογίζει τα αμέσως επόμενα στάδια της πορείας του αλλά είναι αρκετά δύσκολο να χρησιμοποιηθούν για την δημιουργία ενός επαρκούς χάρτη.

Ο λόγος είναι ότι τα αντικείμενα που ανιχνεύονται δεν μπορούν να αναγνωριστούν με σκοπό να υπολογιστεί το πόσο μετατοπίζονται κατά την διάρκεια της κίνησης, αφού δεν είναι δυνατή η ταυτοποίηση τους.

Σε αυτό το σημείο αναδεικνύεται η χρησιμότητα της επιπλέον οπτικής πληροφορίας που προσφέρει μία κάμερα. Αρχικά κάθε σημείο που ανιχνεύεται από τον σαρωτή laser ταυτίζεται με κάποιο ή κάποια χαρακτηριστικά της εικόνας μετά την ανάλυσή της, όπως είδαμε στην προηγούμενη ενότητα. Έπειτα, μετά την κίνηση του ρομπότ

υπάρχει και μία δεύτερη ομάδα μετρήσεων η οποία επεξεργάζεται με τον ίδιο ακριβώς τρόπο. Τέλος γίνεται μία προσπάθεια να βρεθούν τα σημεία που είναι όμοια και στις δύο εικόνες. Παρακάτω παρουσιάζεται ένας συγκεκριμένος αλγόριθμους που μπορεί να χρησιμοποιηθεί.

Τα χαρακτηριστικά κάθε εικόνας μπορούν να υπολογιστούν χρησιμοποιώντας τον αλγόριθμο SIFT που περιγράψαμε σε προηγούμενη ενότητα. Μετά σε αυτά τα χαρακτηριστικά παρεμβάλλονται όπου είναι δυνατόν οι μετρήσεις απόστασης. Έπειτα τα χαρακτηριστικά των εικόνων ταυτίζονται με σκοπό να βρεθούν τα όμοια αντικείμενα. Κατά συνέπεια καταλήγουμε να έχουμε για το ίδιο αντικείμενο, όπως αυτό προκύπτει από την σύγκριση των εικόνων, δύο μετρήσεις απόστασης: την αρχική και αυτή μετά την κίνηση.

Συγκρίνοντας το πόσο άλλαξε η απόσταση ενός αντικειμένου ανάμεσα στις δύο εικόνες μπορούμε να βρούμε πόσο απομακρύνθηκε ή πλησίασε κατά την κίνηση και με αυτόν τον τρόπο να σχηματίσουμε έναν βελτιωμένο χάρτη του περιβάλλοντος αλλά και να βρούμε πόσο μετακινήθηκε το ρομπότ μέσα σε αυτό.

Η μεγαλύτερη πρόκληση για αυτή την διαδικασία είναι η αντιστοίχηση του βάθους σε κάθε χαρακτηριστικό της εικόνας. Σε πειραματικό επίπεδο έχουν δοκιμαστεί διάφοροι αλγόριθμοι, αλλά με μεγαλύτερη επιτυχία οι εξής:

Ο αυστηρός επαναληπτικός υπολογισμός πλησιέστερου σημείου (Rigid iterative closest point), όπου ανάμεσα σε δύο διαδοχικές εικόνες – σαρώσεις της περιοχής, βρίσκεται η μετάβαση από ένα σημείο της πρώτης εικόνας στο αντίστοιχο σημείο της δεύτερης. Αυτό γίνεται με μέτρηση της απόστασης για να βρεθεί το κατάλληλο ζευγάρι. Καθώς όμως αυτή η διαδικασία είναι χρονοβόρα έχει προταθεί η χρήση της δομής Kd-δέντρων. Αξίζει να σημειωθεί ότι κατά την εύρεση του ζευγαριού συνυπολογίζεται και ο Γκαουσιανός θόρυβος (Gaussian noise), ο οποίος προκαλεί αρκετά πρακτικά προβλήματα.

Στη συνέχεια ο αυστηρός γενικευμένος υπολογισμός συνολικών ελάχιστων τετραγώνων ICP (Rigid generalized total squares ICP) αποτελεί μια προέκταση του αυστηρού επαναληπτικού υπολογισμού πλησιέστερου σημείου (ICP) όπου αντί για σημεία των δύο εικόνων χρησιμοποιούνται περιοχές σημείων. Αυτός ο αλγόριθμος όμως δεν μπορεί να απλοποιηθεί αρκετά κι έτσι παραμένει αρκετά χρονοβόρος.

Τέλος μια βελτιωμένη λύση είναι αυτή της αυστηρής περιορισμένης προέκτασης (Rigid trimmed extension), όπου υποθέτει ότι εφόσον αντιστοιχηθούν κάποια από τα σημεία, τα υπόλοιπα δε χρειάζεται να ταυτοποιηθούν, καθώς μόνο ένα ποσοστό από αυτά αρκεί για να δώσει μια σαφή αντιστοίχηση των εικόνων. Αν και υπάρχει ο κίνδυνος ένα μέρος του ποσοστού αυτού να είναι λανθασμένο, κάτι που θα οδηγούσε πιθανώς σε λάθος αντιστοίχηση των αντικειμένων, τα πειραματικά

αποτελέσματα έδειξαν ότι αν αντιστοιχηθούν το 70% των σημείων, τότε αυτά αρκούν για μια ακριβή καταγραφή της μετάβασης.

Χαρτογράφηση και προσδιορισμός θέσης στον χώρο με αισθητήρες όρασης και απόστασης

Προηγουμένως αναφέραμε ότι ο συνδυασμός των οπτικών δεδομένων από μία κάμερα και των μετρήσεων απόστασης από έναν σαρωτή laser μπορεί να χρησιμοποιηθεί για την χαρτογράφηση του χώρου μέσα στον οποίο κινείται ένα ρομπότ και τον προσδιορισμό της θέσης του σε αυτόν, αντιμετωπίζοντας τα προβλήματα που παρουσιάζονται από το SLAM.

Οι τυπικοί αλγόριθμοι SLAM μπορούν να χωριστούν σε τρεις κατηγορίες ανάλογα με τον τρόπο που αναπαριστούν τον χάρτη.

- Μέθοδοι που παρακολουθούν κάποια σημεία ενδιαφέροντος
- Μέθοδοι που βασίζονται στις σχετικές θέσεις των αντικειμένων μεταξύ τους (συνήθως βασισμένοι σε γράφους).
- Μέθοδοι που χωρίζουν τον χάρτη σε ένα πλέγμα.

Σχεδόν όλες οι υλοποιήσεις του SLAM με την χρήση οπτικής πληροφορίας αποφεύγουν τον υπολογισμό των σχετικών θέσεων των αντικειμένων. Ο λόγος είναι ότι για τον εντοπισμό αυτών των σημείων πρέπει να ξεπεραστεί το εμπόδιο του υψηλού θορύβου που υπάρχει σε κάθε εικόνα.

Για την επιτυχή ανίχνευση των σημείων χρειάζεται η επεξεργασία αρκετών εικόνων για να μπορούν να ανιχνευτούν τα όμοια σημεία που υπάρχουν στην εικόνα και ο τρόπος με τον οποίο αυτά μετακινούνται, ενώ οι μεταξύ τους αποστάσεις είναι και πάλι αρκετά δύσκολο να υπολογιστούν.

Αντίθετα στα συστήματα που βασίζονται στους σαρωτές laser είναι περισσότερο κοινές οι μέθοδοι που επεξεργάζονται τις αποστάσεις μεταξύ των μοντέλων εξαιτίας της υψηλής ακρίβειας που προσφέρουν στις μετρήσεις των αποστάσεών. Το μικρότερο ποσοστό σφάλματος στις μετρήσεις καθιστά περιττή την εξαγωγή και την παρακολούθηση τον χαρακτηριστικών από τις εικόνες. Επιπλέον πολλοί αλγόριθμοι αντιστοίχησης εικόνων δεν λαμβάνουν αυτά τα χαρακτηριστικά υπόψη τους, όπως ο αλγόριθμος ICP.

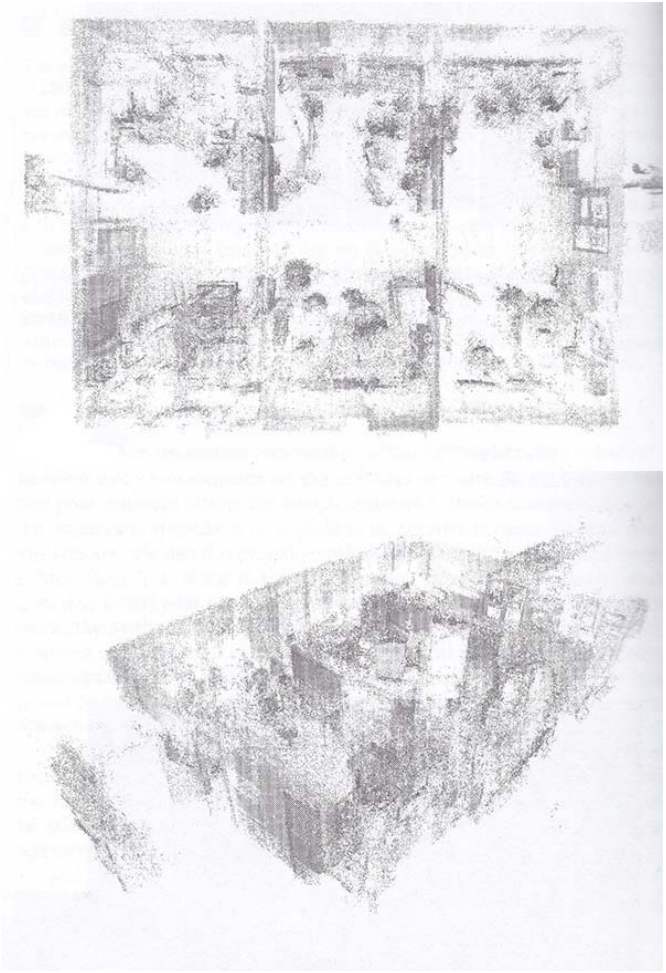
Προσδιορισμός θέσης στον χώρο με χρήση κάμερας και τρισδιάστατου σαρωτή

Ο σφαιρικός προσδιορισμός θέσης (global localization) προσπαθεί να προσδιορίσει την θέση του ρομπότ σε έναν χάρτη που ήδη γνωρίζει ή έχει δημιουργήσει χωρίς να γνωρίζει την αρχική του θέση. Καθώς μας ενδιαφέρει ο προσδιορισμός της θέσης στον χώρο και όχι στο επίπεδο ερευνούμε το θέμα από γεωμετρική και όχι από

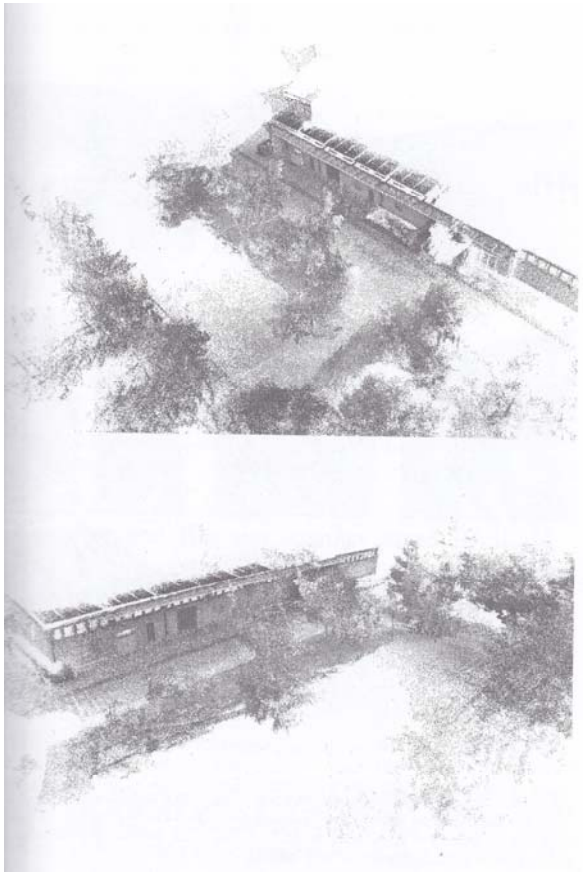
τοπολογική σκοπιά, δηλαδή το που βρίσκεται σε σχέση με τα αντικείμενα στον χώρο.

