Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Διακοπτικό Σύστημα Ελέγχου Κίνησης Ρομπότ με Χρήση Εκτίμησης Θέσης βάσει Συνόλου

Διπλωματικη Εργασια

του

Κριθαρούλα Διονύσιου

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής
Network Management and Optimal Design Laboratory

Διακοπτικό Σύστημα Ελέγχου Κίνησης Ρομπότ με Χρήση Εκτίμησης Θέσης βάσει Συνόλου

Διπλωματικη Εργασια

του

Κριθαρούλα Διονύσιου

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Φεβρουαρίου, 2024.

.....	.....	.....
Συμεών Παπαβασιλείου	Ιωάννα Ρουσσάκη	Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Αν. Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2024

.....

Διονυσιος Κριθαρουλας
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός διακοπτικού μηχανισμού μεταφόρτωσης υπολογιστικών διεργασιών στα άκρα του δικτύου, για ρομποτικές εφαρμογές της Βιομηχανίας 4.0. Οι εφαρμογές αυτές αφορούν κινητά ρομπότ τα οποία εκτελούν περίπλοκες διεργασίες, οι οποίες έχουν υψηλές απαιτήσεις τόσο σε χρονική απόκριση όσο και σε ασφάλεια. Σε αυτό το πλαίσιο, η μεταφόρτωση των διεργασιών στα άκρα του δικτύου επιτρέπει στα ρομπότ να ελαφρύνουν τον υπολογιστικό τους φόρτο, αναθέτοντας την εκτέλεση των παραπάνω διεργασιών σε μία ισχυρή υπολογιστική υποδομή σε κοντινή απόσταση. Στο σενάριο που εξετάζεται στην παρούσα διπλωματική εργασία θεωρούμε πως διαθέτουμε δύο υπολογιστικές διεργασίες που αφορούν τον εντοπισμού θέσης ενός κινητού ρομπότ: (i) έναν γρήγορο, τοπικά εκτελούμενο αλγόριθμο ο οποίος παρέχει μία αναξιόπιστη εκτίμηση για την ακριβή θέση και τον προσανατολισμό του ρομπότ και (ii) έναν χρονοβόρο, ωστόσο πιο ακριβή αλγόριθμο ο οποίος μπορεί να εκτελείται απομακρυσμένα στο άκρο του δικτύου. Ο σκοπός του διακοπτικού μηχανισμού είναι η απόφαση σε κάθε χρονική στιγμή για το εάν είναι απαραίτητη η χρήση του πιο χρονοβόρου αλγορίθμου προκειμένου το ρομπότ να αποκτήσει μία πιο ακριβή εκτίμηση της θέσης και του προσανατολισμού του ή εάν αρκεί η χρήση του τοπικού αλγορίθμου εντοπισμού θέσης. Ανάλογα με του τελικούς στόχους της ρομποτικής εφαρμογής, ο συγκεκριμένος μηχανισμός μπορεί να διαμορφωθεί κατάλληλα έτσι ώστε να παρέχει μία εξισορρόπηση μεταξύ της ακρίβειας που προσφέρεται από τον αλγόριθμο εντοπισμού θέσης που εκτελείται απομακρυσμένα στην άκρη του δικτύου και της υπολογιστικής ταχύτητας που προσφέρεται από τον αλγόριθμο εντοπισμού θέσης που εκτελείται τοπικά στο ρομπότ.

Για την ανάπτυξη του διακοπτικού μηχανισμού πραγματοποιήθηκε η σχεδίαση και υλοποίηση ενός αλγορίθμου απόφασης μεταφόρτωσης υπολογιστικών διεργασιών, ο οποίος λαμβάνει υπόψιν του την δυναμική φύση των κινήσεων του ρομπότ και αντιμετωπίζει την αβεβαιότητά που υπάρχει στον ακριβή εντοπισμό της θέσης του στο χρόνο. Πιο συγκεκριμένα, η απόφαση για τη μεταφόρτωση λαμβάνεται βάσει της αβεβαιότητας ως προς την τρέχουσα θέση του ρομπότ, την εγγύτητα του ρομπότ σε εμπόδια που βρίσκονται στον χώρο της κίνησης του αλλά και την δυσκολία της διαδρομής που αυτό καλείται να ακολουθήσει.

Ο βαθμός της αβεβαίοτητας που παρουσιάζει το ρομπότ σε κάθε χρονική στιγμή ως προς την ακριβή του θέση λαμβάνεται μέσω μίας προσεγγιστικής τεχνικής η οποία χρησιμοποιεί τα δεδομένα που προέρχονται από την οδομετρία του ρομπότ και τα ενσωματώνει σε ένα αντίστροφο μοντέλο κίνησης στο οποίο έχουν προστεθεί μεταβλητές θορύβου οι οποίες προσομοιώνουν τους τύπους της αβεβαιότητας που υπάρχουν στην κίνηση του ρομπότ. Μέσω του μοντέλου αυτού, υπολογίζεται σε κάθε χρονική στιμγή ένα σύνολο δειγμάτων το οποίο περιέχει προσεγγιστικά όλες τις πιθανές στάσεις στις οποίες μπορεί να βρίσκεται το ρομπότ την συγκεκριμένη χρονική στιγμή με βάση το σύνολο δειγμάτων που έχουν προκύψει από την προηγούμενη χρονική στιγμή. Το εμβαδόν που καταλαμβάνει το συγκεκριμένο σύνολο μεγαλώνει με την κίνηση του ρομπότ καθώς τα δείγματα του διασκορπίζονται σε ένα αυξανόμενο μεγάλο χώρο. Έτσι το συγκεκριμένο εμβαδόν αποτελεί μία καλή μετρική που ποσοτικοποιεί τον βαθμό της αβεβαιότητας που υπάρχει ως προς την ακριβή θέση του ρομπότ.

Η αξιολόγηση ολόκληρου του διακοπτικού μηχανισμού πραγματοποιήθηκε σε περιβάλλον προσομοίωσης χρησιμοποιώντας ένα κινητό ρομπότ διαφορικής οδήγησης το οποίο επιθυμεί να μεταβεί από μία γνωστή αρχική θέση σε μία γνωστή τελική θέση. Προκειμένου το ρομπότ να καταφέρει να φτάσει με επιτυχία στην τελική θέση, χρησιμοποιήθηκαν γνωστοί αλγόριθμοι τόσο για τον σχεδιασμό της διαδρομής που πρέπει να ακολουθήσει όσο και για την παρακολούθηση της διαδρομής αυτής.

Λέξεις-κλειδιά — Διακοπτικός Μηχανισμός, Εκτίμηση στάσης κινητού ρομπότ, Αβεβαιότητα ως προς την στάση του ρομπότ, Υπολογιστική ταχύτητα, Ακρίβεια εκτίμησης στάσης.

Abstract

The purpose of this thesis is the development of a switching offloading mechanism for robotic applications within Industry 4.0. These applications rely on mobile robotic agents performing numerous complex tasks that with strict safety and time requirements. In this context, by offloading their computationally intensive tasks to powerful computing infrastructures in their vicinity, robots can reduce their computational load. In the scenario examined in this thesis, we assume that two localization algorithm are available: (i) a fast, locally-executed one, providing an unreliable estimation while the robot is moving and (ii) a time-consuming, however more precise one, executed remotely on an edge server. The objective of the offloading switching mechanism is to determine at each time step whether the use of the remotely executed localization algorithm is necessary for the robot to obtain a more accurate estimate of its position and orientation, or whether the the locally executed localization algorithm is sufficient. Depending on the goals of the robotic application, this mechanism can be configured accordingly to balance the accuracy offered by the remotely executed localization algorithm against the computational speed provided by the locally executed localization algorithm.

For the development of the switching offloading mechanism, a utility function-based decision algorithm is designed and implemented that considers the dynamic nature of robot's movement and addresses the uncertainty regarding its position and orientation. More specifically, the decision is based on the uncertainty regarding the current robot pose, the proximity of the robot to obstacles during its navigation and the complexity of the path the robot must follow.

The degree of uncertainty at each time step regarding the current robot pose is obtained through an approximation technique. This technique utilizes from the robot's odometry and integrates them into an inverse motion model incorporating noisy variables that simulates the types of uncertainty present in the robot's motion.Through this model, a set of samples is calculated at each time instant containing approximately all the possible poses that the robot can reach at this particular time step based on the set of samples that have been generated in the previous time step. During robot's movement, the area occupied by this set grows as its samples are scattered over an increasingly large space. Thus, this area can serve as a metric that quantifies the degree of uncertainty with respect to the robot's current pose.

The evaluation of the switching mechanism was conducted in a simulation environment using a differential drive mobile robot that is able to move from a known initial position to a known final position. To facilitate successful navigation, widely used mobile robot path planning and path tracking algorithms were implemented.

Keywords — Switching Mechanism, Mobile robot pose estimation, Uncertainty regarding robot's pose, Computational speed, Accurate pose estimation.

Ευχαριστίες

Θα ήθελα να τονίσω ότι το έργο αυτό δεν θα ήταν δυνατό να πραγματοποιηθεί χωρίς την υποστήριξη πολλών ανθρώπων. Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα μου, κ. Συμεών Παπαβασιλείου, για την πολύτιμη καθοδήγηση του στην εκπόνηση αυτής της διπλωματικής, καθώς και τον κ. Δημήτριο Σπαθαράκη για τη στενή συνεργασία, την υποστήριξη και την καθοριστική συμβολή του στην εργασία αυτή.

Πολύ σημαντική για εμένα ήταν ακόμα η συναισθηματική συνεισφορά της οικογένειας και των φίλων μου που μου παρείχαν αγάπη, στήριξη και γέλιο. Ιδιαίτερα, θα ήθελα να ευχαριστήσω την μητέρα μου Ελένη αλλά και την αδερφή μου Αναστασία με την οποία μάλιστα μοιραστήκαμε αυτό το ακαδημαϊκό ταξίδι από την αρχή μέχρι το τέλος και βγήκαμε νικητές.

Contents

List of Figures

Chapter 1

Εισαγωγή

1.1 Computational Offloading

Τα δίκτυα επικοινωνίας τρέχουσας και επόμενης γενιάς αναμένεται να φιλοξενήσουν έναν μεγάλο αριθμό νέων και απαιτητικών σε πόρους εφαρμογών που μπορούν να προσφερθούν σε έναν μεγάλο εύρος τελικών χρηστών. Παρόλο που οι τελικές συσκευές γίνονται όλο και πιο ισχυρές, οι διαθέσιμοι τοπικοί πόροι δεν μπορούν να ανταπεξέλθουν στις απαιτήσεις αυτών των εφαρμογών. Για παράδειγμα, στον τομέα της συνεργατικής ρομποτικής, όπου οι άνθρωποι και τα ρομπότ συνεργάζονται σε δυναμικά περιβάλλοντα, αλγόριθμοι υψηλών υπολογιστικών απαιτήσεων επιτρέπουν στα ρομπότ να ανιχνεύουν το περιβάλλον τους και να ενεργούν κατάλληλα. Κατά συνέπεια υπάρχει μεγάλος όγκος πληροφοριών για επεξεργασία ενώ απαιτείται εκτέλεση πολύπλοκων αλγορίθμων σε πραγματικό χρόνο.

Οι ανάγκες αυτές έχουν οδηγήσει στο επονομαζόμενο Computational Offloading, όπου απαιτητικές υπολογιστικά εργασίες χρειάζεται να μεταφέρονται σε απομακρυσμένες συσκευές με περισσότερους και πιο ισχυρούς πόρους.

1.2 Cloud Computing

Το Cloud Computing είναι μια καλά δομημένη και ευρέως χρησιμοποιούμενη υποδομή που μπορεί να διευκολύνει το Computational Offloading. Πιο συγκεκριμένα, το Cloud Computing έχει φέρει επανάσταση στο διαδίκτυο και έχει αλλάξει τον τρόπο με τον οποίο οι εφαρμογές, το λογισμικό και οι πόροι προσφέρονται στις τελικές συσκευές. Σύμφωνα με το NIST [17], ως Cloud Computing ορίζεται "ένα μοντέλο που επιτρέπει βολική, κατά απαίτηση δικτυακή πρόσβαση σε μία κοινή δεξαμενή από διαμορφώσιμους υπολογιστικούς πόρους (π.χ. δίκτυα, διακομιστές, μνήμες, εφαρμογές και υπηρεσίες) που μπορούν να παρασχεθούν και να διανεμηθούν γρήγορα με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδραση με τον πάροχο της υπηρεσίας".

Τα οφέλη που παρέχει ένα τέτοιο μοντέλο είναι πολυάριθμα. Πιο αναλυτικά, οι υπολογιστικοί πόροι προσφέρονται στις τελικές συσκευές κατόπιν ζήτησης, ανεξάρτητα από τον τύπο της συσκευής και την τοποθεσία της. Επιπλέον οι υπολογιστικοί και δικτυακοί πόροι που είναι διαθέσιμοι στο Cloud, μπορεί να μοιράζονται μεταξύ πολλών τελικών συσκευών και να κλιμακώνουν δυναμικά.

Όσον αφορά το Computational Offloading, η πρακτική της εκφόρτωσης εργασιών υψηλών υπολογιστικών απαιτήσεων από τις τελικές συσκευές σε μία κεντρική υποδομή Cloud είναι μία λύση που ήδη έχει διερευνηθεί αρκετά. Το Cloud μπορεί να επεκτείνει τις δυνατότητες σε πόρους των τελικών συσκευών έχοντας ισχυρές τοπολογίες κέντρων δεδομένων. Εκτός αυτού είναι εξοπλισμένο με τα κατάλληλα εργαλεία και πλατφόρμες αυτοματισμού προκειμένου να προσφέρει την απαραίτητη διαφάνεια στις τελικές συσκευές κρύβοντας την πολυπλοκότητα και τις υλικοτεχνικές λεπτομέρειες της υλοποίησης του.

Παρά τα πολυάριθμα πλεονεκτήματα που προαναφέρθηκαν, το Cloud Computing θέτει ορισμένους σοβαρούς περιορισμούς. Αυτοί οι περιορισμοί, αν και υπάρχουν από την αρχή του Cloud, δεν έχουν επισυμανθεί αρκετά μέχρι πρόσφατα. Ο λόγος είναι ότι νέες τεχνολογίες επικοινωνίας, νέες εφαρμογές και υπηρεσίες έχουν αυξήσει

τον όγκο δεδομένων που παράγονται και ταυτόχρονα έχουν αυξήσει τις απαιτήσεις για επικοινωνίες χαμηλής καθυστέρησης. Ως εκ τούτου, προσφέροντας πόρους οι οποίοι βρίσκονται συγκεντρωμένοι σε μία κεντρική δομή πολύ μακριά από τις τελικές συσκευές δημιουργεί σοβαρές καθυστερήσεις. Αυτές οι καθυστερήσεις μπορεί να είναι καταστροφικές σε εφαρμογές όπου η επιτυχία τους είναι αναγκαία, όπως εφαρμογές που σχετίζονται με την υγεία, ή εφαρμογές που είναι κρίσιμο να πραγματοποιηθούν όσο το δυνατόν πιο γρήγορα, όπως ο έλεγχος ρομποτικών συσκευών που χρησιμοποιούνται στην βιομηχανία.

Δεν υπάρχει αμφιβολία πως απαιτείται μία πιο κατανεμημένη υποδομή που θα ενισχύσει την τοπική αποτελεσματικότητα φέρνοντας παρόμοιες δυνατότητες με αυτές που προσφέρει το Cloud πιο κοντά στις τελικές συσκευές, στην άκρη του δικτύου. Αυτός ακριβώς είναι και ο λόγος της εμφάνισης νέων μοντέλων όπως είναι το Fog και το Edge Computing.