Η μεγαλύτερη διαφορά ανάμεσα στον προσδιορισμό θέσης στον χώρο και στον προσδιορισμό στο επίπεδο είναι ο αρκετά μεγαλύτερος αριθμός των πιθανών καταστάσεων. Στο επίπεδο αρκεί να προσδιορίσει την θέση βάση δύο συντεταγμένων ενώ στον χώρο χρειάζεται ο υπολογισμός τριών.

Το μεγαλύτερο πλεονέκτημα από την χρήση ενός τρισδιάστατου σαρωτή είναι ότι αυτός συνήθως παράγει ένα μοντέλο αρκετά υψηλότερης ανάλυσης από αυτό που θα παρήγε για παράδειγμα μία στερεοσκοπική κάμερα. Έτσι οι ομοιότητες που προκύπτουν από δύο διαδοχικά μοντέλα (που παράγονται από τον σαρωτή) μπορούν να χρησιμοποιηθούν για να βρουν την σχετική μετατόπιση ανάμεσα σε δύο διαφορετικές θέσεις του ρομπότ στον χώρο. Συμπληρώνοντας την αντιστοίχηση των αντικειμένων βάση των μοντέλων του σαρωτή, τα οπτικά χαρακτηριστικά μπορούν να χρησιμοποιηθούν για να βρεθεί ποια από τις ήδη γνωστές θέσεις στον χάρτη ανταποκρίνεται καλύτερα στην τρέχουσα.



Εικόνα 10 οπτικοποιημένο αποτέλεσμα από τη φωτογράφιση και σάρωση εσωτερικού χώρου



Εικόνα 22 οπτικοποιημένο αποτέλεσμα από τη φωτογράφιση και σάρωση εξωτερικού χώρου

Συμπέρασμα

Ένα αυτόνομο ρομπότ για να κινηθεί επιτυχώς στο χώρο και να μπορέσει να σχεδιάσει την πορεία του, ουσιαστικά θα πρέπει να λύσει το πρόβλημα γνωστό ως SLAM. SLAM σημαίνει Simultaneous Localization and Mapping, δηλαδή Ταυτόχρονη αυτοτοποθέτηση και Χαρτογράφηση. Το πρόβλημα αυτό αναλύεται στα εξής υποπροβλήματα.

Καταρχάς της κίνησης του ρομπότ στο χώρο, χωρίς αυτό να συγκρούεται με τα διάφορα εμπόδια που μπορεί να υπάρχουν στο χώρο. Αυτό επιτυγχάνεται μέσω κάποιων αλγορίθμων με μαθηματικά μοντέλα εύρεσης καλύτερης διαδρομής, οι οποίοι πλέον βρίσκονται σε αρκετά καλό επίπεδο απόδοσης.

Στη συνέχεια το ρομπότ θα πρέπει να είναι σε θέση να χαρτογραφεί τον περιβάλλοντα χώρο μέσα στον οποίο κινείται. Βάση του μοντέλου που θα κατασκευάσει θα πρέπει να μπορεί να προσδιορίσει την πορεία και τον προορισμό του.

Ο τρόπος με τον οποίο το ρομπότ κατασκευάζει το μοντέλο του χώρου, εξαρτάται από τους αισθητήρες με τους οποίους είναι εξοπλισμένο. Παραπάνω μελετήσαμε την περίπτωση που υπάρχει μια κάμερα και ένας τρισδιάστατος σαρωτής laser για την εύρεση αποστάσεων.

Με τη χρήση κάμερας έχουν αναπτυχθεί αρκετοί αλγόριθμοι ανάλυσης και κατάτμησης μιας εικόνας σε αντικείμενα από τα οποία αποτελείται, καθώς πρόκειται για ένα πρόβλημα που έχει αναλυθεί αρκετά και αρκετές τεχνικές χρησιμοποιούνται και βελτιώνονται εδώ και αρκετά χρονιά, όπως η εύρεση ακμών και η ανάλυση ιστογραμμάτων. Αν και τα αποτελέσματα δεν είναι πάντα ακριβή και είναι ευαίσθητα σε θόρυβο ή σε απότομες αλλαγές της φωτεινότητας γενικά παράγουν ικανοποιητικά αποτελέσματα.

Οι τρισδιάστατοι σαρωτές laser είναι μία σχετικά πρόσφατη εξέλιξη και τα μέχρι στιγμής αποτελέσματα είναι αρκετά ενθαρρυντικά. Ο συνδυασμός των δύο μπορεί να βγάλει ακριβέστερα αποτελέσματα και είναι πρακτικά υλοποιήσιμο, αν και ακόμη η διαδικασία είναι σε πειραματικό στάδιο.

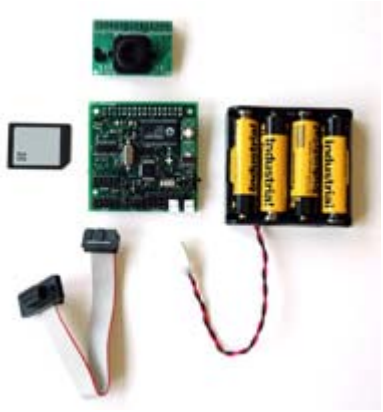
Εφαρμογές ανίχνευσης Χρώματος, Κίνησης και Ανίχνευσης προσώπου με το chipset Cmucam3.

Εικόνα 23 Η πλατφόρμα CMUcam3

CMUcam3: Ενσωματομένη πλατφόρμα έγχρώμης όρασης ανοικτού κώδικα (Open Source Programmable Embedded Color Vision Platform).

Επισκόπηση

Ο στόχος του έργου CMUcam είναι να παρέχει απλές δυνατότητες όρασης σε μικρά ενσωματωμένα συστήματα, με τη μορφή ενός ευφυούς αισθητήρα. Η CMUcam3 επεκτείνει επάνω σε αυτήν την ιδέα, παρέχοντας ένα ευέλικτο και εύκολο στη χρήση ανοικτό περιβάλλον ανάπτυξης κώδικα το οποίο συμπληρώνεται από μια χαμηλού κόστους πλατφόρμα υλικού. Η CMUcam3 είναι ένας πλήρως προγραμματιζόμενος ενσωματωμένος αισθητήρας όρασης βασιζόμενος στον επεξεργαστή ARM7TDMI. Ο κύριος επεξεργαστής είναι ο NXP LPC2106 που συνδέεται με μια μονάδα αισθητήρα-κάμερας πάνω σε CMOS της OmniVision . Προσαρμοσμένα στις ανάγκες μας προγράμματα κώδικα C μπορούν να αναπτυχθούν για την CMUcam3 χρησιμοποιόντας το σετ εργαλείων της GNUtoolchain μαζί με ένα σύνολο από βιβλιοθήκες ανοικτού κώδικα και των ήδη έτοιμων παραδειγμάτων που παρέχονται.Επίσης τα εκτελούμενα προγράμματα μας μπορούν να να φορτώθούν στην μνήμη της πλατφόρμας απευθείας μεσω της σειριακή θύρας χωρίς την απαίτηση άλλης εξωτερικής συσκευής.



Εικόνα 24 Τα μέρη της CMUcam3 : αριστερά MMC/SD Flash card, επάνω μονάδα αισθητήρα-κάμερας πάνω σε CMOS της OmniVision ,μέση Ο κύριος επεξεργαστής είναι ο NXP LPC2106 , κάτω Σειριακό καλώδιο , δεξιά Τροφοδοσία

Χαρακτηριστηκά πλατφόρμας CMUcam3.

 Πλήρως ανοικτού κώδικα και την προγραμματιζόμενη με τη χρήση του GCC.

 Ανάλυση CIF (352x288), αισθητήρα χρωματικού πλαισίου RGB

 Ανοικτό περιβάλλον ανάπτυξης κώδικα για Windows και Linux

 Θύρα μνήμης MMC / SD flash με την υποστήριξη του οδηγού FAT16

 Τέσσερις θύρες χειρισμού σερβο

 Επεξεργασία εικόνας ταχύτητας 26 καρέ ανά δευτερόλεπτο

 Ελαφρύ διερμηνέα γλώσσας προγραμματισμού Lua για ταχεία πρωτοτυποποίηση

 Λογισμικό συμπίεσης εικόνων JPEG

 Βασική βιβλιοθήκη χειρισμού εικόνας :

- Ψαλίδισμα αυθαίρετων στοιχείων εικόνας
- Downsampling εικόνας
- Κατώτατο όριο και λειτουργίες συνέλιξης
- RGB, YCrCb και HSV χρωματικοί χώροι

- Εξομοιωτής λειτουργιών της πλατφόρμας CMUcam2
  - Ορισμός χρωματικών σταγόνων από το χρήστη
  - Πλαίσιο διάκρισης των διαφορών μεταξύ εικόνων
  - Συλλογή δεδομένων διασποράς και μέση τιμής
  - Συλλογή ακατέργαστων εικόνων από τη σειριακή
  - Δημιουργία ιστογράμματος εικοστοιχείων
- B / W αναλογική έξοδο εικόνας (PAL ή NTSC)
- FIFO buffer εικόνας για πολλαπλές περάσει hi-res επεξεργασία εικόνας
- Ασύρματη διασύνδεση δικτύων Mote (802.15.4)
- Εικονικό περιβάλλον cmucam για τη δημιουργία πρωτοτύπων για το PC
- CMUcam3-Frame-Grabber για την προβολή εικόνων στον υπολογιστή

Ανίχνευση κίνησης και χρώματος με την CMUcam3.

Για τις 2 εφαρμογές θα χρησιμοποιηθούν

- Ο εξομοιωτής CMUcam2
- Το πρόγραμμα java CMUcam2GUI

Βασικές έννοιες και λειτουργίες για την κατανόηση των εφαρμογών ανίχνευσης και κίνησης.

Λόγοι χρησιμοποίησης του εξομοιωτή CMUcam2 για τις εφαρμογές

Μία από τις κύριες χρήσεις του CMUcam2 είναι η ανίχνευση και παρακολούθηση χρώματος. Η βέλτιστη απόδοση μπορεί να επιτευχθεί όταν υπάρχουν πολύ αντικρουόμενα και έντονα χρώματα. Για παράδειγμα, μπορεί να παρακολουθείτε εύκολα μία κόκκινη μπάλα σε λευκό φόντο, αλλά θα ήταν δύσκολο να γίνει διαφοροποίηση μεταξύ διαφόρων αποχρώσεων του καστανού σε μεταβαλλόμενο φως. Η ανίχνευση πολύχρωμων αντικειμένων μπορεί να χρησιμοποιηθεί για να εντοπίστούν σημεία, να ακολουθήσουμε γραμμές, ή να κυνηγήσουμε ένα κινούμενο φάρο.Χρησιμοποιώντας στατιστικές χρώματων, είναι δυνατόν να παρακολουθήσουμε μια σκηνή, να ανιχνεύσουμε ένα συγκεκριμένο χρώμα ή να κάνουμε πρωτόγονες ανίχνευσης κίνησης. Αν η κάμερα ανιχνεύσει μια δραστική

αλλαγή χρώματος , τότε οι πιθανότητες είναι ότι κάτι στο σκηνικό άλλαξε.Χρησιμοποιώντας τη "λειτουργία γραμμής," η CMUcam2 μπορεί να λειτουργήσει ως ένας εύκολος τρόπος για να πάρουμε χαμηλής ανάλυση δυαδικές εικόνες πολύχρωμων αντικειμένων. Αυτό μπορεί να χρησιμοποιηθεί για να κάνουμε πιο εξελιγμένη παρακολούθηση γραμμής που θα περιλαμβάνει τον εντοπισμό κλάδων, ή ακόμα και αναγνώριση απλών σχημάτων. Αυτές οι πιο προηγμένες εργασίες θα απαιτούσαν προσαρμοσμένους αλγόριθμους οι οποίοι θα εκτελούσαν προ-επεξεργασία των δυαδικών εικόνων που αποστέλλονται από την CMUcam2.

Μετατροπή εικόνας σε μία σειρά από pixel

Ο αισθητήρας εικόνας CMOS είναι η καρδιά της συγκέντρωσης των πληροφοριών. Πρόκειται για ένα τσιπ πυριτίου που περιέχει ένα πλέγμα από τετράγωνα, το καθένα από τα οποία είναι ευαίσθητο σε διαφορετικά χρώματα του φωτός. Αφού το φως περάσει μέσα από το φακό, διεγείρει τα κουτιά, δημιουργώντας μια διαφορετική τάση ανάλογη με την ποσότητα του φωτός. Αυτή η τάση μετατρέπεται σε μια ενιαία αριθμητική τιμή για κάθε κανάλι. Στην περίπτωση του CMUcam2, αυτή η τιμή είναι της τάξεως των 16 έως 240. Υπάρχει ένα κόκκινο κανάλι, ένα μπλε κανάλι και δύο πράσινα κανάλια, καθένα από τα οποία μόνο είναι ευαίσθητο σε αυτό το ειδικό χρώμα του φωτός. Το επιπλέον πράσινο κανάλι βοηθά να συμπληρώσουμε την πλέγμα, ώστε κάθε pixel μπορεί να είναι ομοιόμορφα κατανεμημένο σε όλο την αισθητήρα. Το επιπλέον πράσινο επίσης προσεγγίζει το ανθρώπινο μάτι το οποίο είναι πιο ευαίσθητο στο πράσινο χρώμα. Για σκοπούς απλοποίησης, η CMUcam2 αγνοεί το δεύτερο πράσινο κανάλι.

Χαρτογράφηση των pixel

Μερικές φορές είναι χρήσιμο να κατανοήσουμε με μεγαλύτερη ακρίβεια τον τρόπο με τον οποίο τα δεδομένα από τον αισθητήρα κάμερας μεταφράζονται σε pixel. Εδώ εξηγούμε για το OV6620 (CMUcam2) αισθητήρα, αλλά η ίδια βασική διάταξη ισχύει και για τον αισθητήρα OV7620(CMUcam3).

Ο αισθητήρας έχει 356 στήλες και 292 γραμμές ευαίσθητων κυττάρων φωτός που διοργανώνονται σε ένα πλέγμα. Κάθε τοποθεσία μπορεί να ανιχνεύσει ένα μόνο χρώμα: κόκκινο, πράσινο ή μπλε. Εδώ είναι η διάταξη αισθητήρα του οι τέσσερις πρώτες σειρές:

Row 1: B(1,1) G(1,2) B(1,3) G(1,4) B(1,5) G(1,6) ...B(1,355) G(1,356)
Row 2: G(2,1) R(2,2) G(2,3) R(2,4) G(2,5) R(2,6) ...G(2,355) R(2,356)
Row 3: B(3,1) G(3,2) B(3,3) G(3,4) B(3,5) G(3,6) ...B(3,355) G(3,356)
Row 4: G(4,1) R(4,2) G(4,3) R(4,4) G(4,5) R(4,6) ...G(4,355) R(4,356)

Η μονάδα της κάμερας λαμβάνει τα δεδομένα από δύο σειρές του αισθητήρα κάθε φορά για να δημιουργήσει την γραμμή κάθε γραμμή εξόδου της κάμερας:

Row 1: B(1,1) G(2,1) R(2,2) G(1,2) B(1,3) G(2,3) R(2,4) G(1,4) ...
Row 2: B(3,1) G(2,1) R(2,2) G(3,2) B(3,3) G(2,3) R(2,4) G(3,4) ...

Η μονάδα της κάμερας CMUcam2 παίρνει τα παραπάνω δεδομένα και μας δίνει τα ακόλουθα pixel (λαμβάνουμε υπόψιν ότι για λόγους απλοποίησης αγνοεί το ένα κανάλι πράσινου χρώματος) :

Row 1: [R(2,2):G(1,2):B(1,1)] [R(2,4):G(1,4):B(1,3)] ...
Row 2: [R(2,2):G(3,2):B(3,1)] [R(2,4):G(3,4):B(3,3)] ...

Η ανίχνευση ενός χρώματος με τη CMUcam2

Ανίχνευση χρώματος είναι η δυνατότητα να πάρουμε μια εικόνα, να απομονώσουμε ένα ιδιαίτερο χρώμα και να εξάγουμε πληροφορίες για την τοποθεσία μιας περιοχής της εικόνας που περιέχει ακριβώς αυτό το χρώμα. Για παράδειγμα, ας υποθέσουμε ότι μας δίνεται μια φωτογραφία που περιέχει μία κόκκινη μπάλα που κάθεται σε ένα χωματόδρομο. Αν κάποιος έπρεπε να μας ζητήσει να επιστήσουμε ένα πλαίσιο γύρω από οτιδήποτε ήταν κόκκινο χρώμα στην εικόνα, θα ήταν αρκετά εύκολο να σχεδιάσουμε ένα ορθογώνιο γύρω από την μπάλα. Αυτή είναι η βασική ιδέα πίσω από την ανίχνευση χρώματος. Εμείς δεν χρειάζεται να γνωρίζουμε ότι το αντικείμενο ήταν μια μπάλα. Το μόνο που χρειάζεται γνωρίζουμε ποιο χρώμα είναι το κόκκινο, για να απομονώσουμε το αντικείμενο στην εικόνα.Παρακάτω εξηγούμε το πώς η CMUcam2 χρησιμοποιεί τις πληροφορίες σε μια εικόνα της κάμερας για να εκτελέσει την ανίχνευση χρώματος. Για να καθορίσουμε το χρώμα, θα πρέπει να καθιερωθεί ένα ελάχιστο και το μέγιστο επιτρεπόμενο όριο-αξία για κάθε ένα από αυτά τα τρία κανάλια χρώματος. Κάθε μοναδικό χρώμα εκπροσωπείται από μία κόκκινη, πράσινη και μπλε τιμή που δηλώνει πόσο του κάθε καναλιού αναμειγνύεται σε αυτο το τελικό χρώμα. Το δύσκολο μέρος για τον καθορισμό ενός χρώματος είναι ότι θα πρέπει να προσδιορίσει μια σειρά από επιτρεπόμενες τιμές για τα τρία κανάλια χρώματος. Δεδομένου ότι το φως δεν είναι απόλυτα ομοιόμορφο και το χρώμα ενός αντικειμένου που δεν είναι απόλυτα ενιαίο, θα πρέπει να συμπεριλάβουμε αυτές τις διακυμάνσεις. Ωστόσο, δεν θέλουμε να χαλαρώσουμε τα όρια πάρα πολύ,διότι πολλά ανεπιθύμητα χρώματα θα γίνονται δεκτά.Δεδομένου ότι, στην περίπτωση της CMUcam2, κάθε κανάλι χρώματος

μετατρέπεται σε αριθμό μεταξύ 16 και 240, μπορούμε να δεσμεύουμαι κάθε κανάλι με δύο αριθμούς, ένα ανώτατο και κατώτατο όριο. Αν χρειαζόμαστε λοιπόν δύο όρια για κάθε ένα από τα τρία κανάλια, σημαίνει ότι έξι τιμές μπορούν να χρησιμοποιηθούν για να περιορίσουν το σύνολο του χρωματικού χώρου που θέλουμε να ανιχνεύσουμε. Εάν φανταστούμε τα χρώματα να εκπροσωπούνται από ένα κύβο όπου η κάθε πλευρά έχει ένα διαφορετικό κανάλι χρώματος (κόκκινο, πράσινο και μπλε), στη συνέχεια, οι έξι τιμές που χρησιμοποιήσαμε για να επιλέξουμε το χρώμα που θέλαμε θα σχεδίαζαν ένα τρισδιάστατο πλαίσιο (πλαίσιο οριοθέτησης – αναπήδησης ή bounding box) το οποίο ορίζει το επιθυμητό σύνολο χρωμάτων μέσα στον κύβο. Μόλις έχουμε την αναπήδηση για το χρώμα που θέλουμε να ανιχνεύσουμε, η CMUcam2 λαμβάνει αυτά τα όρια και επεξεργάζεται την εικόνα.Υπάρχουν πολλοί τρόποι για να ανιχνεύσουμε τα χρώματα σε μια εικόνα που μπορεί να είναι αρκετά περίπλοκοι. Η CMUcam2 χρησιμοποιεί ένα απλό αλγόριθμο ενός περάσματος που επεξεργάζεται κάθε νέο καρέ εικόνας από την κάμερα ανεξάρτητα.Ξεκινά στο πάνω αριστερό μέρος της εικόνας και διαδοχικά εξετάζει κάθε pixel γραμμή προς γραμμή. Αν το pixel επιθεωρεί πέφτει μέσα στο φάσμα των χρωμάτων που ο χρήστης έχει προσδιορίσει, σηματοδοτείται ως ανιχνευμένο. Εξετάζει επίσης τη θέση του τρέχοντος ανιχνευμένου pixel για να δει αν είναι περισ ωβότερο στην κορυφή , περισσότερο κάτω , περισσότερο αριστερά ή περισσότερο δεξιά σε σχέση με τη θέση όλων των εντοπισμένων pixel που βρέθηκαν μέχρι τώρα στην εικόνα. Αν διαπιστώσει ότι το pixel είναι έξω από το δεδομένο εως τώρα πλαίσιο οριοθέτησης της ανιχνευμένης περιοχής, αυξάνεται το πλαίσιο οριοθέτησης ώστε να περιέχει το νέο αυτό pixel. Επειδή η τοποθεσία έστω και ενός pixel μπορεί να αλλάξει το πλαίσιο οριοθέτησης, το bounding box μπορεί να διακυμαίνεται ορισμένες φορές αρκετά από το ένα καρέ στο άλλο.Υπάρχει η δυνατότητα φιλτραρίσματος θορύβου, που μπορεί να χρησιμοποιηθεί για να μειώσει ένα ποσοστό αυτών των διακυμάνσεων. Το μόνο άλλο σημαντικό κομμάτι των πληροφοριών που αποθηκεύονται είναι το σύνολο των οριζοντίων και κάθετων συντεταγμένων των ανιχνευμένων pixels. Στο τέλος η εικόνα που μπορείτε ναλαμβάνει το οριζόντιο άθροισμα και το κάθετο άθροισμα των εντοπισμένων pixels και διαιρούμε καθένα με το συνολικό αριθμό των εντοπισμένων pixels, θα πάρετε μια τιμή που δείχνει που βρίσκεται το κέντρο του εντοπισμένου αντικειμένου. Επειδή κάθε εντοπισμένο pixel συνεισφέρει μόνο ένα μικρό μέρος στα τελικά οριζόντια και κάθετα ποσά, το κέντρο(συχνά ονομάζεται το κέντρο βάρους - centroid) των εντοπισμένων pixels είναι συνήθως μία πολύ πιο σταθερή μέτρηση από το πλαίσιο οριοθέτησης (bounding box). Μόλις όλα τα pixel της εικόνας έχουν ελεγχθεί, ο συνολικός αριθμός των εντοπισμένων pixels μπορεί επίσης να χρησιμοποιηθεί σε συνδυασμό με την περιοχή του πλαισίου οριοθέτησης για τον υπολογισμό της ορθότητας του ανιχνευμένου αντικειμένου.