1.3 Fog Computing

Το Fog Computing ήταν η πρώτη πρωτοβουλία της βιομηχανίας να ορίσει ρητά μια αρχιτεκτονική για την εφαρμογή των δυνατοτήτων που παρέχει το Cloud στην άκρη του δικτύου. Συγκεκριμένα ο όρος Fog Computing επινοήθηκε από τον Cisco το 2012 και ορίζεται ως "η διαδικασία επέκτασης των δυνατοτήτων του Cloud Computing στην άκρη του δικτύου." Το Fog Computing ενσωματώνει υπολογιστικούς, δικτυακούς και αποθηκευτικούς πόρους κοντά στο επίπεδο των συσκευών για να διευκολύνει την επεξεργασία δεδομένων. Γίνεται λοιπόν φανερό πως το Fog Computing εισήχθει για να διευκολύνει την παρακολούθηση, τον έλεγχο και την ανάλυση των συσκευών σε πραγματικό χρόνο αφαιρώντας την μεγάλη καθυστέρηση που υπάρχει στην επικοινωνία μεταξύ των συσκευών και των κεντρικών διακομιστών που βρίσκονται σε ένα απομακρυσμένο Cloud. Στόχος του είναι η εν μέρη επεξεργασία του φόρτου εργασίας και των υπηρεσιών τοπικά στις Fog συσκευές αντί για την μετάδοση τους στο Cloud.

Σημαντικό είναι επίσης πως το Fog Computing μπορεί να υποστηρίξει και να προωθήσει εφαρμογές οι οποίες δεν ταιριάζουν με το Cloud όπως εφαρμογές που περιλαμβάνουν πολύ χαμηλό και σταθερό λανθάνοντα χρόνο, γεωγραφικά κατανεμημένα συστήματα όπως είναι ο έλεγχος αγώνων και τα δίκτυα αισθητήρων, κινητές εφαρμογές όπως είναι τα έξυπνα συνδεδεμένα αυτοκίνητα, συστήματα προσαρμοστικού ελέγχου μεγάλη κλίμακας όπως έξυπνη παροχή ενέργειας και έξυπνα φανάρια κίνησης.

Συμπερασματικά, το Fog Computing κατάφερε να παρακάμψει πολλούς από τους περιορισμούς του Cloud Computing αυξάνοντας την απόδοση των συσκευών μέσω του Computational Offloading. Ωστόσο ακόμα πιο αυστηρές απαιτήσεις, όπως εξαιρετικά χαμηλή καθυστέρηση και υψηλή αξιοπιστία, έχουν δημιουργήσει την ανάγκη για πληροφορίες που είναι τοποθετημένες ακόμα πιο κοντά στις τελικές συσκευές.

1.4 Edge Computing

Το Edge Computing μπορεί να οριστεί ως ένα επίπεδο δικτύου που περιλαμβάνει τις τελικές συσκευές και τους χρήστες τους, προκειμένου να παρέχει τοπικές υπολογιστικές δυνατότητες σε αισθητήρες, έξυπνους μετρητές, ή άλλες συσκευές προσβάσιμες στο δίκτυο. Το Edge Computing έχει συσχετιστεί και με τον όρο Mist Computing. Όπως ορίζει το όνομα, το Mist Computing καλύπτει την υπολογιστική και επικοινωνιακή ικανότητα που είναι διαθέσιμη στο ίδιο επίπεδο με τις τελικές συσκευές. Σύμφωνα με το NIST, το Mist Computing ορίζεται ως ένας ελαφρύς τύπος του Fog Computing το οποίο βρίσκεται στην άκρη του δικτύου, φέρνοντας το επίπεδο του Fog Computing πιο κοντά στις τελικές συσκευές.

1.5 Edge και Fog Computing στην Ρομποτική

Ένας τομέας όπου η ανάγκη του Fog και Edge Computing είναι εμφανής είναι αυτός των ρομποτικών εφαρμογών. Την τελευταία δεκαετία, πολύ πολύπλοκες ρομποτικές εφαρμογές οι οποίες σχετίζονται μεταξύ άλλων με αυτόνομα κινητά ρομπότ, βραχίονες αλλά και συνεργασία μεταξύ ρομπότ και ανθρώπου έχουν αναδυθεί. Αποτελεσματική, ασφαλής αλλά και αυτόνομη λειτουργία των ρομπότ στην βιομηχανία, την υγεία, την εκπαίδευση και εξερεύνηση απαιτεί την εκτέλεση αλγορίθμων απαιτητικών σε υπολογιστική ισχύ και μνήμη οι οποίοι σχετίζονται με την επεξεργασία εικόνων, τον σχεδιασμό κίνησης, τον εντοπισμό θέσης, την χαρτογράφηση και την ρομποτική

μάθηση. Παράλληλα με τις υψηλές υπολογιστικές απαιτήσεις, οι ρομποτικές εφαρμογές είναι σημαντικό να εκτελούνται με ελάχιστη καθυστέρηση. Για παράδειγμα, ένα ρομπότ χρειάζεται να γνωρίζει την θέση των εμποδίων που βρίσκονται στον δρόμο του πριν συγκρουστεί με αυτά. Σε περίπτωση που ο χρόνος εκτέλεσης του αλγορίθμου που ανιχνεύει τα εμπόδια είναι πολύ μεγάλος, κάτι τέτοιο μπορεί να αποβεί μοιραίο για την ασφάλεια του ρομπότ.

Έτσι το Fog και Edge Computing μέσω του Computational Offloading μπορούν να αποτελέσουν μία πολύ αποτελεσματική λύση προσφέροντας υπολογιστικούς πόρους κοντά στο σημείο που βρίσκεται η ρομποτική συσκευή ώστε να επιτευχθεί η ελάχιστη δυνατή καθυστέρηση. Πολλές ευκαιρίες για Computational Offloading εμφανίζονται σε αλγορίθμους σχεδίασης κίνησης και χαρτογράφησης που προορίζονται για ρομποτικούς βραχίονες, κινητά ρομπότ και άλλα είδη ρομποτικών συστημάτων. Αξίζει να σημειωθεί ότι υπάρχουν ήδη διαθέσιμα εμπορικά προϊόντα που επιτρέπουν το Computational Offloading σε ρομποτικές εφαρμογές.

Όσο και αν το Computational Offloading φαίνεται κατάλληλο στην πραγματικότητα από μόνο του δεν αρκεί. Και αυτό γιατί ένας αριθμός από ανεπιθύμητα φαινόμενα ενδέχεται δυνητικά να λάβουν χώρα κατά την μετάδοση και επεξεργασία της πληροφορίας όπως η καθυστέρηση δικτύου, η μεταβλητή ποιότητα υπηρεσίας και η διακοπή λειτουργίας για συγκεκριμένο χρόνο.

Για τους λόγους αυτούς εφαρμογές όπως τα αυτόνομα κινητά ρομπότ συχνά έχουν κάποια ικανότητα για τοπική επεξεργασία όταν στοχεύουν σε αποκρίσεις χαμηλής καθυστέρησης ή κατά περιόδους που η πρόσβαση στο δίκτυο δεν είναι διαθέσιμη ή είναι αναξιόπιστη. Κατά συνέπεια, μία σημαντική πρόκληση, από πλευράς σχεδιασμού ελέγχου, εκτίμησης κατάστασης, και δικτυακής βελτιστοποίησης είναι ο συνδυασμός τοπικών και απομακρυσμένων πόρων με αποτελεσματικό τρόπο.

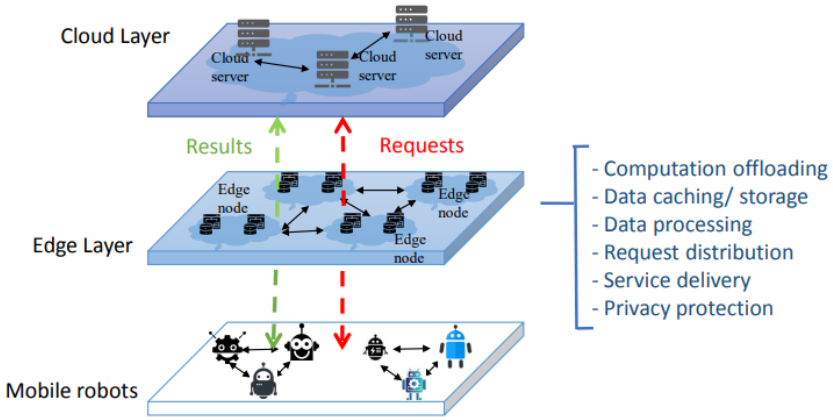

Figure 1.5.1: Αρχιτεκτονική μεταξύ Ρομπότ, Edge και Cloud

1.6 Προκλήσεις

Στις μέρες μας, τεράστιες προσπάθειες γίνονται με στόχο την βελτίωση της απόδοσης των ρομπότ μέσω της χρήσης του computational offloading. Ωστόσο, υπάρχουν ακόμα αρκετές ανοιχτές προκλήσεις για τις οποίες απαιτείται περαιτέρω έρευνα. Στην συνέχεια αναφέρονται ορισμένες από αυτές:

- *Ασφάλεια*: Η ασφάλεια είναι μια σημαντική διάσταση της απόδοσης σε όλα τα ρομποτικά σενάρια στα οποία συμμετέχει και το δίκτυο. Η επικοινωνία και η μεταφορά δεδομένων μεταξύ ρομπότ και απομακρυσμένων υπολογιστικών πόρων μπορεί να οδηγήσει σε ορισμένες απειλές που σχετίζονται με την ασφάλεια των ρομποτικών αυτών συστημάτων. Για παράδειγμα, ένας εισβολέας μπορεί να εκτελέσει παράνομες εργασίες που ενδέχεται να απαγορεύσουν σε άλλες εργασίες την πρόσβαση σε ενδιάμεσα δεδομένα. Το γεγονός αυτό ενδέχεται να οδηγήσει σε σοβαρές ζημιές κατά την εκτέλεση της ρομποτικής εφαρμογής. Επιπλέον, μια κακόβουλη εργασία η οποία τροποποιεί τα ενδιάμεσα δεδομένα που ανταλλάσσονται στο δίκτυο μπορεί να επηρεάσει την εργασία που λαμβάνει αυτά τα δεδομένα ή τους υπολογιστικούς πόρους (είτε βρίσκονται

πάνω στο ίδιο το ρομπότ είτε είναι απομακρυσμένοι) που εκτελούν την συγκεκριμένη εργασία. Έτσι η μείωση των απειλών κατά της ασφάλειας είναι ένα σημαντικό ζήτημα που θα πρέπει να ληφθεί υπόψιν για την παροχή ενός ασφαλούς περιβάλλοντος εκτέλεσης ρομποτικών εφαρμογών.

- *Επίγνωση του πλαισίου*: Μία λύση που βασίζεται στο computational offloading θα είναι πιο αποτελεσματική εάν έχει την δυνατότητα να αντιδρά σε αλλαγές που λαμβάνουν χώρα στις συνθήκες του δικτύου, στην διαθεσιμότητα των υπολογιστικών πόρων, στην τοποθεσία του ρομπότ κ.λπ. Δυστυχώς παρά τις προσπάθειες που γίνονται για την βελτίωση της απόδοσης του computational offloading, οι διακυμάνσεις που παρατηρούνται στις συνθήκες του δικτύου, εξακολουθούν να είναι ένα κρίσιμο ζήτημα που δεν έχει διερευνηθεί στις περισσότερες από τις λύσεις που έχουν προταθεί μέχρι σήμερα.

- *Απώλεια σύνδεσης*: Ένας αρκετά σοβαρός κίνδυνος που μπορεί να προκύψει κατά την διάρκεια του computational offloading είναι η απώλεια της σύνδεσης με τους απομακρυσμένους υπολογιστικούς πόρους. Για παράδειγμα συχνή είναι η περίπτωση όπου ένα ρομπότ είναι απασχολημένο με την εκτέλεση ορισμένων εργασιών ενώ έχει μεταφορτώσει άλλες για υπολογισμό σε απομακρυσμένους πόρους. Σε περίπτωση που το ρομπότ χάσει την επικοινωνία του με τους απομακρυσμένους πόρους θα αναλάβει αναγκαστικά την εκτέλεση όλων των εργασιών τοπικά, κάτι το οποίο θα οδηγήσει σε καθυστερήσεις στην εκτέλεση της εφαρμογής και σε απώλεια πληροφορίας.

- *Ανοχή σε σφάλματα*: Μια από τις πιο κρίσιμες προκλήσεις κατά την χρήση του computational offloading για κινητά ρομπότ είναι η εξάρτηση από τις συνθήκες το δικτύου. Η εξάρτηση αυτή μπορεί να προκαλέσει δυσλειτουργία στην διαδικασία της εκφόρτωσης, η οποία θα οφείλεται είτε σε κακή σύνδεση μεταξύ ρομπότ και απομακρυσμένων πόρων είτε σε πρόβλημα που σχετίζεται με τον απομακρυσμένο διακομιστή. Δεδομένου ότι τα ρομπότ πολλές φορές λειτουργούν σε κρίσιμες εφαρμογές, όπως είναι για παράδειγμα ο στρατός, είναι ζωτικής σημασίας να αποφευχθούν αυτές οι αστοχίες και να διασφαλίζεται η επιτυχής εκτέλεση της εφαρμογής. Όταν παρουσιαστεί κάποιο πρόβλημα σύνδεσης, τότε το κινητό ρομπότ θα πρέπει να συνεχίζει την εκτέλεση των αλγόριθμών του χρησιμοποιώντας άλλες εναλλακτικές, όπως για παράδειγμα τους τοπικούς πόρους που διαθέτει. Όλα τα σφάλματα που μπορεί να συμβούν κατά την διάρκεια της εκφόρτωσης, όπως η απώλεια σύνδεσης με τον διακομιστή ή η εναλλαγή διακομιστών, συνοδεύονται από επιπλέον κατανάλωση ενέργειας και χρόνου εκτέλεσης. Μια αποτελεσματική λύση που βασίζεται στο computational offloading θα πρέπει να λαμβάνει υπόψιν το κόστος αυτών των σφαλμάτων αλλά και τις λειτουργίες που πρέπει να πραγματοποιηθούν για την αντιμετώπιση τους. Δυστυχώς τα περισσότερα συστήματα επικεντρώνονται μόνο στην λύση που βασίζεται στο computational offloading χωρίς να υλοποιούν κάποια διαδικασία ανάκτησης ή διαδικασία χειρισμού μιας ενδεχόμενης αποτυχίας.