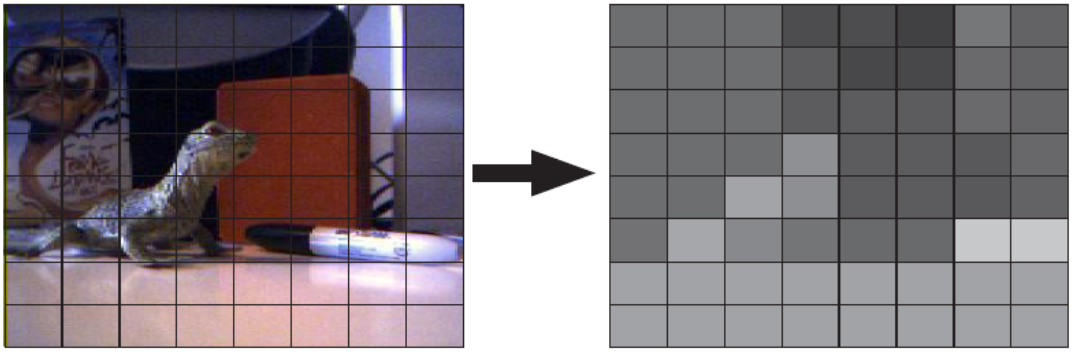
Εικόνα 25 Ανίχνευση κόκκινου χρώματος σε μία εικόνα του αισθητήρα.

Φιλτράρισμα του θορύβου σε μια εικόνα

Το φιλτράρισμα θορύβου μας επιτρέπει να αυξήσουμε τις τιμές των χρωματικών ανοχών ενός χρώματος χωρίς να προκαλέσουμε ανίχνευση τυχαίων χρωματικών ανωμαλιών στα pixels της εικόνας. Η ιδέα πίσω από το φιλτράρισμα θορυβου είναι ότι εμείς θέλουμε να θεωρούμε ένα pixel ότι είναι του χρώματος που ανιχνεύουμε μόνο εάν αποτελεί μέρος της ομάδας των pixels που βρίσκονται εντός των ορίων ανίχνευσης χρώματος. Και πάλι στη CMUcam2 υλοποιούμε αυτό με έναν τρόπο που απαιτεί μόνο ένα πέρασμα πάνω από την εικόνα. Κατά την επεξεργασία των pixel σε μια εικόνα η CMUcam2 διατηρεί ένα μετρητή που κρατά τον αριθμό των διαδοχικών pixels στην τρέχουσα γραμμή που ήταν πριν από το τρέχον pixel και είναι εντός των ορίων του χρώματος που ανιχνεύουμε. Αν η τιμή είναι πάνω από το τιμή του φίλτρου θορύβου, τότε το τρέχον εικονοστοιχείο χαρακτηρίζεται ως ανιχνευμένο pixel.Π.χ έχουμε θέσει την τιμή του φίλτρου σε 2, αυτό σημαίνει ότι χρειαζόμαστε 2 διαδοχικά pixel του χρώματος που ζητάμε να προηγούνται του τρέχοντος ανιχνευμένου pixel.

● Υλοποίηση διαφοροποίησης των καρέ (Frame Differencing)

Η διάκριση των διαφορών των καρέ είναι μια μέθοδος προσδιορισμού των αλλαγών σε μια σειρά από εικόνες.Εάν έχουμε πολλαπλές εικόνες σε διαφορετικό χρόνο από τα ίδια ή παρόμοια σημεία λήψης, είναι δυνατή η σύγκρισή τους, προκειμένου να απομονώσουμε τα αντικείμενα που μπορεί να έχουν μετακινηθεί.Χρησιμοποιώντας το πλαίσιο λειτουργιών του CMUcam2 για διάκριση των διαφορών είναι ένας καλός τρόπος για να ανιχνεύσουμε και να παρακολουθείσουμε κίνηση σε μια σκηνή. Αντί για την αποθήκευση μιας ολόκληρης εικόνας, η CMUcam2 αποθηκεύει ένα αφαίρεστικό μοντέλο της εικόνας. Χρησιμοποιώντας μια παρόμοια διαδικασία με την ανίχνευση χρώματος, η CMUcam2 θα δημιουργήσει ή θα συγκρίνει την εικόνα γραμμή προς γραμμή καθώς λαμβάνει τα δεδομένα. Η CMUcam2 αντιπροσωπεύει εσωτερικά μια εικόνα αναφοράς σαν ένα πίνακα 8x8 bytes. Κάθε στοιχείο του εν λόγω πίνακα αποθηκεύει το μέσο όρο μιας αντίστοιχης περιοχής σχετικά με την κύρια εικόνα της κάμερας. Η προεπιλεγμένη ρύθμιση χρησιμοποιεί το πράσινο ή αλλιώς κανάλι έντασης, αλλά αυτό μπορεί να αλλάξει για τις περιπτώσεις όπου ένα κανάλι με σαφήνεια προκαλεί μεγαλύτερη διακύμανση από τα άλλα. Όταν μια νέα εικόνα διαβάζεται,επίσης μετατρέπεται σε ένα πίνακα από 8x8 bytes. Για να αναζητήσουμε μια αλλαγή, κάθε μπλοκ στο 8x8 πλέγμα αφαιρείται από το αντίστοιχο μπλοκ στην εικόνα αναφοράς. Εάν η τιμή είναι μεγαλύτερη από ένα συγκεκριμένο όριο, σηματοδοτήται μια αλλαγή. Το υπόλοιπο των δεδομένων,όπως η μέση μάζα, υπολογίζονται σε ένα σχεδόν πανομοιότυπο τρόπο, με τον τρόπο ανίχνευσης χρώματος.



Εικόνα 26 Δέιχνει τη μετατροπή μιας εικόνας στο εσωτερικό της κάμερας

Ανίχνευση προσώπου με την CMUcam3.

Για την εφαρμογή θα χρησιμοποιηθούν

- Η βιβλιοθήκη cc3 της CMUcam3
- Το πρόγραμμα C Face Detector: viola-jones_lpc2106-cmucam3.hex
- Η έξοδος θα διαβάζεται μέσω του προγράμματος επικοινωνίας HyperTerminal

→ Γιά λόγους συντομίας όπου αναφέρεται η λέξη ορθογώνιο εννοείται ορθογώνιο παραλληλόγραμμο.

• Επισκόπηση

Η εφαρμογή ανίχνευσης προσώπου που πραγματοποιείται με την CMUcam3 βασίζεται στην πολύ γνωστή εργασία "Ισχυρή ανίχνευση προσώπου σε πραγματικό χρόνο" των Viola και Jones, αποδεκτή στο International Journal of Computer Vision 2004 . Παλαιότερη έκδοση του ίδιου εγγράφου ήταν είχε υποβληθεί στην ICCV 2001.Η εργασία εισάγει μια νέα τεχνική για την ανίχνευση προσώπων σε πραγματικό χρόνο και με πολύ υψηλό ποσοστό ανίχνευσης. Πρόκειται ουσιαστικά για μία προσέγγιση που βασίζεται σε χαρακτηριστικά, κατά την οποία ένας ταξινομητής έχει εκπαιδευτεί για την ταξινόμηση ορθογώνιων χαρακτηριστικών τύπου Haar , που επιλέγονται από τον αλγόριθμο AdaBoost. Η εικόνα της δοκιμής σαρώνεται υπό διαφορετικές κλίμακες και θέσεις χρησιμοποιώντας ένα ορθογώνιο παράθυρο, και οι περιοχές που διέρχονται από τον ταταξινομητή δηλώνονται ως πρόσωπα. Μια από τις σημαντικότερες συνεισφορές της μεθόδου αυτής είναι ο ταχύτατος υπολογισμός αυτών των χαρακτηριστικών που χρησιμοποιούν την έννοια της αναπόσπαστου εικονιδίου (integral image), η οποία επιτρέπει την ανίχνευση σε πραγματικό χρόνο. Επιπλέον, αντί να μάθουμε ένα ενιαίο ταξινομητή και να υπολογίζουμε όλα τα χαρακτηριστικά για όλες τα σαρωμένα παράθυρα στην εικόνα, μαθαίνεται ένας αριθμός ταξινομητών , οι οποίοι τίθενται από κοινού σε μια σειρά για να σχηματίσουν μία αλυσίδα ταξινομητών (classifier cascade) . Οι ταξινομητές στις αρχές της αλυσίδας είναι απλούστεροι και αποτελούνται από μικρότερο αριθμό χαρακτηριστικών.Ωστόσο, καθώς συνεχίζουμε στην αλυσίδα, οι ταξινομητές γίνονται πιο σύνθετοι. Μια περιοχή αναφέρεται ως ανίχνευση μόνο αν περάσει όλα τα στάδια των ταξινομητών στην αλυσίδα. Αν απορριφθεί σε οποιοδήποτε στάδιο, αποβάλλεται και δεν χρειάζεται περαιτέρω επεξεργασία. Με αυτό τον τρόπο, τα ευκολότερα σημεία στην εικόνα για τα οποία η «αλυσίδα των ταξινομητών" είναι σίγουρη ότι δεν ανήκουν σε πρόσωπο, απορρίπτονται από πολύ νωρίς, ενώ οι δύσκολες περιοχές επεξεργάζονται από πιο σύνθετους ταξινομητές. Αυτό επιταχύνει σημαντικά τη διαδικασία ανίχνευσης χωρίς να θέτει σε κίνδυνο την ακρίβεια και παρέχει υψηλό ποσοστό ανίχνευσης. Το σύστημα αυτό παρέχει συνολική απόδοση συγκρίσιμη με τα υφιστάμενα βέλτιστα συστήματα ανίχνευσης

προσώπου αλλά σε σύγκριση μεγεθών ταχύτερα από οποιοδήποτε από τα συστήματα αυτά. Σε ένα συμβατικό υπολογιστή γραφείου, μπορεί να εντοπίσει τα πρόσωπα στα 15 καρέ ανά δευτερόλεπτο.

● Τα χαρακτηριστικά Haar

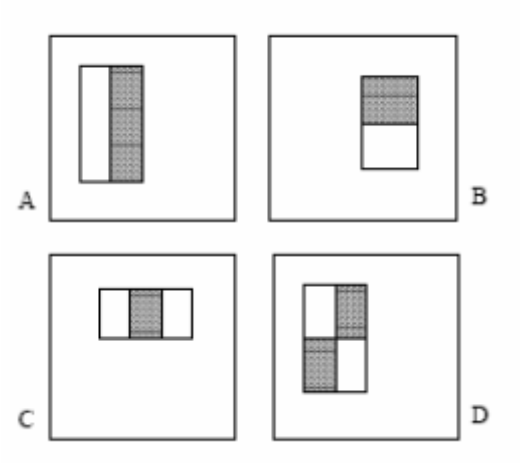
Τα Haar χαρακτηριστικά είναι χαρακτηριστικά γνωρίσματα ψηφιακής εικόνας που χρησιμοποιούνται για την αναγνώριση αντικειμένων. Οφείλουν το όνομά τους στην ομοιότητά τους με τα Haar wavelets και χρησιμοποιήθηκαν στον πρώτο ανιχνευτή πρόσωπο πραγματικού χρόνου.Ιστορικά, χρησιμοποιόντας μόνο εντάσεις εικόνας (π.χ, τις τιμές RGB σε καθένα από pixel της εικόνας) ο υπολογισμός των χαρακτηριστικών κόστιζε πανάκριβά σε υπολογιστική ισχύ. Μέχρι που μία δημοσίευση από τον έλληνα επιστήμονα Παπαγεωργίου πρότεινε μια εναλλακτική μέθοδο που βασίζεται σε Haar wavelets, αντί της συνήθους έντασης της εικόνας. Οι Viola και Jones , υιοθέτησαν την ιδέα της χρησιμοποίησης των Haar wavelets και αναπτύχθηκαν έτσι τα χαρακτηριστικά τύπου Haar. Ένα Haar χαρακτηριστικό θεωρεί παρακείμενες ορθογώνιες περιφέρειες σε μια συγκεκριμένη θέση σε ένα παράθυρο ανίχνευσης, συνοψίζει τις εντάσεις των pixel σε αυτές τις περιοχές και υπολογίζει τη διαφορά μεταξύ τους. Η διαφορά αυτή στη συνέχεια χρησιμοποιείται για την κατηγοριοποίηση των υποπεριοχών μιας εικόνας. Για παράδειγμα, ας πούμε ότι έχουμε μια βάση δεδομένων εικόνων με ανθρώπινα πρόσωπα. Είναι κοινή διαπίστωση ότι σε όλα τα πρόσωπα η περιοχή των ματιών είναι πιο σκούρα από την περιοχή των μάγουλων. Συνεπώς, ένα κοινό χαρακτηριστικό Haar για την ανίχνευση των προσώπων είναι το σύνολο των δύο γειτονικών ορθογωνίων που βρίσκονται πάνω από την περιοχή των ματιών και τα μάγουλα. Η θέση αυτών των ορθογωνίων ορίζεται σε σχέση με ένα παράθυρο ανίχνευσης που λειτουργεί σαν ένα πλαίσιο οριοθέτησης(bounding box) με το αντικείμενο-στόχο (το πρόσωπο σε αυτήν την περίπτωση).

Εικόνα 27 Δείχνει τα δύο πρώτα χαρακτηριστικά που επιλεχθήκανε μέσω AdaBoost από τους Viola-Jones.Το πρώτο είναι το παράδειγμα που ανφέραμε παραπάνω(haar – γνώρισμα 2-ορθογωνίων) ενώ στο δεύτερο χρησιμοποιουνται οι περιοχές μύτης και ματιών σαν χαρακτηριστικό (haar – γνώρισμα 3-ορθογωνίων).

Κατά το στάδιο της ανίχνευσης της μεθόδου Viola-Jones πλαισίου ανίχνευσης, ένα παράθυρο στο μέγεθος του αρχείου κινείται πάνω από την εικόνα εισόδου, και για κάθε υποτμήμα της εικόνας του υπολογίζεται Haar-χαρακτηριστικό. Η διαφορά αυτή, στη συνέχεια, συγκρίνετε με ένα όριο (threshold) το οποίο έχει μάθει ο ταξινομητής και χωρίζει μη αντικείμενα από αντικείμενα. Επειδή ένα τέτοιο Haar-χαρακτηριστικό θεωρείται ένας αδύναμος μαθητής ή ταξινομητής (η ποιότητα ανίχνευση του είναι ελαφρώς καλύτερη από μία τυχαία εικασία) ένας μεγάλος αριθμός Haar-γνωρισμάτων είναι αναγκαίος για να περιγράψουμε ένα αντικείμενο με επαρκή ακρίβεια . Ως εκ τούτου η μέθοδος Viola-Jones οργανώνει τα Haar-χαρακτηριστικά σε κάτι που ονομάζεται αλυσίδα-ταξινομιτής ώστε να σχηματίσει ένα ισχυρό μαθητή ή ταξινομητή.Το βασικό πλεονέκτημα ενός Haar-χαρακτηριστικού σε σχέση με άλλου τύπου χαρακτηριστικά είναι η ταχύτητα υπολογισμού τους. Λόγω της χρήσης των αναπόσπαστων εικόνων(integral images),ένα Haar-γνώρισμα ανεξαρτήτως μεγέθους μπορεί να υπολογιστούν σε συγκεκριμένο χρόνο (περίπου 60 οδηγίες μικροεπεξεργαστή για ένα χαρακτηριστικό 2-ορθογωνίων).

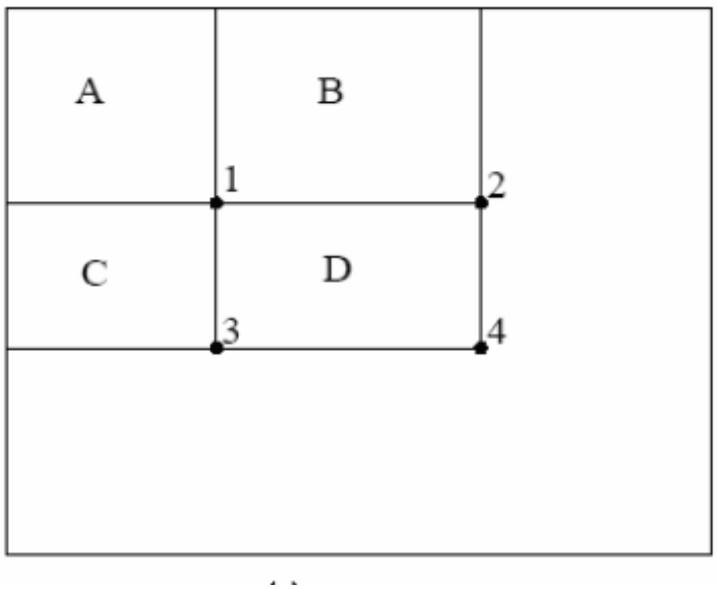
● Τύποι χαρακτηριστικών Haar στη μέθοδο Viola-Jones και ο υπολογισμός τους με integral images

Ένα απλό ορθογώνιο Haar-χαρακτηριστικό μπορεί να οριστεί ως η διαφορά του αθροίσματος των pixels των περιοχών εντός του ορθογωνίου, το οποίο μπορεί να βρίσκεται σε οποιαδήποτε θέση και κλίμακα κατά την αρχική εικόνα. Αυτό το τροποποιημένο σύνολο χαρακτηριστικών γνωρισμάτων ονομάζεται χαρακτηριστικό 2-ορθογωνίων.Οι Viola και Jones ορίσανε και χρησιμοποιήσανε επίσης χαρακτηριστικά 3-ορθογωνίων και 4-ορθογωνίων. Οι τιμές τους δείχνουν ορισμένα χαρακτηριστικά της συγκεκριμένης περιοχής της εικόνας. Κάθε τύπος χαρακτηριστικού μπορεί να αναφέρει την ύπαρξη (ή απουσία) ορισμένα συστατικών της εικόνας, όπως τα άκρα ή αλλαγές στην υφή. Για παράδειγμα, ένα χαρακτηριστικό 2-ορθογωνίων μπορεί να αναφέρει πού βρίσκονται τα σύνορα μεταξύ μιας σκοτεινής περιοχής και μιας φωτισμένης περιοχής.



Εικόνα 27 Τα (A) και (B) είναι χαρακτηριστικά 2-ορθογωνίων,το (C) δείχνει ένα χαρακτηριστικό 3-ορθογωνίων και το (D) ένα 4-ορθογωνίων.

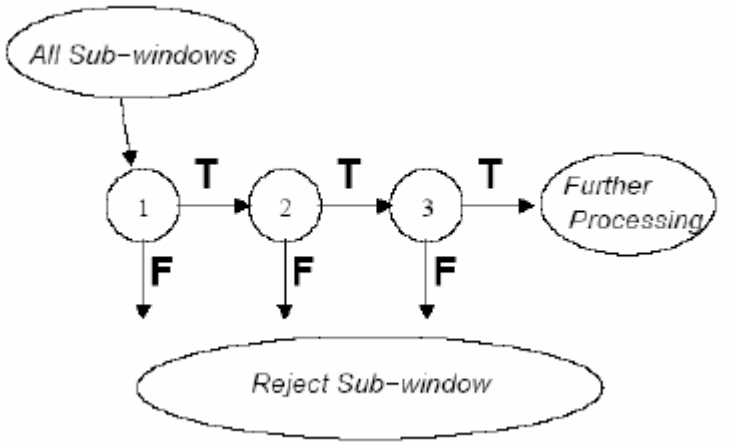
Μία από τις εισφορές των Viola και Jones ήταν να χρησιμοποιήσουν πίνακες συνόψισεως τομέων (summed area tables), τους οποίους αποκάλεσαν αναπόσπαστα εικονίδια(integral images).Οι αναπόσπαστες εικόνες μπορούν να οριστούν ως δύο διαστάσεων τομείς αναζήτησης σε μορφή πίνακα με το ίδιο μέγεθος της αρχικής εικόνας. Κάθε στοιχείο της ολοκληρωμένης εικόνας περιλαμβάνει το άθροισμα όλων των pixels που βρίσκεται στην επάνω αριστερή περιοχή της αρχικής εικόνας (σε σχέση με τη θέση του στοιχείου). Αυτό επιτρέπει τον υπολογισμό του αθροίσματος των ορθογώνιων περιοχών της εικόνας, σε οποιαδήποτε θέση ή κλίμακα, χρησιμοποιώντας μόνο τέσσερις αναζητήσεις:



Εικόνα 28 Η τιμή του integral image στο 1 είναι το σύνολο των pixel στο ορθογώνιο A .Η τιμή στο σημείο 2 είναι A+B,στο σημείο 3 A+C και στο σημείο 4 A+B+C+D . Το σύνολο του ορθογωνίου D είναι 4+1-(2+3).

Για κάθε Haar-χαρακτηριστικό μπορεί να χρειαστούν περισσότερες από τέσσερις αναζητήσεις, ανάλογα με το πώς ορίστηκε.Για 2-ορθογωνίων χαρακτηριστικά χρειάζονται έξι αναζητήσεις, 3-ορθογωνίων χαρακτηριστικά χρειάζονται οκτώ αναζητήσεις, και 4-ορθογωνίων χαρακτηριστικά ανάγκη εννέα αναζητήσεις.

● Η αλυσίδα ανίχνευσης (Detection cascade)



Εικόνα 28 Σχηματική αναπαράσταση της αλυσίδας ανίχνευσης ή αλυσίδας ταξινομίσεων (Detection cascade)

Η εφαρμογή που περιγράφεται χρησιμοποιεί δύο χαρακτηριστικά για το πρώτο στάδιο που μπορεί να απορρίψει το 50% των υπο-παραθύρων που δεν περιέχουν πρόσωπα, διατηρώντας ταυτόχρονα κοντά στο 100% εκείνων που περιέχουν πρόσωπα.Το επόμενο στάδιο, έχει δέκα χαρακτηριστικά, τα οποία μπορούν να απορρίπτουν το 80% των ψευδών θετικών του σταδίου 1, ταξινομόντας ταυτόχρονα όλα τα πρόσωπα. Οι επόμενοι δύο ταξινομητές έχουν είκοσι πέντε χαρακτηριστικά ο καθένας ακολουθούμενοι από τρείς ταξινομητές πενήντα χαρακτηριστικών γνωρισμάτων. Η πλήρης αλυσίδα αποτελείται από 38 στάδια με συνολικά 6060 χαρακτηριστικά. Είναι σε σημαντικό να σημειωθεί εδώ είναι ότι κάθε συγκεκριμένο στάδιο στην αλυσίδα είναι εκπαιδευμένο μόνο για τις εικόνες που δεν περιέχουν πρόσωπα και δεν έχουν ταξινομιθεί ορθώς μέχρι αυτό το στάδιο της αλυσίδας. Ο μέγιστος αριθμός των εικόνων χωρίς πρόσωπα που χρησιμοποιήθηκαν σε οποιοδήποτε στάδιο ήταν 6000, ενώ η ίδια βάση δεδομένων των προσώπων χρησιμοποιήθηκε σε κάθε στάδιο.