- *Μοντέλο για την απόδοση του Computational Offloading*: Η μέτρηση της απόδοσης μία λύσης που βασίζεται στο computational offloading είναι μία ακόμα πρόσθετη πρόκληση. Το computational offloading πρόβλημα μπορεί να μοντελοποιηθεί ως ένα σύστημα όπου η ενέργεια ή η καθυστέρηση είναι οι τυπικές μεταβλητές εξόδου ενώ οι διαθέσιμοι υπολογιστικοί πόροι (π.χ. CPU, μνήμη), τα εισερχόμενα αιτήματα και το εύρος ζώνης του δικτύου είναι οι μεταβλητές εισόδου. Στις περισσότερες από τις τρέχουσες μελέτες, τα μοντέλα απόδοσης που έχουν προκύψει είναι μίας εισόδου/μίας εξόδου, εμπειρικά προερχόμενα ή σταθερά. Αν και η υπόθεση αυτή είναι ρεαλιστική, ο χρόνος επεξεργασίας μιας εργασίας που έχει μεταφορτωθεί εξαρτάται από πολλούς χρονικά μεταβαλλόμενους παράγοντες οι οποίοι συνήθως δεν είναι εύκολα μετρήσιμοι. Από την άλλη πλευρά, μοντέλα πολλών εισόδων-πολλών εξόδων είναι πιο ακριβή, αλλά η διαδικασία αναγνώρισης είναι συνήθως κουραστική. Συγκεκριμένα, η απόφαση για offloading εκτελείται επαρκώς μόνο για συγκεκριμένες συνθήκες λειτουργίας, μην μπορώντας να εγγυηθεί την σταθερότητα όταν υπάρχουν διακυμάνσεις στον φόρτο εργασίας και ετερογενείς υποδομές επικοινωνίας, όπως συμβαίνει με τις IoT συσκευές. Επομένως, το μοντέλο του συστήματος θα πρέπει να προσαρμοστεί προκειμένου να συμπεριληφθούν οι μετρικές απόδοσης, που μπορούν να ρυθμιστούν από τις παραμέτρους ελέγχου (δηλαδή τις μεταβλητές εισόδου του συστήματος). Το σύστημα αυτό θα είναι ικανό να αντιλαμβάνεται δομικές αλλαγές που ερμηνεύονται ως διακριτά άλματα στην δυναμική του συστήματος όπως για παράδειγμα η κίνηση του ρομπότ, οι αλλαγές στις συνθήκες του δικτύου, και η προσθήκη/αφαίρεση Edge διακομιστών.

- *Σχεδιασμός Ελεγκτή*: Μια ενεργή και πολύ πρακτική πρόκληση στο πλαίσιο του computational offloading είναι ο σχεδιασμός του ελεγκτή των ρομποτικών συστημάτων. Στο πλαίσιο της ύπαρξης δυναμικών δικτύων αλλά και απομακρυσμένων υπολογιστικών πόρων, μια στρατηγική που έχει σχεδιαστεί ώστε να λαμβάνει απόφαση από κοινού για το computational offloading, την κατανομή των πόρων και τον σχεδιασμό του ελεγκτή του ρομπότ είναι πιο κατάλληλη σε σύγκριση με την δημιουργία ξεχωριστών επιπέδων για τον

έλεγχο των απομακρυσμένων πόρων και του υπό-έλεγχο συστήματος. Αυτή η νέα γενιά ελεγκτών θα δημιουργηθεί με την συγχώνευση δύο μοντέλων:

- Το μοντέλο απόδοσης του συστήματος το οποίο αναφέρθηκε προηγουμένως και
- Το δυναμικό μοντέλο που περιγράφει την εξέλιξη της κατάστασης του συστήματος το οποίο αποτελείται από μεταβλητές που σχετίζονται με την θέση, τον προσανατολισμό και την ταχύτητα του ρομπότ αλλά και μεταβλητές σχετικές με τις συνθήκες φωτισμού, την θερμοκρασία του χώρου και την κατάσταση λειτουργίας των αισθητήρων.

Αξίζει να σημειωθεί πως στην επιστήμη του ελέγχου, διαμοιραζόμενα και ατελή δίκτυα επικοινωνίας μεταξύ του ελεγκτή και των αισθητήρων/ενεργοποιητών έχουν μελετηθεί λεπτομερώς, δημιουργώντας τον κλάδο των Network Control Systems (NCS). Πολλές μέθοδοι που έχουν αναπτυχθεί λαμβάνουν υπόψιν φαινόμενα που συμβαίνουν συχνά στον πραγματικό κόσμο όπως χρονικές καθυστερήσεις ή εγκατάλειψη πακέτων, χρησιμοποιώντας μεθόδους που προέρχονται από την θεωρία διαταραχής, την θεωρία ευστάθειας Lyapunov και την ανάλυση υβριδικών συστημάτων ενώ χρησιμοποιούνται και πιθανοτικές μέθοδοι όπως αλυσίδες Markov και στοχαστικά αυτόματα. Ως αποτέλεσμα, μια σημαντική εξέλιξη θα είναι η δημιουργία *event-triggered* και *self-triggered* μεθόδων ελέγχου που θα επιτρέπουν την ασύγχρονη δειγματοληψία, μειώνοντας έτσι την κίνηση του δικτύου ενώ ταυτόχρονα θα παρέχουν εγγυημένες αντισταθμίσεις για την υποβάθμιση της απόδοσης του συστήματος κλειστού βρόγχου. Κατά συνέπεια, στο πλαίσιο του computational offloading, μια επίκαιρη πρόκληση είναι η σχεδίαση ελεγκτικών μηχανισμών οι οποίοι παρέχουν ταυτόχρονα τόσο δυναμικό computational offloading όσο και έλεγχο του ρομποτικού συστήματος, λαμβάνοντας υπόψιν τους διαθέσιμους και τους ζητούμενους υπολογιστικούς πόρους, τους διαθέσιμους πόρους της Edge υποδομής αλλά και την συνολική κατανάλωση ενέργειας του συστήματος. Αναμένεται ότι τέτοιοι αλγόριθμοι ελέγχου θα βελτιώσουν την απόδοση των συστημάτων, την χρήση της συνολικής υποδομής, καθώς και την ανθεκτικότητα και ευρωστία των προς έλεγχο συστημάτων

1.7 Αντικείμενο Διπλωματικής

Το σενάριο που εξετάζεται στην παρούσα διπλωματική εργασία περιλαμβάνει ένα κινητό ρομπότ διαφορικής οδήγησης εξοπλισμένο με δυνατότητες ανίχνευσης, υπολογισμού και ασύρματης επικοινωνίας, το οποίο μετακινείται από μία αρχική θέση σε μία θέση-στόχο σε έναν χώρο λειτουργίας (π.χ. ένα εργοστάσιο) ο οποίος περιλαμβάνει εμπόδια. Ένα τέτοιο σενάριο εμφανίζεται αρκετά συχνά στην Βιομηχανία 4.0, για παράδειγμα σε ρομπότ αποθήκευσης και εφοδιασμού τα οποία αυτοματοποιούν την λειτουργία αποθήκευσης και μετακίνησης αγαθών. Το ρομπότ σχεδιάζει αρχικά μία βέλτιστη διαδρομή από την αρχική του θέση στην θέση-στόχο και στην συνέχεια προσπαθεί να ακολουθήσει την διαδρομή αυτή με σκοπό τελικώς να φτάσει στον στόχο του. Η διαδρομή αυτή επαναϋπολογίζεται με μία προκαθορισμένη συχνότητα, καθώς το ρομπότ κινείται, θεωρώντας κάθε φορά ως αρχική θέση την τρέχουσα θέση του ρομπότ.

Σε ένα τέτοιο σενάριο η γνώση της ακριβούς θέσης και του προσανατολισμού του ρομπότ σε κάθε χρονική στιμγή είναι απαραίτητη προκειμένου τόσο το ρομπότ να μπορέσει να σχεδιάσει σωστά την βέλτιστη διαδρομή του όσο και να μπορέσει να ακολουθήσει την συγκεκριμένη διαδρομή φτάνοντας επιτυχώς στην θέση-στόχο χωρίς να συγκρουστεί με κάποιο από τα εμπόδια που βρίσκονται στον χώρο. Ωστόσο ένα πρόβλημα που εμφανίζεται αρκετά συχνά σε τέτοια σενάρια είναι το γεγονός ότι η αβεβαιότητα στην εκτίμηση της ακριβούς θέσης του ρομπότ αυξάνεται με την πάροδο του χρόνου κατά την κίνηση, λόγω ανακρίβειας των αισθητήρων, ολίσθησης των τροχών, βλάβης του υλικού κ.λπ. Έτσι η πραγματική θέση του ρομπότ ενδέχεται να είναι διαφορετική από αυτήν που εκείνο θεωρεί με αποτέλεσμα να μην καταφέρει τελικώς να φτάσει στον επιθυμητό στόχο ενώ επιπλέον υπάρχει μεγάλος κίνδυνος σύγκρουσης του με τα εμπόδια που βρίσκονται στον χώρο.

Μια πρώτη λύση στο παραπάνω πρόβλημα θα ήταν η χρήση ενός αλγορίθμου εντοπισμού θέσης ο οποίος θα εντόπιζε την θέση του ρομπότ με μεγάλη ακρίβεια. Δυστυχώς όμως στην πράξη τέτοιοι αλγόριθμοι έχουν υψηλές απαιτήσεις σε υπολογιστική ισχύ και αποθήκευση με αποτέλεσμα να μην μπορούν να εκτελεστούν στους τοπικούς πόρους που διαθέτει το ρομπότ ενώ δημιουργούν προβλήματα στην απόδοση του ρομπότ όταν εκτελούνται για μεγάλη χρονική διάρκεια καθώς δημιουργούν υψηλές καθυστερήσεις.

Στο δικό μας σενάριο, θεωρούμε πως υπάρχουν δύο αλγόριθμοι εντοπισμού θέσης: (i) ένας γρήγορος, τοπικά εκτελούμενος ο οποίος παρέχει μία αναξιόπιστη εκτίμηση για την ακριβή θέση και τον προσανατολισμό του ρομπότ και (ii) ένας χρονοβόρος, ωστόσο πιο ακριβής ο οποίος μπορεί να εκτελείται απομακρυσμένα σε έναν Edge

Server. Σκοπός μας είναι η δημιουργία ενός αλγορίθμου απόφασης μεταφόρτωσης υπολογιστικών διεργασιών ο οποίος θα διαμορφωθεί κατάλληλα έτσι ώστε να αναλάβει την στρατηγική της εναλλαγής μεταξύ των δύο αυτών αλγορίθμων. Η στρατηγική αυτή θα υπαγορεύει σε κάθε χρονική στιγμή ποια από τις δύο διαθέσιμες τεχνικές εντοπισμού θέσης θα χρησιμοποιηθεί; η επιρρεπής σε σφάλματα που εκτελείται τοπικά στο ρομπότ ή η ακριβής που απαιτεί σημαντικό αριθμό υπολογιστικών πόρων και εκτελείται απομακρυσμένα σε έναν Edge Server. Προκειμένου να αποφασίσει, ο αλγόριθμος θα λαμβάνει υπόψιν του τις ακόλουθες μετρικές:

- Την αβεβαιότητα ως προς την ακριβή θέση του ρομπότ.
- Την «πυκνότητα» των εμποδίων.
- Την καμπυλότητα της διαδρομής που πρέπει να ακολουθήσει το ρομπότ.

Σκοπός του αλγορίθμου είναι η εξισορρόπηση μεταξύ της ακρίβειας παρακολούθησης της διαδρομής που προσφέρεται από τον αλγόριθμο εντοπισμού θέσης που εκτελείται απομακρυσμένα στον Edge Server και της υπολογιστικής ταχύτητας που προσφέρεται από τον αλγόριθμο εντοπισμού θέσης που εκτελείται τοπικά στο ρομπότ.

Το διακοπτικό αυτό σύστημα υλοποιήθηκε και αξιολογήθηκε χρησιμοποιώντας ένα κινητό ρομπότ διαφορικής οδήγησης σε περιβάλλον προσομοίωσης. Κατά την αξιολόγηση του, γίνεται φανερό πως η χρήση του βοηθά στην επιτυχή έκβαση της αποστολής του ρομπότ, το οποίο φτάνει επιτυχώς στην θέση-στόχο παρά την αβεβαιότητα ως προς την θέση του χωρίς παράλληλα να γίνεται χρήση αποκλειστικά του απαιτητικού σε υπολογιστικούς πόρους αλγόριθμου εντοπισμού θέσης ο οποίος εισάγει καθυστερήσεις στην λειτουργία του ρομπότ.

1.8 Οργάνωση Κειμένου

Η παρούσα διπλωματική εργασία είναι οργανωμένη σε 5 Κεφάλαια. Πιο συγκεκριμένα:

- Στο Κεφάλαιο 2 παρουσιάζεται σχετική βιβλιογραφία που αφορά την μεταφόρτωση υπολογιστικών διεργασιών σε ρομποτικές εφαρμογές.
- Στο Κεφάλαιο 3 παρουσιάζεται το θεωριτικό υπόβαθρο το οποίο είναι χρήσιμο για την κατανόηση του επιστημονικού πεδίου με το οποίο σχετίζεται η συγκεκριμένη διπλωματική εργασία αλλά και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Επιπλέον αναλύονται λεπτομερώς οι ρομποτικοί αλγόριθμοι που χρησιμοποιήθηκαν στα πειράματα που ακολουθούν στην συνέχεια.
- Στο Κεφάλαιο 4 επεξηγείται η σχεδίαση και η υλοποίηση του διακοπτικού μηχανισμού αλλά και το πλαίσιο μέσα στο οποίο αξιολογήθηκε. Επιπλέον παραθέτονται τα πειράματα που διεξήχθησαν και αναλύονται τα συμπεράσματα που προέκυψαν όσον αφορά την χρησιμότητα του διακοπτικού μηχανιμσού που προτείνεται.
- Τέλος, στο Κεφάλαιο 5 πραγματοποιείται η σύνοψη της διπλωματικής εργσίας ενώ παρέχονται ιδέες και κατευθύνσεις για την προοπτική μελλοντικής της εξέλιξης.

Chapter 2

Σχετική Βιβλιογραφία

Οι λύσεις που βασίζονται στο Computational Offloading για ρομποτικά συστήματα εξαρτώνται από πολλούς και διαφορετικούς παράγοντες [2]. Ο στόχος για τον οποίο πραγματοποιείται το Computational Offloading αποτελεί την πρωταρχική παράμετρο που πρέπει κανείς να λαμβάνει υπόψιν. Στις πρώτες εφαρμογές του στα ρομποτικά συστήματα, το Computational Offloading είχε ως κύριο στόχο την μείωση της κατανάλωσης ενέργειας για τα κινητά ρομπότ. Ωστόσο, πιο πρόσφατα, οι ερευνητές εξετάζουν και άλλες μετρικές βελτιστοποίησης, όπως είναι οι εγγυήσεις εκτέλεσης σε πραγματικό χρόνο [14] αλλά και η ποιότητα των υπηρεσιών που παρέχεται από τις ρομποτικές εφαρμογές. Επιπλέον, η τοποθεσία των απομακρυσμένων πόρων έχει μεγάλο αντίκτυπο στην ποιότητα του Computational Offloading. Μοντέλα που βασίζονται στο Cloud αποτέλεσαν την πρώτη λύση για το Computational Offloading σε ρομποτικές εφαρμογές υψηλών υπολογιστικών απαιτήσεων [9]. Για παράδειγμα, το DAvinCi [1] ήταν η πρώτη Computational Offloading λύση που σχεδιάστηκε για ρομποτικά συστήματα που αλληλεπιδρούν με το Cloud. Σήμερα, διάφορα άλλα μοντέλα έχουν προταθεί για να φέρουν ισχυρούς υπολογιστικούς πόρους πιο κοντά στα κινητά ρομπότ όπως το Edge [13] και το Fog Computing [18]. Τέλος οι ερευνητές εξτάζουν διάφορες τεχνικές βελτιστοποίησης που αποφασίζουν πότε είναι κατάλληλο να πραγματοποιηθεί το Computational Offloading , συμπεριλαμβανωμένων ευρετικών μεθόδων, μετα-ευρετικών μεθόδων, τεχνικών μηχανικής μάθησης αλλά και θεωρίας παιγνίων.