● Στάδιο Εκπαίδευσης

Η εκπαίδευση για τον ανιχνευτή προσώπου έγινε στο Matlab με χρήση 4916 εικόνων προσώπου και για 150.000 εικόνων χωρίς πρόσωπο που επιλέχθηκαν τυχαία. Όλες οι εικόνες κατάρτισης ήταν σε κλίμακα 24x24 pixels.Το ελάχιστο μέγεθος για κάθε χαρακτηριστικό επιλέχθηκε να είναι 8x8 και μείωσε τον συνολικό αριθμό των χαρακτηριστικών σε περίπου 50.000. Η τελική αλυσίδα ανίχνευσης αποτελείται από πέντε στάδια - πρώτο στάδιο έχει 9 χαρακτηριστικά, το επόμενο έχει 10, ακολουθούμενο από 25, με τα δύο τελευταία αποτελούνται από 100 χαρακτηριστικά. Ο αριθμός των σταδίων της αλυσίδας μπορεί εύκολα να αυξηθεί με μερικές απλές τροποποιήσεις στον κώδικα C.Αν και ο συνολικός αριθμός των σταδίων της αλληλουχίας της και το μέγεθος του κάθε επιμέρους στάδιο όσον αφορά τον αριθμό των χαρακτηριστικών δεν είναι κρίσιμα για την απόδοση του συστήματος όσον αφορά την ακρίβεια, ωστόσο, μία έξυπνη επιλογή αυτών των παραμέτρων μπορεί να ενισχύσει την συνολική ταχύτητα ανίχνευσης. Οι αρχικές φάσεις έχουν μικρότερο αριθμό χαρακτηριστικών, έτσι ώστε τα περισσότερα από τα «εύκολα» χωρίς πρόσωπο παράθυρα να μπορούν να απορριφθούν στις αρχές χωρίς μεγάλο κόστος υπολογισμού.Τα πιο «δύσκολα» υπο-παράθυρα πρεπει να τεθούν σε πιο περίπλοκη εξέταση αλλά από την άλλη είναι πολύ λιγότερα σε αριθμο.