Μετρικές Βελτιστοποίησης

Ένα πολύ σημαντικό κομμάτι του Computational Offloading είναι ο καθορισμός της χρονικής στιγμής όπου αυτό θα πραγματοποιηθεί. Η απόφαση για το πότε πρέπει να γίνει η εκφόρτωση των υπολογισμών σχετίζεται με τον καθορισμό των συνθηκών κάτω από τις οποίες θεωρείται ότι κάτι τέτοιο αξίζει. Η επιλογή των κατάλληλων μετρικών βελτιστοποίησης ανάλογα με τους στόχους της εφαρμογής καθορίζουν σε μεγάλο βαθμό την απόδοση του Computational Offloading.

Οι μετρικές βελτιστοποίησης που χρησιμοποιούνται συνήθως σε ρομποτικά σενάρια είναι οι εξής:

- *Καθυστέρηση*: Η καθυστέρηση ορίζεται ως ο χρόνος που απαιτείται για την ολοκλήρωση μίας υπολογιστικής εργασίας, συμπεριλαμβανομένου τόσο του χρόνου επεξεργασίας όσο και του χρόνου επικοινωνίας. Η ελαχιστοποίηση της καθυστέρησης είναι συχνά κρίσιμη στις ρομποτικές εφαρμογές, ειδικά για εργασίες που απαιτούν απόκριση σε πραγματικό χρόνο. Για αλγορίθμους που είναι απαιτητικοί σε υπολογιστική ισχύ, είναι σημαντικό να μειωθεί η καθυστέρηση τους και να βελτιωθεί η απόκριση. Έτσι, όταν η καθυστέρηση μίας ρομποτικής εφαρμογής αυξάνεται με την ύπαρξη εντατικών υπολογιστικά αλγορίθμων, η καλύτερη στρατηγική θα ήταν η εκφόρτωση των αλγορίθμων αυτών σε απομακρυσμένους πόρους για να εκμεταλλευτούμε την δυνατότητα που αυτοί παρέχουν στο να εκτελούνται υπολογισμοί με μεγαλύτερη ταχύτητα. Παρόλα αυτά και οι καθυστερήσεις στην επικοινωνία παίζουν σημαντικό ρόλο όταν θέλουμε να επιτύχουμε απόδοση πραγματικού χρόνου και έτσι πρέπει να λαμβάνονται υπόψιν.
- *Κατανάλωση ενέργειας*: Η κατανάλωση ενέργειας αποτελεί μία από τις βασικές μετρικές όπου η ελαχιστοποίηση της είναι κυρίαρχης σημασίας. Η μετρική αυτή εξαρτάται τόσο από τους υπολογισμούς όσο και από την επικοινωνία. Από την πλευρά του Computational Offloading, η επικοινωνία αποτελεί την κύρια πηγή κατανάλωσης ενέργειας λόγο της μετάδοσης μεγάλου όγκου δεδομένων από το ρομπότ στις

απομακρυσμένες υπολογιστικές μονάδες. Στην περίπτωση της τοπικής επεξεργασίας, ο ενσωματωμένος υπολογισμός που πραγματοποιείται τοπικά στο ρομπότ αποτελεί την κύρια πηγή κατανάλωσης ενέργειας. Ως εκ τούτου είναι σημαντικό να γίνει η βέλτιστη αντιστάθμιση μεταξύ υπολογισμού και επικοινωνίας για να μεγιστοποιηθεί το συνολικό όφελος.

- *Χρόνος Ολοκλήρωσης*: Ο χρόνος ολοκλήρωσης είναι μία ακόμα μετρική βελτιστοποίησης που συχνά το Computational Offloading στοχεύει να ελαχιστοποιήσει στις ρομποτικές εφαρμογές. Ο χρόνος ολοκλήρωσης ορίζεται ως ο συνολικός χρόνος που απαιτείται από την έναρξη μίας εργασίας μέχρι την επιτυχή ολοκλήρωση της.

- *Εύρος Ζώνης*: Η συγκεκριμένη μετρική βελτιστοποίησης ορίζεται ως το ποσό των δεδομένων που μεταδίδονται μέσω του δικτύου κατά την διάρκεια του computational offloading. Η αποτελεσματική χρήση του εύρους ζώνης είναι απαραίτητη για την αποφυγή συμφόρησης και καθυστέρησης στην επικοινωνία. Η ελαχιστοποίηση της περιττής μεταφοράς δεδομένων μπορεί να βελτιώσει την απόδοση του δικτύου.

- *Αξιοποίηση Πόρων*: Η συγκεκριμένη μετρική βελτιστοποίησης αφορά την αποτελεσματική χρήση των υπολογιστικών πόρων συμπεριλαμβανομένων τόσο των τοπικών όσο και των απομακρυσμένων. Η εξισορρόπηση του φορτίου και η αποτελεσματική χρήση των διαθέσιμων πόρων συμβάλλουν στην συνολική αποδοτικότητα του συστήματος.

- *Ποιότητα Υπηρεσίας*: Η συγκεκριμένη μετρική βελτιστοποίησης αναφέρεται στο επίπεδο της παρεχόμενης υπηρεσίας από την άποψη παραγόντων όπως η αξιοπιστία, η διαθεσιμότητα και η απόδοση. Η διασφάλιση ότι οι εργασίες που εκφορτώνονται πληρούν τις επιθυμητές απαιτήσεις ποιότητας υπηρεσίας είναι απαραίτητη για την επιτυχία των ρομποτικών εφαρμογών, ιδίως εκείνων που εκτελούν κρίσιμες εργασίες.

- *Επεκτασιμότητα*: Η συγκεκριμένη μετρική βελτιστοποίησης αναφέρεται στην ικανότητα του computational offloading να διαχειρίζεται έναν αυξανόμενο αριθμό ρομποτικών συσκευών ή εργασιών. Η διασφάλιση ότι η στρατηγική για το computational offloading που επιλέγεται παραμένει αποτελεσματική και αποδοτική καθώς το σύστημα κλιμακώνεται είναι σημαντική για εύρωστες και κλιμακούμενες ρομποτικές εφαρμογές.

- *Χρόνος απόφασης για Computational Offloading*: Η συγκεκριμένη μετρική βελτιστοποίησης αναφέρεται στον χρόνο που απαιτείται για την λήψη απόφασης σχετικά με το αν μια εργασία θα πρέπει να μεταφερθεί ή να επεξεργαστεί τοπικά. Η ελαχιστοποίηση του χρόνου λήψης αποφάσεων βοηθά στην ταχεία προσαρμογή στις μεταβαλλόμενες συνθήκες και απαιτήσεις.

Η τελική απόφαση για την εκφόρτωση των υπολογισμών σχετίζεται συνήθως με την βελτιστοποίηση μίας ή ενός συνδυασμού μετρικών βελτιστοποίησης από αυτές που αναλύθηκαν προηγουμένως.

Για παράδειγμα, στο [11] οι συγγραφείς πρότειναν ένα προσαρμοστικό σύστημα για την παράταση του χρόνου λειτουργίας μή επανδρομένων αεροσκαφών με αξιόπιστο σύστημα επικοινωνίας. Η αρχιτεκτονική του συστήματος αυτού αποτελείται από τρεις κύριες μονάδες: i) μία μονάδα πρόβλεψης του χρόνου απόκρισης, ii) μία μονάδα εκφόρτωσης εργασιών και iii) μία μονάδα απομακρυσμένης εκτέλεσης εργασιών. Η μονάδα πρόβλεψης χρόνου απόκρισης και η μονάδα εκφόρτωσης εργασιών εκτελούνται στο σύστημα του μή επανδρομένου αεροσκάφους ενώ η μονάδα απομακρυσμένης εκτέλεσης εργασιών εκτελείται στο επίγειο σύστημα ελέγχου. Η μονάδα πρόβλεψης χρόνου απόκρισης λαμβάνει πληροφορίες σχετικά με τους διαθέσιμους πόρους και τους παράγοντες των ασύρματων συνθηκών για να αποφασίσει την εκφόρτωση υπολογισμών ή την τοπική εκτέλεση της εργασίας. Σε αυτήν την μονάδα λαμβάνονται υπόψιν τρεις τύποι καθυστέρησης, συμπεριλαμβανομένης της καθυστέρησης επικοινωνίας, της καθυστέρησης πρόσβασης εισόδου/εξόδου και της καθυστέρησης εκτέλεσης εργασιών, για την επίτευξη ακριβών αποτελεσμάτων. Η μονάδα εκφόρτωσης εργασιών αποτελείται από μια υπηρεσία διαχείρισης απομακρυσμένων εργασιών, μία υπηρεσία δικτύου για απομακρυσμένους υπολογισμούς εργασιών και μία υπηρεσία επικοινωνίας δεδομένων. Η μονάδα απομακρυσμένης εκτέλεσης εργασιών είναι υπεύθυνη για την εκτέλεση των εργασιών στο επίγειο σύστημα ελέγχου και την απόκριση των αποτελεσμάτων εκτέλεσης στο σύστημα του μη επανδρομένου αεροσκάφους. Επιπλέον η μονάδα αυτή ενημερώνει την μονάδα πρόβλεψης του χρόνου απόκρισης σχετικά με τους διαθέσιμους πόρους του επίγειου συστήματους ελέγχου. Το σύστημα υλοποιείθηκε σε ένα πραγματικό μή επανδρομένο αεροσκάφους και σε έναν φορητό υπολογιστή ως σύστημα επίγειου ελέγχου. Τα αποτελέσματα των πειραμάτων που διεξήχθησαν έδειξαν ότι το προτεινόμενο σύστημα, εκτελώντας εκφόρτωση υπολογισμών, βοηθά στην βελτίωση του χρόνου λειτουργίας ενός μη επανδρομένου αεροσκάφους βελτιώνοντας επίσης και τον χρόνο απόκρισης.

Στο [21] οι συγγραφείς παρουσίασαν μία αρχιτεκτονική εκφόρτωσης εργασιών και ελέγχου για την ανάλυση

βίντεο από μη επανδρομένα αεροσκάφη. Η προτεινόμενη αρχιτεκτονική επιτρέπει την αντιστάθμιση μεταξύ της καθυστέρησης επεξεργασίας και της απόδοσης των αλγορίθμων ανάλυσης βίντεο που χρησιμοποιούνται στα μή επανδρομένα αεροσκάφοι. Η αρχιτεκτονική αποτελείται από μία στρατηγική λήψης αποφάσεων, η οποία αποφασίζει για τις κατάλληλες θέσεις όπου πρέπει να εκτελεστούν οι αυτόνομες λειτουργίες. Μέσα σε αυτό το πλαίσιο, οι αυτόνομες λειτουργίες μπορούν να εκτελούνται ξεχωριστά σε διαφορετικούς υπολογιστικούς κόμβους όπως στο Edge ή στο Cloud. Τα αποτελέσματα των πειραμάτων που διεξήχθησαν δείχνουν ότι η προτεινόμενη αρχιτεκτονική μπορεί να επιλέξει μία κατάλληλη τοπολογία και να κατανείμει σωστά τους πόρους σύμφωνα με τις απαιτήσεις που υπάρχουν σε καθυστέρηση και εύρος ζώνης.

Στο [27] οι συγγραφείς πρότειναν μία συγκεκριμένη αρχιτεκτονική για την εκτέλεση ρομποτικών εφαρμογών σχετικών με τον σχεδιασμό διαδρομής και τον εντοπισμό θέσης. Η αρχιτεκτονική αυτή αποτελείται απο κινητά ρομπότ τα οποία επικοινωνούν με έναν Edge Server για απομακρυσμένους υπολογισμούς ενώ συμπεριλαμβάνει επίσης τρεις διακοπτικούς μηχανισμούς: i) μηχανισμό επιλογής αισθητήρα για τον έλεγχο της αβεβαιότητας στην εκτίμηση της στάσης του ρομπότ, ii) μηχανισμό εκφόρτωσης του αλγορίθμου εκτίμησης της στάσης του ρομπότ ανάλογα με τον χρόνο εκτέλσης του και iii) μηχανισμό εκφόρτωσης του αλγορίθμου σχεδιασμού διαδρομής με βάση το κόστος της τροχιάς. Η αξιολόγηση της προτεινόμενης αρχιτεκτονικής πραγματοποιήθηκε με την χρήση ενός ρομπότ AlphaBot και ενός Edge Server. Η συγκεκριμένη αρχιτεκτονική επιτυγχάνει ακριβέστερη πλοήγηση του ρομπότ σε σχέση με την περίπτωση της αποκλειστικής τοπικής εκτέλεσης των εφαρμογών σχεδιασμού διαδρομής και εντοπισμού θέσης χωρίς όμως να πληρώνει το τίμημα ενός μεγαλύτερου χρόνου ολοκλήρωσης, όπως συμβαίνει στην περίπτωση όπου οι αλγόριθμοι εκτελούνται αποκλειστικά σε απομακρυσμένους πόρους.

Παρόμοια, οι συγγραφείς των [12], [19], [15], [16] σχεδίασαν κατάλληλα το σύστημα κατανομής των πόρων τους ώστε να επιτύχουν μείωση του συνολικού χρόνου ολοκλήρωσης της ρομποτικής εργασίας.

Στο [25] οι συγγραφείς σχεδίασαν μία αρχιτεκτονική τεσσάρων επιπέδων για την εκφόρτωση ακριβών υπολογιστικά εργασιών που σχετίζονται με το πρόβλημα του ταυτόχρονου εντοπισμού θέσης και χαρτογράφησης (SLAM): το επίπεδο του ρομπότ, το Edge επίπεδο, το Fog επίπεδο και το Cloud. Τα ρομπότ ήταν υπεύθυνα για την συλλογή πληροφοριών και για την εκτέλεση ορισμένων βασικών λειτουργιών. Το Edge επιπέδο ήταν υπεύθυνο για την συλλογή δεδομένων από τα ρομπότ και την εκτέλεση ορισμένων εργασιών υπολογιστικής όρασης. Το Fog επίπεδο ήταν υπεύθυνο για την διαχείριση της διασύνδεσης μεταξύ των Edge πυλών αλλά και την σύνδεσης των ρομπότ με τις πύλες αυτές. Τέλος το Cloud προσέφερε στους διαχειριστές ή στους τελικούς χρήστες την δυνατότητα οπτικοποίησης αλλά και εργαλεία ελέγχου για την παρακολούθηση των ρομπότ. Μέσα από τα πειράματα τους, οι συγγραφείς κατέδειξαν το όφελος της αρχιτεκτονικής τους στην μείωση της κατανάλωσης ενέργειας των ρομπότ αλλά και στην μείωση της καθυστέρησης που παρατηρείται στις εφαρμογές εντοπισμού θέσης και χαρτογράφησης.

Στο [22] οι συγγραφείς πρότειναν μία αρχιτεκτονική για ρομπότ που επικοινωνούν με το Cloud με σκοπό την δυναμική εκφόρτωση υπολογισμών ώστε να διασφαλίζεται η απαιτούμενη ποιότητα υπηρεσίας εφαρμογών που προορίζονται για "έξυπνες πόλεις". Στην εργασία αυτή, η απόφαση εκφόρτωσης διαμορφώρθηκε με τέτοιο τρόπο ώστε να βελτιστοποιήσει την κατανάλωση ενέργειας η οποία περιορίζεται από τον ανεκτό χρόνο ολοκλήρωσης. Προκειμένου να επιτύχουν τον στόχο τους, οι συγγραφείς πρότειναν μία προσέγγιση βασισμένη σε έναν γενετικό αλγόριθμο για την επίλυση του προβλήματος. Αξιολόγησαν την λύση τους μέσω προσομοίωσης μετρώντας το μέσο σκορ καταλληλότητας και το χαμηλότερο σκορ καταλληλότητας που πέτυχε ο γενετικός αλγόριθμος. Απέδειξαν επίσης ότι η λύση τους μπορεί να επιτύχει ελάχιστη κατανάλωση ενέργειας. Εκτός αυτού αξιολόγησαν την προσαρμοστικότητα της πρότασης τους στην αλλαγή του εύρους ζώνης. Ωστόσο καθώς η διακύμανση στο εύρος ζώνης του δικτύου δεν μπορούσε να προβλεφθεί κατά την διαδικασία λήψης απόφασης για την εκφόρτωση, το σύστημα σχεδιάστηκε με σταθερή τιμή εύρους ζώνης, κάτι το οποίο απέχει αρκετά από πραγματικά σενάρια. Οι ίδιοι συγγραφείς στο [23] διατύπωσαν το πρόβλημα κίνησης και επικοινωνίας των ρομπότ που χρησιμοποιούνται σε εφαρμογές συντήρησης ενός "έξυπνου εργοστασίου" ως ένα πρόβλημα κοινής βελτιστοποίησης. Το συγκεκριμένο πρόβλημα λαμβάνει υπόψιν του τρεις παράγοντες: την απόφαση για την εκφόρτωση εργασιών, το σημείο πρόσβασης (που επιτρέπει την επικοινωνία με το Cloud), και την υπολογισμένη διαδρομή που έχει σχεδιαστεί για να κινηθεί το ρομπότ. Για τον λόγο αυτό, προσάρμοσαν τον γενετικό αλγόριθμο προκειμένου να βελτιστοποίησει την κατανάλωση ενέργειας του ρομπότ η οποία περιορίζεται από μία επιβαλλόμενη καθυστέρηση. Πρότειναν μία κωδικοποίηση χρωμοσωμάτων τριών επιπέδων για την αναπαράσταση της εκφόρτωσης εργασιών, της επιλογής σημείου πρόσβασης και του σχεδιασμού διαδρομής. Τελικώς απέδειξαν μέσω προσομοίωσης την αποτελεσματικότητα της λύσης τους στην εύρεση της ελάχιστης κατανάλωσης ενέργειας. Τέλος στο [24] πρότειναν μία διαφορετική μελέτη για την βελτιστοποίηση της κατανάλωσης ενέργειας

σε ρομποτικές εφαρμογές που χρησιμοποιούν το Cloud. Η καινοτομία της συγκεκριμένης εργασίας έγγυται στην εξέταση της κινητικότητας του ρομπότ, της μεταβλητότητας του εύρους ζώνης και της μεταφόρτωσης από ρομπότ σε ρομπότ της διαδικασίας λήψης αποφάσης. Κατά συνέπεια, διατύπωσαν το πρόβλημα τους ως ένα πρόβλημα κοινής βελτιστοποίησης το οποίο επιλύθηκε με την ανάπτυξη ενός γενετικού αλγορίθμου τεσσάρων επιπέδων. Η αξιολόγηση των επιδόσεων της συγκεκριμένης εργασίας επικεντρώθηκε κυρίως στην μέτρηση της ελάχιστης κατανάλωσης ενέργειας.

Στο [6] οι συγγραφείς παρουσιάσαν μια αρχιτεκτονική που επιτρέπει την εκφόρτωση εργασιών τόσο στο Cloud όσο και σε άλλα ρομπότ. Για τον σκοπό αυτό, πρότειναν έναν γενετικό αλγόριθμο ο οποίος λαμβάνει υπόψιν την ενέργεια όλων των ρομπότ για την κατανομή των εργασιών. Ο προτεινόμενος αλγόριθμος εξασφαλίζει ισορροπημένη ανάθεση εργασιών μεταξύ των ρομπότ για να αποφευχθεί η εξάντληση της μπαταρίας ορισμένων ρομπότ, τηρώντας παράλληλα έναν ανεκτό χρόνο ολοκλήρωσης. Μέσω προσομοίωσης απέδειξαν την αποτελεσματικότητα της προσέγγισης τους στην εξισορρόπηση της κατανάλωσης ενέργειας των ρομπότ μετρώντας τέσσερις μετρικές απόδοσης: χρόνος λήψης απόδοσης, ακρίβεια απόφασης, διάρκεια ζωής του δικτύου και τυπική απόκλιση της κατανάλωσης ενέργειας. Στο [7] οι ίδιοι συγγραφείς πρότειναν μία λύση για την βελτιστοποίηση της κατανάλωσης ενέργειας και του χρόνου ολοκλήρωσης σε ένα ρομποτικό μοντέλου που χρησιμοποιεί το Cloud. Παρουσίασαν τα ρομπότ ως ένα συνδεδομένο κυρίαρχο σύνολο (Connected Dominating Set ή CDS) για την διατήρηση ισορροπημένης κατανάλωσης ενέργειας μεταξύ τους. Στην συνέχεια χρησιμοποίησαν τον αλγόριθμο Dijkstra αλλά και τον γενετικό αλγόριθμο για να βρουν την βέλτιστη στρατηγική εκφόρτωσης. Κατά την αξιολόγηση των επιδόσεων, αξιολόγησαν αρχικά την επίδραση του αριθμού και της αραιότητας των ρομπότ στην κατασκευή του CDS. Στην συνέχεια αξιολόγησαν την απόδοση της προσέγγισης τους στην επίτευξη βέλτιστων αποτελεσμάτων.

Τεχνικές Βελτιστοποίησης

Η επιλογή μίας αποτελεσματικής τεχνικής για την λήψη απόφασης σχετικά με το Computational Offloading παίζει καθοριστικό ρόλο στην τελική του απόδοση και στην εξασφάλιση ότι θα επιτευχθεί το προσδοκόμενο αποτέλεσμα. Ανάλογα με την τεχνική βελτιστοποίησης που επιλέγεται να χρησιμοποιηθεί προκύπτουν και διαφορετικές αποφάσεις σχετικά με την χρονική στιμγή που θεωρείται κατάλληλη προκειμένου να πραγματοποιηθεί η εκφόρτωση υπολογισμών. Δυστυχώς δεν υπάρχουν ενιαίες προσεγγίσεις που να ταιριάζουν σε όλα τα προβλήματα εκφόρτωσης και το κάθε πρόβλημα πρέπει να εξετάζεται ξεχωριστά. Οι πιο συχνές τεχνικές βελτιστοποίησης που χρησιμοποιούνται σε ρομποτικές εφαρμογές είναι οι ακόλουθες:

- *Ευρετικές Μέθοδοι*: Οι μέθοδοι αυτοί αποτελούνται από ένα σύνολο κανόνων που έχουν σχεδιαστεί για την επίλυση ενός συγκεκριμένου προβλήματος. Συνήθως προσαρμόζονται σε ένα συγκεκριμένο πρόβλημα και δεν λειτουργούν καλά σε άλλα προβλήματα ενώ συχνα είναι άπλειστες καθώς μπορεί να εγκλωβιστούν σε ένα τοπικό βέλτιστο καθώς αναζητούν ένα ολικό βέλτιστο. Το κύριο πλεονέκτημα των ευρετικών μεθόδων είναι πως αποτελούν απλούς αλγορίθμους που έχουν επινοηθεί για την επίλυση του εκάστοτε προβλήματος έχοντας χαμηλό κόστος εκτέλεσης. Επιπλέον, δεν απαιτούν εξιδεικευμένα εργαλεία βελτιστοποίησης για την υλοποίηση τους και έτσι μπορούν να υλοποιηθούν εύκολα σε οποιαδήποτε γλώσσα προγραμματισμού. Για τους λόγους αυτούς, οι ευρετικές μέθοδοι εφαρμόζονται πολύ συχνά σε προβλήματα εκφόρτωσης υπολογισμών.

- *Μέτα-ευρετικές Μέθοδοι*: Οι μέθοδοι αυτοί έχουν σχεδιαστεί για να επιλύουν ένα ευρύ φάσμα προβλημάτων. Εφαρμόζονται για προβλήματα που περιέχουν ένα ολικό βέλτιστο και πολλά τοπικά βέλτιστα. Γενικά, δεν είναι άπλειστες και διαθέτουν μηχανισμούς για να μην εγκλωβίζονται σε τοπικά βέλτιστα.

- *Μηχανική Μάθηση*: Οι μέθοδοι που βασίζονται στην Μηχανική Μάθηση στοχεύουν στην βελτιστοποίηση ενός κριτηρίου απόδοσης με βάση παρατηρήσεις ή δεδομένα. Η Βαθιά Ενισχυτική Μάθηση αποτελεί μία από τις πιο ευρέως χρησιμοποιούμενες τεχνικές μηχανικής μάθησης που εφαρμόζεται στην περίπτωση της εκφόρτωσης υπολογισμών στα ρομποτικά συστήματα. Επιπλέον, ιδαίτερα διαδεομένες είναι και οι μέθοδοι που βασίζονται στην Βαθιά Μάθηση.

- *Θεωρία Παιγνίων*: Η Θεωρία Παιγνίων είναι μία μαθηματική προσέγγιση που χρησιμοποιείται για την μελέτη της σύγκρουσης και της συνεργασίας μεταξύ δύο ή περισσότερων ορθολογικών παικτών που λαμβάνουν αποφάσεις. Χρησιμοποιεί μαθηματικά μοντέλα για την επίτευξη είτε μίας ισσοροπίας Nash είτε ενός ολικού βέλτιστου. Στην πραγματικότητα οι κύριες έννοιες της θεωρίας παιγνίων είναι εμπνευσμένες από τα συνηθισμένα παίγνια. Περιλαμβάνουν πολλούς παίκτες, μία σειρά κανόνων και διάφορες στρατηγικές.

Κάθε παίκτης ακολουθεί μία στρατηγική που παρουσιάζει όφελος για τον εαυτό του και ζημιά για τους άλλους παίκτες. Κάθε ενέργεια συνδέεται με μία "πληρωμή" η οποία καθορίζει αν η ενέργεια ήταν επωφελής ή επιζήμια για κάθε παίκτη. Κάθε παίκτης ενεργεί με στόχο την μεγιστοποίηση της "πληρωμής" του. Το παίγνιο καταλήγει σε μία σταθερή κατάσταση η οποία ονομάζεται ισορροπία Nash. Η ισορροπία Nash αποτελεί μία ομάδα στρατηγικών όπου κανένας παίκτης δεν μπορεί να επωφεληθεί τροποποιώντας την στρατηγική του ενώ οι στρατηγικές των υπολοίπων παικτών παραμένουν αμετάβλητες.

Στις ρομποτικές εφαρμογές επιλέγεται συνήθως μία ή συνδιασμός των παραπάνω τεχνικών βελτιστοποίησης για την λήψη της απόφασης σχετικά με την εκφόρτωση υπολογισμών.

Για παράδειγμα, στο [4] οι συγγραφείς θεώρησαν το πρόβλημα του computational offloading ως ένα κοινό πρόβλημα βελτιστοποίησης τριών μετρικών: της κατανάλωσης ενέργειας, της καθυστέρησης και του κόστους. Το συγκεκριμένο πρόβλημα βελτιστοποίησης λαμβάνει υπόψιν του τα χαρακτηριστικά των ρομπότ που συνδέονται μέσω δικτύου με το Cloud, καθώς και την ποιότητα της υπηρεσίας που απαιτείται από την ρομποτική εργασία, συμπεριλαμβανομένου του χρόνου απόκρισης. Μετέτρεψαν το πρόβλημα εκφόρτωσης υπολογισμών σε ένα πρόβλημα βελτιστοποίησης μεικτού ακέραιου γραμμικού προγραμματισμού, το οποίο έλυσαν προτείνοντας έναν ευρετικό αλγόριθμο. Μέσω προσομοίωσης απέδειξαν την αποτελεσματικότητα της λύσης τους στην βελτιστοποίηση των τριών αυτών μετρικών.

Στο [28] οι συγγραφείς ανέπτυξαν μία λύση υπολογιστικής εκφόρτωσης για μία ομάδα ρομπότ τα οποία είναι συνδεδεμένα σε έναν Edge Server προκειμένου να μπορούν να εκτελούν απομακρυσμένους υπολογισμούς. Ο κύριος στόχος της πρότασης τους ήταν η βελτιστοποίηση της κατανάλωσης ενέργειας και της καθυστέρησης που σχετίζεται με τον υπολογισμό. Για τον σκοπό αυτό ανέπτυξαν μία μέθοδο βαθιάς ενισχυτικής μάθησης που λαμβάνει υπόψιν της την κίνηση του ρομπότ προκειμένου να αποφασίσει το πότε χρειάζεται να πραγματοποιηθεί το computational offloading. Μέσω προσομοίωσης απέδειξαν την αποτελεσματικότητα της λύσης τους στην μείωση της κατανάλωσης ενέργειας και της καθυστέρησης.

Στο [20] οι συγγραφείς μοντελοποίησαν το πρόβλημα χρονοπρογραμματισμού εργασιών σε ένα ρομποτικό περιβάλλον νέφους ως μία Μαρκοβιανή αλυσίδα απόφασης με σκοπό την βελτιστοποίηση του χρόνου εκτέλεσης της εφαρμογής. Προκειμένου να λύσουν το πρόβλημα, πρότειναν μία προσέγγιση βαθιάς ενισχυτικής μάθησης. Πιο συγκεκριμένα, υπέθεσαν ότι το μέγεθος των δεδομένων εισόδου έχει άμεσο αντίκτυπο στον χρόνο εκτέλεσης της ρομποτικής εφαρμογής, όρισαν ένα σύνολο ενεργειών για κάθε χρονικό βήμα και εκπαίδευσαν ένα δίκτυο βαθιάς ενισχυτικής μάθησης. Η προτεινόμενη προσέγγιση επικυρώθηκε με την χρήση ενός ρομπότ και ενός διακομιστή νέφους AWS που εκτελούν συνεργατικά μία εφαρμογή σχεδιασμού διαδρομής. Απέδειξαν την αποτελεσματικότητα και την ακρίβεια του αλγορίθμου τους σε σχέση με άλλους αλγορίθμους.

Οι συγγραφείς στο [5] διατύπωσαν μαθηματικά το πρόβλημα του χρόνου ολοκκλήρωσης σε ρομποτικά συστήματα που επικοινωνούν με το Cloud ως μία διαδικασία απόφασης Markov (MDP). Χρησιμοποίησαν Βαθιά Ενισχυτική Μάθηση προκειμένου να λύσουν το πρόβλημα της εκφόρτωσης. Μέσω πειραμάτων απέδειξαν την αποτελεσματικότητα της μεθόδου βαθιάς ενισχυτικής μάθησης που χρησιμοποίησαν στην βελτιστοποίηση του χρόνου ολοκλήρωσης, συγκρίνοντας την με ορισμένες κοινές ευρετικές μεθόδους. Επιπλέον, τα πειράματα απέδειξαν ότι η προτεινόμενη λύση υπερτερεί έναντι ορισμένων ευρετικών μεθόδων στην μείωση της καθυστέρησης.