Παράρτημα κώδικα ανίχνευσης

• Σχεδιάγραμμα ροής κώδικα

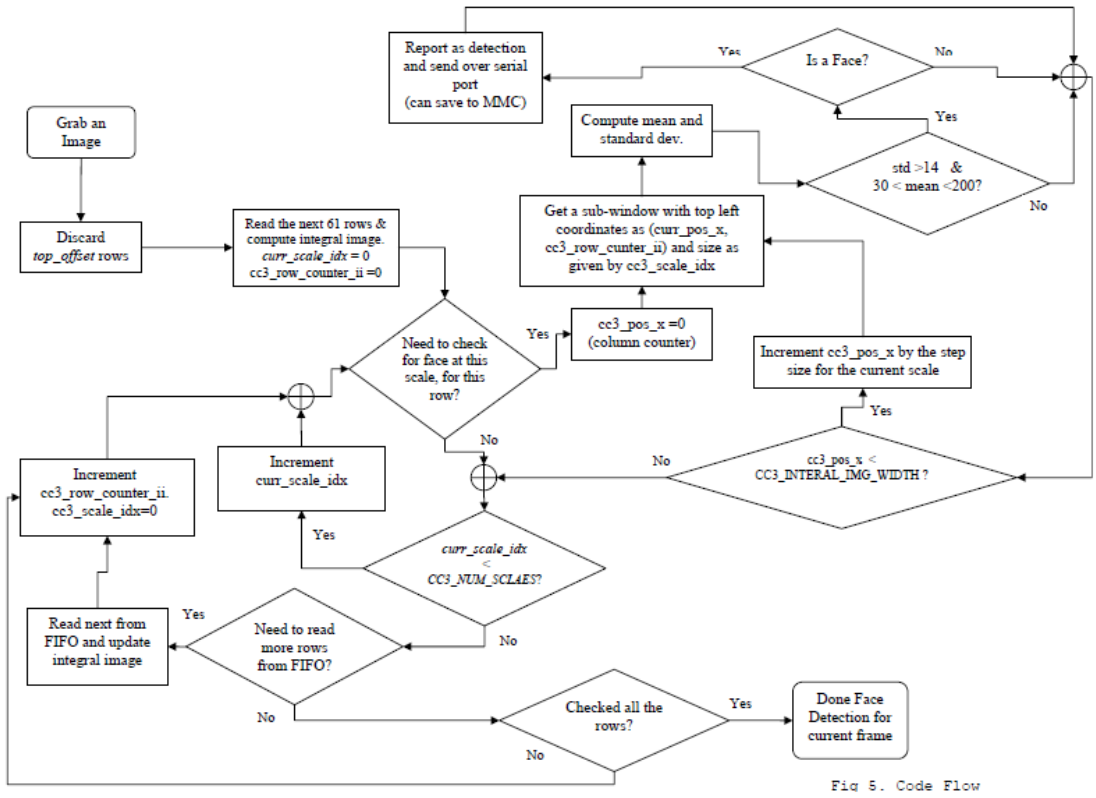


Fig 5. Code Flow

• Κώδικας ανίχνευσης σε C

```
#include <math.h>

#include <stdbool.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <ctype.h>
```

```
#include <cc3.h>
#include <cc3_ilp.h>
#include "vj.h"

/* define for saving the images to mmc */
#define SAVE_IMAGES

/* ---global variables----*/

/* buffer to get the image rows from fifo to compute integral image */
cc3_image_t cc3_img_tmp;

/* keep track of detected faces */
uint8_t cc3_num_detected_faces = 0;

/* row counter in the integral image to denote the current "top row"
   i.e. the row for which the sub-windows are being evaluated for features */
uint8_t cc3_row_counter_ii;

/* row counter in the cropped image (i.e. with top & bottom offset) */
uint8_t cc3_row_counter_cropped_img;

/* row counter in the actual full image */
uint16_t cc3_row_counter_actual_img;

/* integral image counter (for many rows, integral image has been calculated) */
```

```
uint16_t cc3_row_counter_calc_ii;

/* keep track of the current cascade stage for the current sub-window */
int8_t curr_cascade;

/* keep track of no. of frames (or rather no. of images taken) */
int16_t num_frames;

uint32_t mean_val;      /* mean for a subwinow */
uint32_t sum_pix;       /* sum of pixels in a sub-window */
uint32_t sum_pix_sq; /* sum of squares of pixels in a subwindow */
uint32_t std;       /* standard deviation */
uint32_t var;       /* variance  */

uint32_t vert_past_sum_sq_pix[4]; /* length should be equal to CC3_NUM_SCALES
*/
uint32_t horz_past_sum_sq_pix[4];

/* array to store rows while transfering b/w fifo and ii */
static uint8_t image_row[CC3_LO_RES_WIDTH*3];

#ifdef SAVE_IMAGES
FILE* fp;
FILE* fout;
#endif
```

```
/*
 * function to get the current segment from the actual image and simultaneously
 * compute imtegral image for the new segment added
 */
void cc3_get_curr_segment()
{
  cc3_pixel_t pix_temp;



  if (cc3_row_counter_cropped_img == 0)  /* first time */
    {
      // load the upper "CC3_INTEGRAL_IMG_HEIGHT" rows
      for (uint8_t i = 0; i < CC3_INTEGRAL_IMG_HEIGHT; i++)
            {
              /* get a row from the fifo */
              cc3_pixbuf_read_rows(cc3_img_tmp.pix, cc3_img_tmp.height);

              /* copy the first pixel */
              cc3_get_pixel(&cc3_img_tmp, 0, 0, &pix_temp);
              cc3_integral_image[i][0] = pix_temp.channel[1]; // only green channel



              #ifdef SAVE_IMAGES
              fprintf(fp, "%lu ", cc3_integral_image[i][0]);
              #endif
```

```
            // start copying from the next pixel and calculate cumulative sum at the
same time
            for (uint16_t j = 1; j < cc3_img_tmp.width; j++)
             {
               cc3_get_pixel(&cc3_img_tmp, j, 0, &pix_temp);
               cc3_integral_image[i][j] = pix_temp.channel[1]; // only green channel

               #ifdef SAVE_IMAGES
               fprintf( fp,"%lu ",cc3_integral_image[i][j] );
               #endif

               // compute cumulative sum across the row
               cc3_integral_image[i][j] += cc3_integral_image[i][j-1];
             }

        #ifdef SAVE_IMAGES
            fprintf( fp, "\n" );
            #endif
          }

    // find cumulative sum along columns to complete the integral image
computation
    for (uint16_t j = 0; j < CC3_INTEGRAL_IMG_WIDTH; j++)
          {
            for (uint16_t i = 1; i < CC3_INTEGRAL_IMG_HEIGHT; i++)  // start from
second row
              {
```

```
            cc3_integral_image[i][j]+=cc3_integral_image[i-1][j];
          }
        }
  }

 else
  {
    // get a new row
    cc3_pixbuf_read_rows(cc3_img_tmp.pix, cc3_img_tmp.height);

    uint8_t newly_added_row = cc3_row_counter_calc_ii %
CC3_INTEGRAL_IMG_HEIGHT;
    uint8_t prev_row = (cc3_row_counter_calc_ii - 1 + CC3_INTEGRAL_IMG_HEIGHT)
% CC3_INTEGRAL_IMG_HEIGHT;

    // copy the first pixel
    cc3_get_pixel(&cc3_img_tmp, 0, 0, &pix_temp);
    cc3_integral_image[newly_added_row][0] = pix_temp.channel[1];

    #ifdef SAVE_IMAGES
    fprintf( fp,"%lu ",cc3_integral_image[newly_added_row][0] );
    #endif

    // read the row, from next pixel onward and compute cum sum across the row
    for (uint16_t j = 1; j < cc3_img_tmp.width; j++)
        {
          cc3_get_pixel(&cc3_img_tmp, j, 0, &pix_temp);
```

```
            cc3_integral_image[newly_added_row][j] = pix_temp.channel[1];


            #ifdef SAVE_IMAGES

            fprintf( fp,"%lu ",cc3_integral_image[newly_added_row][j] );

        #endif


            // compute cumulative sum across the row

            cc3_integral_image[newly_added_row][j] +=
cc3_integral_image[newly_added_row][j-1];

          }


     #ifdef SAVE_IMAGES

     fprintf( fp, "\n" );

     #endif


     // find cumulative sum along columns to complete the integral image
computation

     for (uint16_t j = 0; j < CC3_INTEGRAL_IMG_WIDTH; j++)

          {


cc3_integral_image[newly_added_row][j]+=cc3_integral_image[prev_row][j];

          }


   }

}
```

```
/*
 * function to get the feature value for a particular sub-window
 *
 */
int16_t cc3_get_feat_val(uint8_t feat_num, uint8_t curr_scale, uint16_t x, uint16_t
y)
{
  int32_t fval = 0;
  uint16_t x_in_ii, y_in_ii;

  /* for stage1 cascade */
  if (curr_cascade == 0)
    {
     // evaluate the subwindow for this feature
     for (uint8_t pt=0; pt < 9 ; pt++)
          {
            x_in_ii = x + CC3_FACE_FEATURES0[feat_num][curr_scale].x[pt];
            y_in_ii = (y + CC3_FACE_FEATURES0[feat_num][curr_scale].y[pt]) %
CC3_INTEGRAL_IMG_HEIGHT; // circular warping

fval+=cc3_integral_image[y_in_ii][x_in_ii]*CC3_FACE_FEATURES0[feat_num][curr_sc
ale].val_at_corners[pt];
          }

     // mult by scaling factor
     fval = fval * CC3_SCALING_FACTOR;
```

```
    // taking into account the effect of normalization
    fval = fval/((int32_t)std);

    // check if fval < threshold
    if (fval*CC3_FACE_FEATURES0[feat_num][curr_scale].parity < CC3_FACE_FEATURES0[feat_num][curr_scale].thresh*CC3_FACE_FEATURES0[feat_num][curr_scale].parity)
        return CC3_FACE_FEATURES0[feat_num][curr_scale].alpha;
    else
        return 0;
  }

 /* for stage2 cascade */
 else if (curr_cascade == 1)
  {
    // evaluate the subwindow for this feature
    for (uint8_t pt=0; pt < 9 ; pt++)
        {
          x_in_ii = x + CC3_FACE_FEATURES1[feat_num][curr_scale].x[pt];
          y_in_ii = (y + CC3_FACE_FEATURES1[feat_num][curr_scale].y[pt]) % CC3_INTEGRAL_IMG_HEIGHT; // circular warping

fval+=cc3_integral_image[y_in_ii][x_in_ii]*CC3_FACE_FEATURES1[feat_num][curr_scale].val_at_corners[pt];
        }

    // mult by scaling factor
    fval = fval * CC3_SCALING_FACTOR;
```

```
    // taking into account the effect of normalization
    fval = fval/((int32_t)std);

    // check if fval < threshold
    if (fval*CC3_FACE_FEATURES1[feat_num][curr_scale].parity < CC3_FACE_FEATURES1[feat_num][curr_scale].thresh*CC3_FACE_FEATURES1[feat_num][curr_scale].parity)
        return CC3_FACE_FEATURES1[feat_num][curr_scale].alpha;
    else
        return 0;
  }

 /* for stage3 cascade */
 else if (curr_cascade == 2)
  {
    // evaluate the subwindow for this feature
    for (uint8_t pt=0; pt < 9 ; pt++)
        {
          x_in_ii = x + CC3_FACE_FEATURES2[feat_num][curr_scale].x[pt];
          y_in_ii = (y + CC3_FACE_FEATURES2[feat_num][curr_scale].y[pt]) % CC3_INTEGRAL_IMG_HEIGHT; // circular warping

fval+=cc3_integral_image[y_in_ii][x_in_ii]*CC3_FACE_FEATURES2[feat_num][curr_scale].val_at_corners[pt];
        }

    // mult by scaling factor
```

```
    fval = fval * CC3_SCALING_FACTOR;


    // taking into account the effect of normalization
    fval = fval/((int32_t)std);


    // check if fval < threshold
    if (fval*CC3_FACE_FEATURES2[feat_num][curr_scale].parity < CC3_FACE_FEATURES2[feat_num][curr_scale].thresh*CC3_FACE_FEATURES2[feat_num][curr_scale].parity)
        return CC3_FACE_FEATURES2[feat_num][curr_scale].alpha;
    else
        return 0;


  }


 /* for stage4 cascade */
 else if (curr_cascade == 3)
  {
    // evaluate the subwindow for this feature
    for (uint8_t pt=0; pt < 9 ; pt++)
        {
          x_in_ii = x + CC3_FACE_FEATURES3[feat_num][curr_scale].x[pt];
          y_in_ii = (y + CC3_FACE_FEATURES3[feat_num][curr_scale].y[pt]) % CC3_INTEGRAL_IMG_HEIGHT; // circular warping

fval+=cc3_integral_image[y_in_ii][x_in_ii]*CC3_FACE_FEATURES3[feat_num][curr_scale].val_at_corners[pt];
        }
```

```
    // mult by scaling factor
    fval = fval * CC3_SCALING_FACTOR;

    // taking into account the effect of normalization
    fval = fval/((int32_t)std);

    // check if fval < threshold
    if (fval*CC3_FACE_FEATURES3[feat_num][curr_scale].parity <
CC3_FACE_FEATURES3[feat_num][curr_scale].thresh*CC3_FACE_FEATURES3[feat_n
um][curr_scale].parity)
        return CC3_FACE_FEATURES3[feat_num][curr_scale].alpha;
    else
        return 0;
  }

 /* for stage5 cascade */
 else if (curr_cascade == 4)
  {
    // evaluate the subwindow for this feature
    for (uint8_t pt=0; pt < 9 ; pt++)
        {
          x_in_ii = x + CC3_FACE_FEATURES4[feat_num][curr_scale].x[pt];
          y_in_ii = (y + CC3_FACE_FEATURES4[feat_num][curr_scale].y[pt]) %
CC3_INTEGRAL_IMG_HEIGHT; // circular warping

fval+=cc3_integral_image[y_in_ii][x_in_ii]*CC3_FACE_FEATURES4[feat_num][curr_sc
ale].val_at_corners[pt];
```

```
        }


    // mult by scaling factor
    fval = fval * CC3_SCALING_FACTOR;


    // taking into account the effect of normalization
    fval = fval/((int32_t)std);


    // check if fval < threshold
    if (fval*CC3_FACE_FEATURES4[feat_num][curr_scale].parity <
CC3_FACE_FEATURES4[feat_num][curr_scale].thresh*CC3_FACE_FEATURES4[feat_n
um][curr_scale].parity)
        return CC3_FACE_FEATURES4[feat_num][curr_scale].alpha;
    else
        return 0;
  }


 else
  return 0;
}




/*--------------- main starts from here----------------*/
int main (void)
{


 // name of the curr image
```

```
  char img_name[50];

  uint8_t x2, y2; // lower right sub-window coordinates
  uint16_t num_pixels; // no. of pixels in a subwindow

  int32_t temp1, temp2; // temp variables used for computations

  cc3_filesystem_init();

  // configure uarts
  cc3_uart_init (0,
CC3_UART_RATE_115200,CC3_UART_MODE_8N1,CC3_UART_BINMODE_BINARY);

  cc3_camera_init ();

  cc3_camera_set_colorspace(CC3_COLORSPACE_RGB);
  cc3_camera_set_resolution(CC3_CAMERA_RESOLUTION_LOW);
  cc3_camera_set_auto_white_balance(true);
  cc3_camera_set_auto_exposure(true);

  printf("Face Detector...\n\r");

  cc3_led_set_state(0,0);
  cc3_led_set_state(1,0);
  cc3_led_set_state(2,0);
```

```
/* sample wait command in ms  */
cc3_timer_wait_ms(1000);
cc3_led_set_state(0,1);

/* setup integral image structure */
cc3_img_tmp.channels=3; // RGB color
cc3_img_tmp.width=cc3_g_pixbuf_frame.width;  // equal to Int_Img_Width
cc3_img_tmp.height = 1;  // image will hold just 1 row for scanline processing
cc3_img_tmp.pix = &image_row;

if (cc3_img_tmp.width != CC3_INTEGRAL_IMG_WIDTH)
 {
   printf("Error...image width and integral image width different \n\r");
 }

num_frames = 0;

while(1)
 {
   cc3_num_detected_faces = 0;
   /* initliaze the past records of sum_sq_pix to zero */
   for (uint8_t i = 0; i < CC3_NUM_SCALES; i++)
        {
          vert_past_sum_sq_pix[i] = (uint32_t) 0;
          horz_past_sum_sq_pix[i] = (uint32_t) 0;
        }
```

```
// printf(" New Frame...\n\r");
 printf( "START\r" );
 /* new image  */

 #ifdef SAVE_IMAGES

 sprintf(img_name, "%s%04d%s","c:/img",num_frames, ".pgm");
 fp=fopen(img_name,"w" );
 if (fp == NULL)
      {
        printf("%s %s\n\r", "Error Opening: ",img_name);

      }

 fprintf( fp, "P2\n%d %d\n255\n", cc3_g_pixbuf_frame.width,
cc3_g_pixbuf_frame.height-top_offset-bottom_offset );
 sprintf(img_name, "%s%04d%s","c:/img",num_frames,".txt");
 fout = fopen(img_name, "w");
 if (fout == NULL)
      {
        printf("%s %s\n\r", "Error Opening  ",img_name);
      }
```

```
#endif

/* This tells the camera to grab a new frame into the fifo and reset
      any internal location information */
cc3_pixbuf_load();

/* discard some top rows, given by top_offset */
for (uint8_t i = 0; i < top_offset ; i++)
     {
      cc3_pixbuf_read_rows(cc3_img_tmp.pix, cc3_img_tmp.height);
     }


/* indicating how many rows have we read in the actual full image
      top_offset rows are discarded and next "CC3_INTEGRAL_IMG_HEIGHT" rows are read in the first
      instance of the following for loop */
cc3_row_counter_actual_img = (top_offset-1)+1;   /* index starts from 0 */
cc3_row_counter_ii = 0;
cc3_row_counter_cropped_img = 0;
cc3_row_counter_calc_ii = (CC3_INTEGRAL_IMG_HEIGHT - 1);

/* iterating over the rows of the actual image until the bottom offset */
while (cc3_row_counter_actual_img < (CC3_IMAGE_HEIGHT - bottom_offset))
     {
      if (cc3_row_counter_calc_ii < CC3_IMAGE_HEIGHT - top_offset - bottom_offset) /* check if we need to load any more rows? */
```

```
            {
                /* get the current segment and its integral image */
                cc3_get_curr_segment();
            }

           for (uint8_t curr_scale_idx = 0; curr_scale_idx < CC3_NUM_SCALES;
curr_scale_idx++)
            {
                /* check if we need to evaluate subwindows at this scale for this sub-
window */
                if
(cc3_rows_to_eval_feat[cc3_row_counter_cropped_img][curr_scale_idx] == 1)
                 {
                   /* iterate over all horizontal shifted sub-window */
                   for (uint8_t curr_pos_x =0; curr_pos_x <
CC3_INTEGRAL_IMG_WIDTH - CC3_SCALES[curr_scale_idx];
curr_pos_x+=CC3_WIN_STEPS[curr_scale_idx])
                    {
                        /* compute the bottom right coordinates for this window */
                        x2 = curr_pos_x + CC3_SCALES[curr_scale_idx]-1 ;
                        y2 = (cc3_row_counter_ii + CC3_SCALES[curr_scale_idx]-1) %
CC3_INTEGRAL_IMG_HEIGHT;
                        if (x2 >= CC3_INTEGRAL_IMG_WIDTH )
                         {
                           printf("*Error....width outside limits!! \n\r");
                         }

                        /* Don't consider the first row and first column in the sub-
window for calculating std & mean */
```

```
                    num_pixels = (CC3_SCALES[curr_scale_idx]-
1)*(CC3_SCALES[curr_scale_idx]-1);


                    temp1 = cc3_integral_image[cc3_row_counter_ii][curr_pos_x]
-  cc3_integral_image[cc3_row_counter_ii][x2];
                    temp2 = cc3_integral_image[y2][x2] -
cc3_integral_image[y2][curr_pos_x];
                    sum_pix = (uint32_t) (temp1 + temp2);
                    mean_val = sum_pix/num_pixels;


                    /* if first column --> use the past vertical sum_sq_pix */
                    /* first time, calculate sum_sq_pix for every scale */
                    if ( (curr_pos_x == 0) && (cc3_row_counter_cropped_img ==
0) )
                     {
                       vert_past_sum_sq_pix[curr_scale_idx] = (uint32_t) 0;
                       horz_past_sum_sq_pix[curr_scale_idx] = (uint32_t) 0;


                       sum_pix_sq = 0;
                       for (uint8_t i = (cc3_row_counter_ii+1) %
CC3_INTEGRAL_IMG_HEIGHT; i != (y2+1)% CC3_INTEGRAL_IMG_HEIGHT; )
                        {
                             for (uint8_t j = curr_pos_x+1; j <= x2; j++)
                              {
                                temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;
                                temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];
```

```
                                    temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];

                                    temp1 = temp1+temp2;

                                    temp1 = temp1*temp1;  /* square of pixel intensity
*/


                                    /* update horizontal past sum_sq_pix */

                                    if ( j > (uint8_t) (curr_pos_x +
CC3_WIN_STEPS[curr_scale_idx]) )

                                     {

                                          horz_past_sum_sq_pix[curr_scale_idx]+=
temp1;

                                     }


                                    /* update vertical past sum_sq_pix */

                                    if ( i > (uint8_t) (cc3_row_counter_ii +
CC3_WIN_STEPS[curr_scale_idx]) )

                                     {

                                          vert_past_sum_sq_pix[curr_scale_idx]+=
temp1;

                                     }


                                    /* update the sum_sq_pix for this window */

                                    sum_pix_sq+= (uint32_t) temp1;


                                  }


                                 i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;
```

```
                    }


                  }

                 /* if the first col but not the first crow --> use vertical past
sum_sq_pix
                    - need to update both vert past & horiz past sum_sq_pix */

                 else if (curr_pos_x == 0)
                  {
                    uint8_t y11 = (cc3_row_counter_ii+1) %
CC3_INTEGRAL_IMG_HEIGHT;

                    /* use the past vert sum_sq_pix */
                    sum_pix_sq = vert_past_sum_sq_pix[curr_scale_idx];

                    /* calcuate sum_sq_pix for top horizontal block */
                    uint32_t top_horz = (uint32_t) 0;
                    uint8_t i = y11;

                    for (uint8_t k = 0; k < CC3_WIN_STEPS[curr_scale_idx]; k++)
                     {
                          for (uint8_t j = curr_pos_x+1; j <= x2; j++)
                           {
                             temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;
```

```
                                    temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];

                                    temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];

                                    temp1 = temp1+temp2;

                                    temp1 = temp1*temp1;  /* square of pixel intensity
*/


                                    top_horz+=temp1;


                                  }


                                i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;

                               }


                              /* calculate left vertical block */

                              uint32_t left_vert = (uint32_t) 0;


                              for (i = y11; i != (y2+1)% CC3_INTEGRAL_IMG_HEIGHT; )

                               {

                                    for (uint8_t j = curr_pos_x+1; j <= curr_pos_x +
CC3_WIN_STEPS[curr_scale_idx]; j++)

                                    {

                                      temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;

                                      temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];

                                      temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];
```

```
                                temp1 = temp1+temp2;

                                temp1 = temp1*temp1;  /* square of pixel intensity
*/


                                left_vert+= temp1;

                               }


                             i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;


                          }


                        /* calculate sum_sq_pix for new pixels */


                        i = (y2 - CC3_WIN_STEPS[curr_scale_idx] + 1 +
CC3_INTEGRAL_IMG_HEIGHT) % CC3_INTEGRAL_IMG_HEIGHT;
                        for ( ; i != (y2 +1 ) % CC3_INTEGRAL_IMG_HEIGHT; )
                         {
                              for (uint8_t j = curr_pos_x+1; j <= x2; j++)
                               {
                                temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;
                                temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];
                                temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];
                                temp1 = temp1+temp2;
                                temp1 = temp1*temp1;  /* square of pixel intensity
*/
```

```
                    sum_pix_sq+= temp1;

                  }
                 i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;

                }

               /* update vert past sum_sq_pix */
               vert_past_sum_sq_pix[curr_scale_idx] = (sum_pix_sq -
top_horz);

               /* update horz past sum_sq_pix */
               horz_past_sum_sq_pix[curr_scale_idx] = (sum_pix_sq -
left_vert);

              }


             /* else use horizontal past sum_sq_pix, no need to update vert
past sum_sq_pix */
             else
              {
               /* use the past horz sum_sq_pix */
               sum_pix_sq = horz_past_sum_sq_pix[curr_scale_idx];
```

```
                    uint8_t y11 = (cc3_row_counter_ii+1) %
CC3_INTEGRAL_IMG_HEIGHT;


                    /* claculate left vertical block */
                    uint32_t left_vert = (uint32_t) 0;


                    for (uint8_t i = y11; i != (y2+1)%
CC3_INTEGRAL_IMG_HEIGHT; )
                     {
                        for (uint8_t j = curr_pos_x+1; j <= curr_pos_x +
CC3_WIN_STEPS[curr_scale_idx]; j++)
                         {
                           temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;
                           temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];
                           temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];
                           temp1 = temp1+temp2;
                           temp1 = temp1*temp1;  /* square of pixel intensity
*/
                           left_vert+= (uint32_t) temp1;


                         }


                        i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;
                     }
```

```
                        /* calculate right vertical block and add to sum_sq_pix */
                        for (uint8_t i = y11; i != (y2+1)%
CC3_INTEGRAL_IMG_HEIGHT; )
                         {
                              for (uint8_t j = x2; j >= x2 -
CC3_WIN_STEPS[curr_scale_idx] + 1; j--)
                               {
                                 temp2 = (i-1+CC3_INTEGRAL_IMG_HEIGHT) %
CC3_INTEGRAL_IMG_HEIGHT;
                                 temp1 = cc3_integral_image[i][j] -
cc3_integral_image[i][j-1];
                                 temp2 = -cc3_integral_image[temp2][j] +
cc3_integral_image[temp2][j-1];
                                 temp1 = temp1+temp2;
                                 temp1 = temp1*temp1;  /* square of pixel intensity
*/

                                 sum_pix_sq+= (uint32_t)temp1;
                               }

                              i = (i+1) % CC3_INTEGRAL_IMG_HEIGHT;

                         }

                        /* update horz past sum_sq_pix */
                        horz_past_sum_sq_pix[curr_scale_idx] = (sum_pix_sq -
left_vert);
                      }
```

```
/* after sum_sq_pixels has been calculated */
uint64_t numer, denom;

numer = ((uint64_t)num_pixels*(uint64_t)sum_pix_sq);
numer =  (numer - (uint64_t)sum_pix*(uint64_t)sum_pix);
denom = ((uint64_t)num_pixels*(uint64_t)num_pixels);
var = (uint32_t)(numer/denom);


if (var < 200)
 {
   std = 1;  /* basically reject the windows with this small
variance */
 }

else
 {
   /* find the standard deviation */
   for (uint32_t i = 14; i < var/2 ; i++)
    {
         if ((i*i <= var) & ((i+1)*(i+1) > var))
          {
            if ((var - i*i) < ((i+1)*(i+1) - var))
              std = i;
            else
```

```
                    std = i+1;


                    break;
                   }
                 }
               }

              /* check for face only if following conditions are satisfied,
otherwise its too uniform  */
              if ( (std > 13) & ( (mean_val > 30) | (mean_val < 200)) )
               {
                int16_t feat_sum = 0;
                int8_t face = 1;

                /* check if this sub-window is a face */

                curr_cascade = 0;
                face = 1;
                while ( (curr_cascade < CC3_NUM_CASCADES) & (face))
                 {
                     feat_sum = 0;
                     for (uint8_t curr_feat_idx = 0; curr_feat_idx <
CC3_NUM_FEATURES[curr_cascade]; curr_feat_idx++)
                      {
                       feat_sum+=cc3_get_feat_val(curr_feat_idx,
curr_scale_idx, curr_pos_x, cc3_row_counter_ii);
                      }
```

```
                    /* if its a face, go to next cascade otherwise quit */
                    if (feat_sum >
CC3_GLOBAL_THRESHOLD[curr_cascade])
                     {
                      face = 1;
                      curr_cascade++;
                     }
                    else
                     {
                      face = 0;
                     }
                  }


                 if (face)
                  {
               #ifdef SAVE_IMAGES
                    fprintf(fout, "%d %d %d \n",curr_pos_x+1,
cc3_row_counter_cropped_img+1, CC3_SCALES[curr_scale_idx]-1);
                    #endif


                    printf("Face Detected at: %d %d, Size: %d
\n\r",curr_pos_x+1, cc3_row_counter_cropped_img+1, CC3_SCALES[curr_scale_idx]-
1);
                    cc3_num_detected_faces++;
```

```
                                }

                                else {printf("No. of faces : %d \n\r",
cc3_num_detected_faces);}









                            } /* end of if (std > lower_bound) and ( lower_bound < mean
< upper_bound) */



                        } /* end of iterating over horizontally shifted sub-windows */



                    } /* end of if loop for this particular scale at this row */



                } /* end of iterating over all scales */



                /*  increment the row counters */

                cc3_row_counter_actual_img++;

                cc3_row_counter_cropped_img++;

                cc3_row_counter_ii = (cc3_row_counter_ii + 1) %
CC3_INTEGRAL_IMG_HEIGHT; /* circular wrapping of integral image row counter */

                cc3_row_counter_calc_ii++;



            } /* end of iterating over all the rows in the actual image (upto
bottom_offset) */
```

```
//   printf("No. of faces : %d \n\r", cc3_num_detected_faces);

/* end of frame */
printf ("Frame Done..\n\r");
    #ifdef SAVE_IMAGES

fprintf( fout, "%d %d %d \n",0,0,0);
fclose(fout);
fclose(fp);
#endif

    num_frames++;


    cc3_led_set_state(0,0);
    cc3_led_set_state(2,1);
    while(!cc3_button_get_state());
    cc3_led_set_state(0,1);
    cc3_led_set_state(2,0);
    // wait for the button to be pressed for the next frame


    // sample non-blocking serial routine
    if(!cc3_uart_has_data(0) ) break;
```

```
    } // end of while

    free(cc3_img_tmp.pix);  // don't forget to free!

    while(1);

    return 0;
}
```