Οι συγγραφείς στο [8] χρησιμοποίησαν μία προσέγγση βασισμένη στην θεωρία παιγνίων για το πρόβλημα απόφασης σχετικά με την εκφόρτωση εργασιών. Μελέτησαν το πρόβλημα δρομολόγησης από κοινού με την εκφόρτωση υπολογισμών πολλαπλών βημάτων για ρομπότ που επικοινωνούν με το Cloud με σκοπό την βελτιστοποίηση της καθυστέρησης και της κατανάλωσης ενέργειας. Για να επιτύχουν τον στόχο τους πρότειναν μία κατανεμημένη λύση η οποία βασίζεται στην θεωρία παιγνίων.

Στο [26] οι συγγραφείς παρουσιάσαν μια προσέγγιση βασισμένη στην μηχανική μάθηση για την εκφόρτωση υπολογισμών και την παροχή πόρων για συστήματα αρχιτεκτονικής τριών επιπέδων που χρησιμοποιούν Edge και Cloud Computing. Συγκεκριμένα οι συγγραφείς χρησιμοποίησαν μαθησιακά αυτόματα ως την κύρια τεχνική για την λήψη αποφάσεων σχετικά με την εκφόρτωση υπολογισμών σύμφωνα με την οποία οι φόρτοι εργασίας αποφασίζεται να υποβληθούν σε επεξεργασία στους Edge ή Cloud servers. Επιπλέον ανέπτυξαν σε έναν Cloud Server τεχνικές βασισμένες στην βαθιά μάθηση για την κατανομή των πόρων, σύμφωνα με τις οποίες ένα μοντέλο LSTM χρησιμοποιήθηκε για την εκτίμηση του μελλοντικού φόρτου εργασίας και μία τεχνική ενισχυτικής μάθησης για την απόφαση της κατάλληλης κλιμάκωσης. Τα αποτελέσματα προσομοίωσης έδειξαν ότι η προτεινόμενη προσέγγιση συμβάλει στην βελτίωση της χρήσης της κεντρικής μονάδας επεξεργασίας και στην ελαχιστοποίηση τόσο της κατανάλωσης ενέργειας όσο και της καθυστέρησης εκτέλεσης.

Στο [10] οι συγγραφείς πρότειναν μια προσέγγιση εκφόρτωσης υπολογισμών και αυτόματης κλιμάκωσης η οποία χρησιμοποιεί έναν βρόγχο ελέγχου MAPE για να λαμβάνει αποφάσεις σχετικά με την τοποθεσία στην οποία πρέπει να υποβάλλονται σε επεξεργασία οι οντότητες μιας εφαρμογής προκειμένου να επιτευχθεί αποδοτική λειτουργία του συστήματος. Συγκεκριμένα μία οντότητα μπορεί να υποβληθεί σε επεξεργασία τοπικά σε μία συσκευή ή να εκφορτωθεί σε διακομιστές Fog ή Cloud σύμφωνα με ένα ποσοστό εκφόρτωσης που αναλύεται μέσω μίας Διαδικασίας Απόφασης Markov. Όταν λαμβάνεται η απόφαση της εκφόρτωσης για την εκτέλεση των οντοτήτων μίας εφαρμογής σε διακομιστές Fog ή Cloud, ένας αλγόριθμος βαθιάς ενισχυτικής μάθησης εφαρμόζεται για την εύρεση της καταλληλότερης θέσης. Προκειμένου να αξιολογηθεί η απόδοση της προτεινόμενης προσέγγισης χρησιμοποιήθηκαν διαφορετικές παράμετροι συμπεριλαμβανομένης της κατανάλωσης ενέργειας, του κόστους εκτέλεσης, της καθυστέρησης και της χρήσης πόρων του δικτύου. Τα αποτελέσματα έδειξαν ότι η προτεινόμενη προσέγγιση έχει καλύτερες επιδόσεις σε σχέση με άλλες προσεγγίσεις.

Chapter 3

Θεωρητικό Υπόβαθρο

3.1 ROS 2

3.1.1 Εισαγωγή

Το ROS 2 είναι μία πλατφόρμα λογισμικού για την ανάπτυξη ρομποτικών εφαρμογών υποστηρίζοντας ένα ευρύ φάσμα διαφορετικών ρομποτικών συστημάτων. Περιλαμβάνει ένα μεγάλο σύνολο αλληλένδετων υποσυστημάτων λογισμικού που χρησιμοποιούνται συνήθως για την ανάπτυξη των ρομποτικών εφαρμογών. Αυτό το οικοσύστημα λογισμικού χωρίζεται σε τρεις κύριες κατηγορίες:

- *Ενδιάμεσο Λογισμικό*: Το ενδιάμεσο λογισμικό περιλαμβάνει την επικοινωνία μεταξύ των υποσυστημάτων, από δικτυακά APIs μέχρι αναλυτές μηνυμάτων.
- *Αλγόριθμοι*: Το ROS 2 παρέχει πολλούς από τους αλγορίθμους που χρησιμοποιούνται συνήθως κατά την ανάπτυξη ρομποτικών εφαρμογών όπως αντίληψη, SLAM, σχεδιασμός κίνησης, έλεγχος κ.λ.π.
- *Προγραμματιστικά Εργαλεία*: Το ROS 2 παρέχει μία σειρά προγραμματιστικών εργαλείων και γραφικών που βοηθούν στην διαμόρφωση, την εκκίνηση, την οπτικοποίηση, τον εντοπισμό σφαλμάτων και την προσομοίωση των ρομποτικών εφαρμογών κατά την διάρκεια της ανάπτυξης τους. Υπάρχουν επίσης πολλά εργαλεία για την διαχείριση των πόρων της ρομποτικής εφαρμογής αλλά και την κατανομή τους.

3.1.2 Σχεδίαση

Ο σχεδιασμός του ROS 2 καθορίζεται από ένα σύνολο συγκεκριμένων αρχών και ένα σύνολο συγκεκριμένων απαιτήσεων.

Οι αρχές που υποστηρίζονται από το ROS 2 είναι οι εξής:

- *Κατανεμημένη επικοινωνία*: Όπως συμβαίνει και σε άλλα παρόμοια πολύπλοκα πεδία, τα προβλήματα στην ρομποτική αντιμετωπίζονται καλύτερα με μια κατανεμημένη προσέγγιση. Οι εργασίες χωρίζονται σε λειτουργικά ανεξάρτητα υποσυστήματα, όπως οδηγοί συσκευών για το υλικό (device drivers), συστήματα αντίληψης, συστήματα ελέγχου κ.λ.π. Κατά τον χρόνο εκτέλεσης τα συστήματα αυτά εκτελούν τις εργασίες τους αυτόνομα και ανταλλάσσουν δεδομένα μέσω ρητής επικοινωνίας.
- *Αφαιρετική προσέγγιση*: Για να ρυθμιστεί σωστά η επικοινωνία, πρέπει να καθοριστούν ορισμένες διεπαφές. Οι διεπαφές αυτές καθορίζουν την σημασιολογία των μηνυμάτων που ανταλλάσσονται. Μια τέτοια αφαιρετική προσέγγιση εξισορροπεί τα οφέλη της έκθεσης των λεπτομερειών ενός υποσυστήματος έναντι του κόστους της υπερβολικής προσαρμογής της υπόλοιπης εφαρμογής σε αυτό το υποσύστημα καθιστώντας έτσι δύσκολη την αντικατάσταση του με μία εναλλακτική λύση στο μέλλον. Η προσέγγιση αυτή οδηγεί σε ένα σύστημα διαλειτουργικών υποσυστημάτων τα οποία είναι ανεξάρτητα από συγκεκριμένους προμηθευτές υλικού ή λογισμικού.
- *Ασύγχρονη επικοινωνία*: Τα μηνύματα κοινοποιούνται μεταξύ των υποσυστημάτων ασύγχρονα, δημιουργώντας ένα σύστημα βασισμένο σε γεγονότα. Με αυτήν την προσέγγιση μία εφαρμογή μπορεί

να λειτουργεί σε πολλαπλά χρονικά πεδία που προκύπτουν από τον συνδυασμό φυσικών συσκευών με πλήθος στοιχείων λογισμικού καθένα από τα οποία μπορεί να έχει την δικιά του συχνότητα για την παροχή δεδομένων, την αποδοχή εντολών ή την σηματοδότηση συμβάντων.

- *Σπονδυλωτή μορφή*: Ο σχεδιαστικός στόχος του UNIX να "κάνει κάθε πρόγραμμα ένα πράγμα καλά" ακολουθείται και στο ROS 2. Η σπονδυλωτή μορφή επιβάλλεται σε πολλαπλά επίπεδα όπως σε API βιβλιοθηκών, στους ορισμούς μηνυμάτων, σε εργαλεία γραμμής εντολών ακόμα και στο ίδιο το οικοσύστημα λογισμικού. Το οικοσύστημα είναι οργανωμένο σε έναν μεγάλο αριθμό από ομοσπονδιακά πακέτα σε αντίθεση με μία έννοιά βάση κώδικα.

Οι σχεδιαστικές αυτές αρχές δεν είναι καθολικές και ούτε προκύπτουν χωρίς συμβιβασμούς. Η ασύγχρονη επικοινωνία μπορεί να κάνει πιο δύσκολο να επιτευχθεί ντετερμινιστική εκτέλεση. Για οποιοδήποτε μεμονωμένο καλά καθορισμένο πρόβλημα, είναι δυνατόν να κατασκευαστεί μία μονολιθική λύση ειδικού σκοπού που είναι πιο αποδοτική από υπολογιστική άποψη επειδή δεν περιλαμβάνει αφαιρετικές προσεγγίσεις ή κατανεμημένη επικοινωνία. Παρόλα αυτά η τήρηση των παραπάνω αρχών οδηγεί γενικότερα σε καλύτερα αποτελέσματα. Μία τέτοια προσέγγιση διευκολύνει την επαναχρησιμοποίηση κώδικα, την πραγματοποίηση δοκιμών στο λογισμικό αλλά την συνεργασία μεταξύ διαφορετικών ομάδων σε ένα κοινό έργο.

Το ROS 2 αποσκοπεί επίσης στην ικανοποίηση ορισμένων απαιτήσεων με βάση τις αρχές σχεδιασμού και τις ανάγκες των προγραμματιστών ρομποτικής:

- *Ασφάλεια*: Κάθε λογισμικό που αλληλεπιδρά με ένα δίκτυο πρέπει να περιλαμβάνει χαρακτηριστικά για την διασφάλιση της αλληλεπίδρασης αυτής απέναντι σε τυχαία και κακόβουλη διαχείριση. Το ολοκληρωμένο σύστημα ασφαλείας του ROS 2 περιλαμβάνει έλεγχο ταυτότητας, κρυπτογράφηση και έλεγχο πρόσβασης. Οι σχεδιαστές μπορούν να διαμορφώσουν το ROS 2 έτσι ώστε να ανταποκρίνεται στις δικές τους ανάγκες μέσω πολιτικών ελέγχου πρόσβασης που καθορίζουν ποιος μπορεί να επικοινωνήσει τι.

- *Ενσωματωμένα συστήματα*: Κατά γενικό κανόνα, ένα ρομπότ περιλαμβάνει αισθητήρες, ενεργοποιητές και άλλες περιφερειακές συσκευές. Αυτές οι συσκευές μπορούν να είναι σχετικά εξελιγμένες, περιλαμβάνοντας μικροελεγκτές που πρέπει να επικοινωνήσουν με τις κεντρικές μονάδες επεξεργασίας όπου εκτελείται το ROS 2. Ολόκληρο το λογισμικό του ROS 2 δεν αναμένεται να τρέχει σε μικρά ενσωματωμένα συστήματα αν και το ROS 2 θα πρέπει να διευκολύνει και να τυποποιεί την ενσωμάτωση των επεξεργαστών των ελεγκτών. Το micro-ros επιτρέπει την επανχρησιμοποίηση του ROS 2 σε ενσωματωμένα συστήματα.

- *Ποικίλα Δίκτυα*: Τα ρομπότ χρησιμοποιούνται σε διάφορα περιβάλλοντα δικτύωσης από ενσύρματα LAN για ρομποτικούς βραχίονες σε γραμμές συναρμολόγησης έως δορυφορικές συνδέσεις πολλαπλών βημάτων για πλανητικά οχήματα. Επιπλέον τα ρομπότ συχνά χρησιμοποιούν εσωτερικά δίκτυα για την σύνδεση διεργασιών εντός και μεταξύ των επεξεργαστών. Το ROS 2 παρέχει ποιότητα υπηρεσίας που ρυθμίζει τον τρόπο με τον οποίο τα δεδομένα ρέουν μέσω του συστήματος, όντας έτσι προσαρμοζόμενο στους περιορισμούς ενός δικτύου.

- *Υπολογισμός σε πραγματικό χρόνο*: Από τα ανθρωποειδή ρομπότ στα αυτόνομα αυτοκίνητα είναι σύνηθες για τις ρομποτικές εφαρμογές να περιλαμβάνουν απαιτήσεις εκτέλεσης σε πραγματικό χρόνο. Για να ικανοποιηθούν οι απαιτήσεις ασφάλειας και απόδοσης, ορισμένα μέρη ενός συστήματος πρέπει να εκτελούνται σε ντετερμινιστικά χρονικά διαστήματα. Το ROS 2 προσφέρει APIs για τους προγραμματιστές συστημάτων πραγματικού χρόνου έτσι ώστε να επιβάλλει περιορισμούς που αφορούν συγκεκριμένες εφαρμογές.

- *Ετοιμότητα προϊόντος*: Όταν ένα ρομπότ ξεφεύγει από το εργαστήριο και μεταφέρεται για εμπορική χρήση, εισάγονται ορισμένοι νέοι περιορισμοί. Το ROS 2 στοχεύει στην ικανοποίηση των απαιτήσεων του προϊόντος κατά την διάρκεια του σχεδιασμού, της ανάπτυξης και της εποπτείας του έργου. Κάτι τέτοιο επιτρέπει στο ROS 2 να τρέχει σε κρίσιμα για την ασφάλεια συστήματα όπως αυτόνομα οχήματα και βαριά μηχανήματα.

3.1.3 Πρότυπα Επικοινωνίας

Το ROS 2 παρέχει πρόσβαση σε ορισμένα πρότυπα επικοινωνίας. Τα πρότυπα αυτά είναι τα topics, τα services και τα actions τα οποία χρησιμοποιούν οι κόμβοι του συστήματος προκειμένου να επικοινωνούν μεταξύ τους (Εικόνα: 3.1.1).

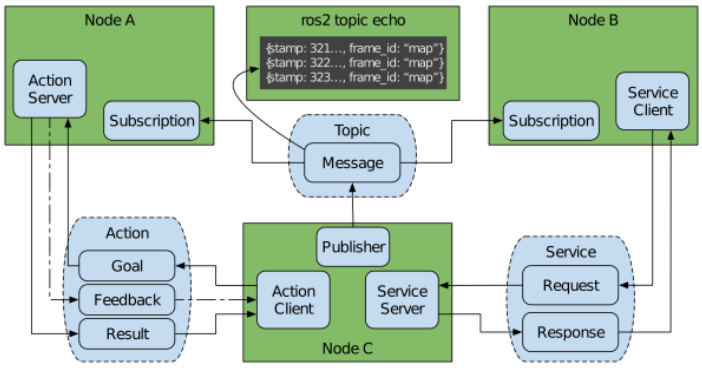

Figure 3.1.1: Διεπαφές κόμβων στο ROS 2: topics, services και actions

Ως κόμβος ορίζεται μια μεμονωμένη οντότητα λογισμικού στο ROS 2. Οι κόμβοι έχουν σχεδιαστεί με τέτοιο τρόπο ώστε η επαναχρησιμοποίηση του κώδικα να είναι πολύ εύκολη. Κάθε κόμβος εκτελεί μία συγκεκριμένη εργασία ή υπολογισμό στο ρομποτικό σύστημα όπως για παράδειγμα τον έλεγχο των κινητήρων των τροχών ή την δημοσίευση των δεδομένων αισθητήρων από ένα λέιζερ. Κάθε κόμβος μπορεί να στείλει και να λάβει δεδομένα μέσω των τριών προτύπων επικοινωνίας που αναφέρθηκαν προηγουμένως. Ένα πλήρες ρομποτικό σύστημα αποτελείται από πολλούς κόμβους οι οποίοι συνεργάζονται μεταξύ τους ενώ ένα ενιαίο εκτελέσιμο πρόγραμμα στο ROS 2 (π.χ. πρόγραμμα C++, πρόγραμμα Python κ.λ.π.) μπορεί να περιέχει έναν ή περισσότερους κόμβους.

Στην συνέχεια αναλύεται το κάθε πρότυπο επικοινωνίας ξεχωριστά.

- *Topics*: Όπως αναφέρθηκε και προηγουμένως, το ROS 2 διασπά τα πολύπλοκα ρομποτικά συστήματα σε πολλούς κόμβους. Τα topics λειτουργούν ως δίαυλος για την ανταλλαγή μηνυμάτων μεταξύ των κόμβων αυτών. Αποτελούν μία ασύγχρονη μετάδοση μηνυμάτων και είναι το πιο σύνηθες μοτίβο επικοινωνίας με το οποίο οι κόμβοι αλληλοεπιδρούν μεταξύ τους. Τα topics εισάγουν ένα μοντέλο δημοσίευσης-παραλαβής επιτρέποντας στους κόμβος να επικοινωνούν μεταξύ τους χωρίς να χρειάζεται να γνωρίζουν ο ένας την ταυτότητα ή την θέση του άλλου. Έτσι ένας κόμβος μπορεί να δημοσιεύσει μηνύματα σε ένα topic ενώ ένας άλλος κόμβος μπορεί να εγγραφεί σε αυτό το topic για να λάβει και να επεξεργαστεί τα μηνύματα αυτά. Η αρχικοποίηση ενός topic γίνεται με την δημιουργία ενός publisher για την αποστολή μηνυμάτων ή ενός subscriber για την λήψη μηνυμάτων που σχετίζονται με ένα συγκεκριμένο topic. Ο publisher ορίζεται από έναν κόμβο και είναι υπεύθυνος για την αποστολή μηνυμάτων σε ένα συγκεκριμένο topic. Ο publisher μπορεί να λειτουργήσει με συγκεκριμένη συχνότητα καθορίζοντας πόσο συχνά αποστέλλονται μηνύματα στο topic με το οποίο έχει συνδεθεί. Ο subscriber ορίζεται από έναν κόμβο και είναι υπεύθυνος για την λήψη μηνυμάτων από ένα συγκεκριμένο topic. Κάθε φορά που λαμβάνεται κάποιο μήνυμα από το topic στο οποίο έχει εγγραφεί, εκτελείται μία συνάρτηση η οποία ονομάζεται callback function. Η λειτουργία της συνάρτησης αυτής εξαρτάται από τον προγραμματιστή. Γενικότερα, οι publishers λειτουργούν ανεξάρτητα από τους subscribers παρέχοντας έναν αποσυνδεδεμένο και κλιμακούμενο ρυθμό επικοινωνίας. Ένας κόμβος μπορεί να δημοσιεύσει δεδομένα σε όσα topics επιθυμεί και ταυτόχρονα να δέχεται δεδομένα από όσα topics θέλει. Έτσι τα topics δεν χρειάζεται να αποτελούν ένα-προς-ένα επικοινωνία αλλά μπορεί να αποτελούν επίσης επικοινωνία ένα-προς-πολλά, πολλά-προς-ένα και πολλά-προς-πολλά.

- *Services*: Η ασύγχρονη επικοινωνία δεν αποτελεί πάντα το καταλληλότερο εργαλείο. Το ROS 2 παρέχει επίσης ένα διαφορετικό πρότυπο επικοινωνίας που βασίζεται στην αίτηση-απόκριση, γνωστό ως services. Ενώ τα topics επιτρέπουν στους κόμβους να εγγραφούν σε ροές δεδομένων και να λαμβάνουν συνεχής ενημερώσεις, τα services παρέχουν δεδομένα όταν καλούνται συγκεκριμένα από έναν κόμβο-πελάτη. Έτσι, η επικοινωνία βασισμένη στην αίτηση-απόκριση παρέχει εύκολη συσχέτιση δεδομένων μεταξύ μίας αίτησης και μίας απόκρισης η οποία μπορεί να είναι χρήσιμη όταν επιθυμούμε να εξασφαλίσουμε ότι μια εργασία ολοκληρώθηκε ή παραλήφθηκε. Το ROS 2 επιτρέπει επίσης στην διεργασία-πελάτη που κάνει το αίτημα να μην μπλοκάρεται κατά την διάρκεια ενός αιτήματος περιμένοντας την απάντηση.

- *Actions*: Ένα επιπλέον μοτίβο επικοινωνίας στο ROS 2 είναι τα actions. Τα actions αποτελούν ασύγχρονες διεπαφές επικοινωνίας προσανατολισμένες στην επίτευξη ενός στόχου ενώ περιλαμβάνουν τρία στοιχεία:

ένα αίτημα, μία περιοδική ανατροφοδότηση και μία απάντηση. Η λειτουργία τους είναι παρόμοια με αυτή των services παρέχοντας επιπλέον την δυνατότητα ακύρωσης τους ενώ βρίσκονται ακόμα σε εξέλιξη. Παρέχουν επίσης περιοδικά μία ανατροφοδότηση σε αντίθεση με τα services που επιστρέφουν απλώς μία απάντηση όταν ολοκληρώσουν την εκτέλεση τους. Αυτό το πρότυπο επικοινωνίας χρησιμοποιείται κυρίως σε εργασίες μεγάλης διάρκειας όπως η αυτόνομη πλοήγηση ή ο χειρισμός ενός ρομποτικού βραχίονα αν και έχει και άλλες ποικίλες χρήσεις. Όπως τα services έτσι και τα actions δεν μπλοκάρουν την εκτέλεση ενός κόμβου.

3.2 Gazebo

Το Gazebo είναι ένα τρισδιάστατο λογισμικό προσομοίωσης ρομπότ που επιτρέπει στους χρήστες να προσομοιώνουν και να απεικονίζουν την συμπεριφορά των ρομπότ σε ένα εικονικό περιβάλλον. Χρησιμοποιείται ευρέως στην κοινότητα της ρομποτικής για την κατασκευή πρωτοτύπων, την δοκιμή και την επικύρωση ρομποτικών αλγορίθμων. Το Gazebo δίνει την δυνατότητα στους χρήστες του να δημιουργούν λεπτομερείς προσομοιώσεις ρομποτικών συστημάτων, συμπεριλαμβανομένων των ρομπότ, των αισθητήρων και των αλληλεπιδράσεων τους με το περιβάλλον. Επιπλέον χρησιμοποιείται ευρέως για την δοκιμή και αποσφαλμάτωση ρομποτικών αλγορίθμων πριν αυτοί χρησιμοποιηθούν σε πραγματικά ρομποτικά συστήματα. Τέλος αποτελεί μία πολύ χρήσιμη πλατφόρμα για την εκπαίδευση μοντέλων μηχανικής μάθησης για ρομποτικές εφαρμογές.

Στην συνέχεια αναλύουμε ορισμένα από τα κύρια συστατικά του Gazebo:

- *ODE(Open Dynamics Engine)*: Το Gazebo χρησιμοποιεί την ODE ως την προεπιλεγμένη μηχανή προσομοίωσης παρέχοντας μία ρεαλιστική προσομοίωση για την δυναμική των άκαμπτων σωμάτων, τις συγκρούσεις και τους περιορισμούς των αρθρώσεων.

- *Αισθητήρες*: Το Gazebo υποστηρίζει διάφορους αισθητήρες που χρησιμοποιούνται συνήθως στην ρομποτική όπως κάμερες, λέιζερ και αισθητήρες μέτρησης βάθους, επιτρέποντας στους χρήστες να προσομοιώνουν δεδομένα αισθητήρων και να δοκιμάζουν αλγορίθμους αντίληψης. Επιπλέον οι χρήστες μπορούν να εισάγουν ρεαλιστικά μοντέλα θορύβου αισθητήρων για την προσομοίωση δεδομένων αισθητήρων του πραγματικού κόσμου.

- *Ενσωμάτωση ROS 2*: Το Gazebo είναι στενά συνδεδεμένο με το ROS 2 επιτρέποντας την απρόσκοπτη επικοινωνία μεταξύ του περιβάλλοντος προσομοίωσης και των ρομποτικών εφαρμογών που έχουν αναπτυχθεί σε ROS 2. Επιπλέον οι χρήστες μπορούν να εισάγουν τρισδιάστατα μοντέλα ρομπότ που έχουν δημιουργηθεί σε εργαλεία όπως το URDF (Unified Robot Description Format) για την προσομοίωση συγκεκριμένων ρομποτικών συστημάτων.

- *Περιβάλλον*: Το Gazebo περιλαμβάνει έναν γραφικό επεξεργαστή κόσμου που επιτρέπει στους χρήστες να δημιουργούν και να προσαρμόζουν εικονικά περιβάλλοντα. Οι χρήστες για παράδειγμα μπορούν να προσομοιοώσουν διάφορα εδάφη και να προσαρμόσουν τις συνθήκες φωτισμού για να μιμηθούν διάφορα σενάρια του πραγματικού κόσμου.

3.3 Rviz

Το Rviz είναι ένα εργαλείο απεικόνισης σχεδιασμένο να λειτουργεί απρόσκοπτα με το ROS 2, επιτρέποντας στους χρήστες να απεικονίζουν και να αλληλεπιδρούν με διάφορες πτυχές των ρομποτικών συστημάτων σε ένα τρισδιάστατο περιβάλλον. Είναι ιδιαίτερα χρήσιμο για τον εντοπισμό σφαλμάτων, τον έλεγχο και την κατανόηση της συμπεριφοράς των ρομπότ, παρέχοντας οπτικές αναπαραστάσεις των δεδομένων αισθητήρων και της κατάστασης του ρομπότ. Καθώς αποτελεί μέρος του ROS 2 οικοσυστήματος, εγγράφεται σε συγκεκριμένα topics προκειμένου να λαμβάνει και να απεικονίζει δεδομένα αισθητήρων, στάσεις του ρομπότ και άλλες σχετικές πληροφορίες. Το Rviz ακολουθεί έναν αρθρωτό σχεδιασμό επιτρέποντας στους χρήστες να προσθέτουν ή να αφαιρούν κομμάτια της οπτικοποίησης ανάλογα με τις ανάγκες τους. Οι χρήστες μπορούν επιπλέον να ρυθμίσουν την διάταξη και τον τρόπο της οπτικοποίησης με βάση τις απαιτήσεις τους. Ένα από τα σημαντικότερα προτερήματα του Rviz είναι ότι παρέχει οπτικοποίηση σε πραγματικό χρόνο επιτρέποντας στους χρήστες να παρατηρούν τις αλλαγές στο περιβάλλον και την κατάσταση του ρομπότ καθώς εξελίσσεται η προσομοίωση ή η λειτουργία του ρομπότ.

3.4 Ρομπότ διαφορικής οδήγησης

3.4.1 Ορισμός

Ένα ρομπότ διαφορικής οδήγησης αποτελεί ένα συγκεκριμένο είδος κινητού ρομπότ του οποίου η κίνηση βασίζεται σε δύο ξεχωριστά κινούμενους τροχούς που τοποθετούνται σε κάθε πλευρά του σώματος του ρομπότ. Έτσι, το συγκεκιρμένο ρομπότ μπορεί να αλλάξει την κατεύθυνση του μεταβάλλοντας την γωνιακή ταχύτητα περιστροφής του κάθε τροχού του και ως εκ τούτου δεν απαιτεί επισπρόσθετη στροφική κίνηση. Για παράδειγμα, εάν και οι δύο τροχοί περιστρέφονται με την ίδια γωνιακή ταχύτητα και προς την ίδια κατεύθυνση τότε το ρομπότ θα κινηθεί σε ευθεία γραμμή. Αντίθετα, εάν οι δύο τροχοί περιστρέφονται με την ίδια γωνιακή ταχύτητα αλλά προς αντίθετες κατεύθυνσεις, τότε το ρομπότ θα περιστραφεί γύρω από το κεντρικό σημείο του άξονα του. Διαφορετικά, ανάλογα με την γωνιακή ταχύτητα περιστροφής και την κατεύθυνση περιστροφής του κάθε τροχού, το κέντρο περιστροφής μπορεί να πέσει οπουδήποτε στην ευθεία γραμμή που ορίζεται από τα σημεία επαφής των τροχώς με το επίπεδο.

Σχηματικά έχουμε:

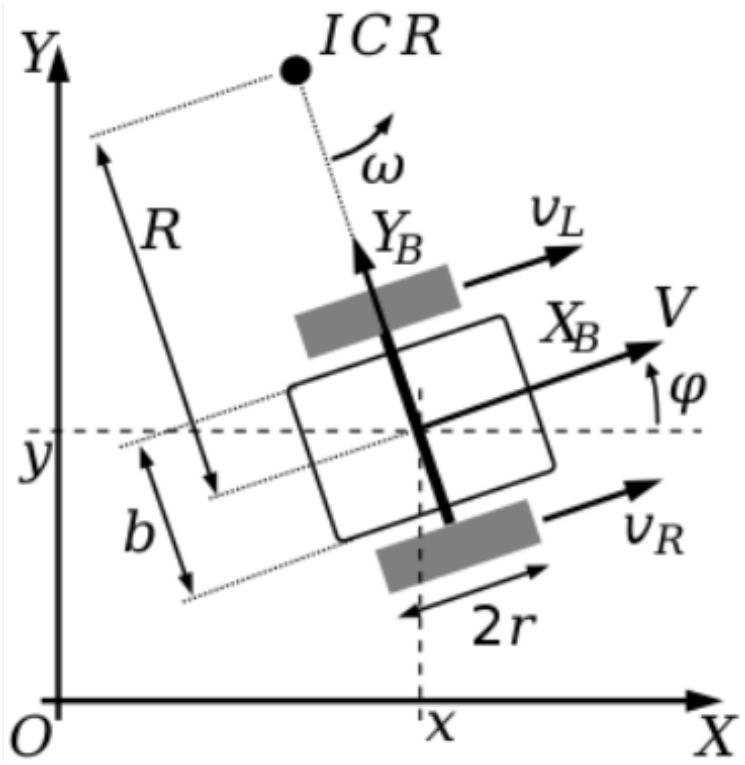

Figure 3.4.1: Ρομπότ Διαφορικής Οδήγησης

όπου:

- το u αποτελεί την γραμμική ταχύτητα κίνησης του ρομπότ.
- το w αποτελεί την γωνιακή ταχύτητα περιστροφής του ρομπότ.
- τα w_r και w_l αποτελούν την γωνιακή ταχύτητα περιστροφής του δεξιού και αριστερού τροχού του ρομπότ αντίστοιχα.
- τα u_r και u_l αποτελούν την γραμμική ταχύτητα του σημείου επαφής με το έδαφος του δεξιού και αριστερού τροχού του ρομπότ αντίστοιχα.
- τα r και b αποτελούν την ακτίνα του κάθε τροχού και την απόσταση μεταξύ των σημείων επαφής των δύο τροχών με το έδαφος αντίστοιχα.