Βιβλιογραφία

Andreasson, H. (2009). *Camera based Navigation by Mobile Robots: Local Visual Feature based Localisation and Mapping.* VDM Verlag Dr. Müller .

Autonomous robot. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Autonomous_robot

Brooks, R. A. (1985). *A robust layered control system for a mobile robot.* Massachusetts: MIT, Artificial Intelligence Laboratory.

Computer vision. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Computer_vision

Craig, J. (2005). *Εισαγωγή στη Ρομποτική.* Εκδόσεις Τζιόλα.

Cummins, M., & Newman, P. M. (2007). *Probabilistic Appearance Based Navigation and Loop Closing.* Oxford: Oxford University Mobile Robotics Research Group.

Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer graphics and image processing,14* , σσ. 227-248.

Dissanayake, M., P.Newman, Clark, S., Durrant-Whyte, H., & Csorba, M. (2006). *A Solution to the Simultaneous Localisation and Map Building (SLAM) Problem.* Australian Centre for Field Robotics, Department of Mechanical and Mechatronic Engineering, The University of Sydney.

Euclidean distance. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Euclidean_distance

Image registration. (n.d.). Ανάκτηση από Wikipedia: http://en.wikipedia.org/wiki/Image_registration

Kuipers, B., & Byun, Y.-T. (1991). *A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations.* Austin, Texas: University of Texas at Austin, Department of Computer Sciences.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, (σσ. 1150-1157).

Machine vision. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Machine_vision

Moutarlier, P., & Chatila, R. (1989). Stochastic multisensor data fusion for mobile robot localization and environment modelling. *International Symposium in Robotics Research*, (σσ. 85-99).

Scale-invariant feature transform. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Segmentation (image processing). (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Segmentation_%28image_processing%29

Shortest path problem. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Shortest_path_problem

Simultaneous localization and mapping. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

Tardos, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2001). *Robust Mapping and licalization in Indoor Environments using Sonar Data.* Massachusetts: MIT Dept. of Ocean Engineering, Universitad de Zaragoza.

What is SLAM? (n.d.). Ανάκτηση από http://www.robots.ox.ac.uk/~pnewman/Intro2SLAM.htm

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *Interational journal of computer vision* , 119-152.

Σ.Γ, Τ. (2003). *Ρομποτική: Ανάλυση, Έλεγχος, Σχεδιασμός και Προγραμματισμός.* Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο.

Khatib, O., Quinlan, S., "*Elastic Bands: Connecting, Path Planning and Control*," inProceedings of IEEE International Conference on Robotics and Automation, Atlanta, GA, May 1993

Lumelsky, V., Skewis, T., "*Incorporating Range Sensing in the Robot Navigation Function*." IEEE Transactions on Systems, Man, and Cybernetics,20:1990, pp. 1058–1068.

Lumelsky, V., Stepanov, A., "*Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape*," in Autonomous Robot Vehicles. New York, Spinger-Verlag, 1990

Pankaj K. Agarwal, Lydia Kavraki, Matthew T. Mason, "Robotics: the algorithmic perspective : the Third Workshop on the Algorithmic Foundations of Robotics", A.K.Peters Ltd, Huston, 1998