3.4.2 Ευθύ Κινηματικό Μοντέλο

Υποθέτωντας ότι οι τροχοί βρίσκονται ανα πάσα στιγμή σε επαφή με το έδαφος (δεν υπάρχει ολίσθηση), οι τροχοί κινούνται πάνω σε τόξα στο επίπεδο με τέτοιο τρόπο ώστε το όχημα να περιστρέφεται σε κάθε χρονική στιγμή γύρω από ένα σημείο, το οποίο αναφέρεται ως στιγμιαίο κέντρο περιστροφής (ICR). Κάτι τέτοιο φαίνεται σχηματικά και στο σχήμα 3.4.1. Έτσι, η ταχύτητα u_r του σημείου επαφής με το έδαφος του δεξιού τροχού και η ταχύτητα u_l του σημείου επαφής με το έδαφος του αριστερού τροχού οδηγεί σε περιστροφή του οχήματος κατά γωνιακή ταχύτητα w. Ως εκ τούτου, με βάση τον ορισμό της γωνιακής ταχύτητας προκύπτουν οι ακόλουθες σχέσεις:

$$\begin{aligned} u_r &= w \cdot (R + \frac{b}{2}) \\ u_l &= w \cdot (R - \frac{b}{2}) \end{aligned} \tag{3.4.1}$$

όπου R είναι η απόσταση μεταξύ του στιγμιαίου κέντρου περιστροφής ICR και του κέντρου του ρομπότ όπως φαίνεται και στο σχήμα 3.4.1

Λύνωντας τις δύο παραπάνω εξισώσεις ως προς w και R προκύπτουν οι ακόλουθες δύο εξισώσεις:

$$w = \frac{u_r - u_l}{b} \tag{3.4.2}$$

$$R = \frac{b}{2} \cdot \frac{u_r + u_l}{u_r - u_l} \tag{3.4.3}$$

Γνωρίζοντας ότι η γραμμική ταχύτητα του ρομπότ συνδέεται με την γωνιακή ταχύτητα του ρομπότ μέσω της σχέσης $u = w \cdot R$ προκύπτει ότι:

$$u = \frac{u_r + u_l}{2} \tag{3.4.4}$$

3.4.3 Έλεγχος του Ρομπότ Διαφορικής Οδήγησης

Στην πράξη αυτό που συμβαίνει είναι πως το ρομπότ λαμβάνει σε κάθε χρονική στιμγή t μία επιθυμητή γραμμική ταχύτητα κίνησης u καθώς και μία επιθυμητή γωνιακή ταχύτητα περιστροφής w. Το ρομπότ στην συνέχεια μετατρέπει τις ταχύτητες αυτές σε γωνιακές ταχύτητες περιστοφής w_r και w_l του αριστερού και δεξιού τροχού του, οι οποίες τελικώς δίνονται ως εντολές κίνησης στους ενεργοποιητές του.

Έτσι, ο έλεγχος ενός ρομπότ διαφορικής οδήγησης πραγματοποιείται ως εξής:

- Το ρομπότ λαμβάνει μία επιθυμητή γραμμική ταχύτητα κίνησης u καθώς και μία επιθυμητή γωνιακή ταχύτητα περιστροφής w.
- Οι ταχύτητες αυτές στην συνέχεια μεταφράζονται σε επιθυμητές γραμμικές ταχύτητες κίνησης u_r και u_l του σημείου επαφής με το έδαφος του δεξιού και αριστερού τροχού του ρομπότ. Για τον σκοπό αυτό οι σχέσεις 3.4.2 , 3.4.4 λύνονται ως προς u_r και u_l και έτσι προκύπτει το ακόλουθο αντίστροφο κινηματικό μοντέλο για το ρομότ διαφορικής οδήγησης:

$$u_r = u + \frac{b \cdot w}{2} \tag{3.4.5}$$

$$u_l = u - \frac{b \cdot w}{2} \tag{3.4.6}$$

- Έπειτα η επιθυμητή γραμμική ταχύτητα u_r και u_l του δεξιού και αριστερού τροχού αντίστοιχα μεταφράζονται σε επιθυμητή γωνιακή ταχύτητα w_R και w_L του κάθε τροχού μέσω των ακόλουθων σχέσεων:

$$w_r = \frac{u_r}{r} \tag{3.4.7}$$

$$w_l = \frac{u_l}{r} \tag{3.4.8}$$

όπου r η ακτίνα των τροχών του ρομπότ.

- Οι γωνιακές αυτές ταχύτητες αποτελούν τις εντολές κίνησης για τους τροχούς του ρομπότ.

3.4.4 Οδομετρία του Ρομπότ Διαφορικής Οδήγησης

Το ρομπότ σε κάθε χρονική στιγμή λαμβάνει μέσω των κωδικοποιητών του τις τρέχουσες γωνιακές ταχύτητες περιστροφής w_r και w_l του δεξιού και αριστερού τροχού του αντίστοιχα. Μέσω των δεδομένων αυτών αλλά και γνωρίζοντας την στάση του την προηγούμενη χρονική στιγμή, το ρομπότ μπορεί να υπολογίσει σε κάθε χρονική στιγμή την τρέχουσα θέση (x_1, x_2) και τον τρέχων προσανατολισμό του ϕ καθώς και την τρέχουσα γραμμική u και γωνιακή ταχύτητα w της κίνησης του. Η διαδικασία υπολογισμού των παραπάνω ποσοτήτων πραγματοποιείται σε κάθε χρονική στιγμή και δίνεται στην συνέχεια:

- Θεωρούμε ότι ρομπότ βρίσκεται στην θέση (x, y, ϕ) την χρονική στιγμή t-1 και μετά από χρονικό διάστημα t φτάνει στην θέση (x', y', ϕ'). Το χρονικό αυτό διάστημα αντιστοιχεί στο χρονικό διάστημα μεταξύ δύο διαδοχικών μετρήσεων των γωνιακών ταχυτήτων w_r και w_l του δεξιού και αριστερού τροχού του ρομπότ που προκύπτουν από τους κωδικοποιητές του. Κάτι τέτοιο φαίνεται σχηματικά στην εικόνα 3.4.2.

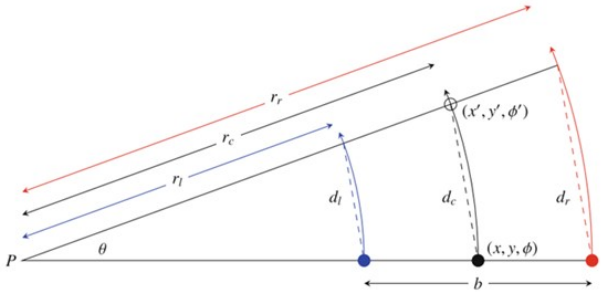

Figure 3.4.2: Μετακίνηση του ρομπότ από την στάση (x, y, ϕ) στην στάση (x', y', ϕ')

- Γνωρίζοντας την γωνιακή ταχύτητα περιστροφής $w_i (i = l, r)$ του αριστερού και δεξιού τροχού και θεωρώντας την σταθερή για το χρονικό διάστημα t μπορούμε να υπολογίσουμε την απόσταση που έχει διανύσει το σημείο επαφής του κάθε τροχού με το έδαφος στο χρονικό αυτό διάστημα με βάση τον ακόλουθο τύπο:

$$d_i = 2\pi R \omega_i t, i = l, r \tag{3.4.9}$$

όπου R η ακτίνα των τροχών του ρομπότ και t το χρονικό αυτό διάστημα.

- Καθώς όπως αναφέραμε και προηγουμένως θεωρούμε ότι η γωνιακή ταχύτητα περιστροφής του κάθε τροχού είναι σταθερή για το χρονικό διάστημα t, μπορούμε να θεωρήσουμε πως το σημείο επαφής του κάθε τροχού με το έδαφος καθώς και το κεντρικό σημείο του ρομπότ κινούνται πάνω σε ένα σταθερό τόξο ακτίνας r_l , r_r και r_c αντίστοιχα με κέντρο το σημείο P για το χρονικό αυτό διάστημα όπως φαίνεται και στο σχήμα 3.4.2. Με βάση την υπόθεση αυτή μπορεί να προκύψει η ακόλουθη σχέση που συνδέει τις αποστάσεις d_l , d_r και d_c που έχουν διανύσει οι δύο τροχοί και το κεντρικό σημείο του ρομπότ αντίστοιχα:

$$\theta = \frac{d_l}{r_l} = \frac{d_c}{r_c} = \frac{d_r}{r_r} \tag{3.4.10}$$

όπου θ είναι η γωνία που φαίνεται στο παραπάνω σχήμα 3.4.2.

- Χρησιμοποιώντας την σχέση 3.4.10 και εκτελώντας ορισμένους υπολογισμούς μπορούμε να εκφράσουμε την γωνία θ ως συνάρτηση των αποστάσεων d_l , d_r και της απόστασης $b = r_l - r_r$ μεταξύ των κέντρων

των δύο τροχών:

$$\begin{aligned} \theta r_r &= d_r \\ \theta r_l &= d_l \\ \theta r_r - \theta r_l &= d_r - d_l \\ \theta &= \frac{d_r - d_l}{r_r - r_l} \\ \theta &= \frac{d_r - d_l}{b} \end{aligned} \tag{3.4.11}$$

- Μέσω της σχέσης 3.4.10 και εκτελώντας ορισμένους υπολογισμούς μπορούμε επίσης να εκφράσουμε την απόσταση d_c που διένυσε το κέντρο του ρομπότ ως συνάρτηση των αποστάσεων d_l και d_r που διένυσαν τα σημεία επαφής των δύο τροχών με το έδαφος:

$$d_c = \theta r_c = \theta(\frac{r_l + r_c}{2}) = \frac{\theta}{2}(\frac{d_l}{\theta} + \frac{d_r}{\theta}) = \frac{d_l + d_r}{2} \tag{3.4.12}$$

- Θεωρώντας ότι η απόσταση d_c που διένυσε το κεντρικό σημείο του ρομπότ στο χρονικό διάστημα t είναι αρκετά μικρή, μπορούμε να προσεγγίσουμε το τόξο πάνω στο οποίο κινείται με μία ευθεία γραμμή η οποία ενώνει το αρχικό με το τελικό σημείο. Η ευθεία αυτή θα είναι κάθετη στην ακτίνα r_c όπως φαίνεται και στο ακόλουθο σχήμα:

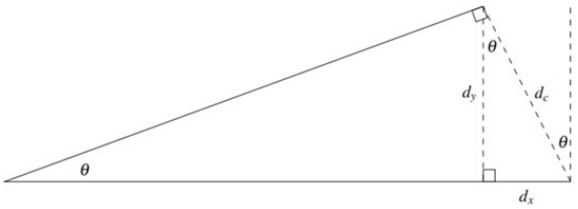

Figure 3.4.3: Προσέγγιση του τόξου με ευθεία

Με βάση την παραπάνω υπόθεση εύκολα προκύπτει όπως φαίνεται και στο παραπάνω σχήμα 3.4.3 πως η γωνία θ αποτελεί την επιπλέον γωνία περιστροφής του ρομπότ στο χρονικό διάστημα t. Θεωρώντας πως $d_c = (d_x, d_y)$ και μέσω απλής τριγωνομετρίας προκύπτουν οι ακόλουθες σχέσεις:

$$\begin{aligned} d_x &= -d_c sin\theta \\ d_y &= d_c cos\theta \end{aligned} \tag{3.4.13}$$

- Τελικώς, οι νέες συντεταγμένες του ρομπότ που προκύπτουν είναι οι ακόλουθες:

$$\begin{aligned} x' &= x - d_c sin\theta \\ y' &= y + d_c cos\theta \\ \phi' &= \phi + \theta \end{aligned} \tag{3.4.14}$$

όπου:

  - Το d_c συμβολίζει την απόσταση που διένυσε το κεντρικό σημείο του ρομπότ και υπολογίζεται απευθείας από την σχέση 3.4.12 ως $d_c = \frac{d_l + d_r}{2}$.
  - Η γωνία ϑ συμβολίζει την γωνία του τόξου πάνω στο οποίο κινείται το κεντρικό σημείο του ρομπότ και υπολογίζεται απευθείας από την σχέση 3.4.11 ως $\theta = \frac{d_r - d_l}{b}$.
  - Τα d_l, d_r συμβολίζουν την απόσταση που διένυσε ο αριστερός και δεξιός τροχός του ρομπότ αντίστοιχα και υπολογίζονται απευθείας από την σχέση 3.4.9 γνωρίζοντας την ταχύτητα περιστροφής w_r, w_l του κάθε τροχού που παρέχεται από τους κωδικοποιητές του.

- Επιπλέον υπολογίζονται η γραμμική ταχύτητας κίνησης u και η γωνιακή ταχύτητα περιστροφής w του ρομπότ με βάση τις ακόλουθες σχέσεις:

$$u = \frac{\sqrt{d_x^2 + d_y^2}}{t}$$
$$w = \frac{\theta}{t} \tag{3.4.15}$$

όπου τα d_x, d_y προκύπτουν από την σχέση 3.4.13.

Η παραπάνω διαδικασία υπολογισμού της νέας κατάστασης (x', y', ϕ') του ρομπότ και των ταχυτήτων του (u, w) με δεδομένη την προηγούμενη του κατάσταση (x, y, ϕ) και τις ταχύτητες περιστροφής w_l και w_r των δύο τροχών του πραγματοποιείται κάθε φορά που έρχεται μία νέα μέτρηση ταχυτήτων (w_l, w_r) από τους κωδικοποιητές του.

3.5 Εντοπισμός στάσης κινητού ρομπότ

3.5.1 Ορισμός του προβλήματος

Ο εντοπισμός της στάσης ενός κινητού ρομπότ ορίζεται ως το πρόβλημα του προσδιορισμού της στάσης του ρομπότ σε σχέση με ένα δεδομένο στατικό χάρτη του περιβάλλοντος στο οποίο κινείται. Συχνά αποκαλείται και εκτίμηση στάσης.

Η στάση ενός κινητού ρομπότ που λειτουργεί στο επίπεδο φαίνεται σχηματικά στην εικόνα 3.5.1.

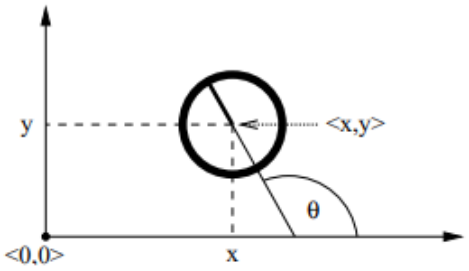

Figure 3.5.1: Η στάση του ρομπότ σε ένα καθολικό σύστημα συντεταγμένων

Όπως μπορούμε να παρατηρούμε, η στάση ενός κινητού ρομπότ αποτελείται από τις διδιάστατες επίπεδες συντεταγμένες του σε σχέση με ένα εξωτερικό πλαίσιο συντεταγμένων, καθώς και το γωνιακό προσανατολισμό του. Συμβολίζοντας τις δισδιάστατες επίπεδες συντεταγμένες του ρομπότ με x, y και τον προσανατολισμό του με θ, η στάση του ρομπότ περιγράφεται από το ακόλουθο διάνυσμα:

$$x_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.5.1}$$

Ο προσανατολισμός θ ενός ρομπότ αποκαλείται συχνά *κατεύθυνση ή κατεύθυνση πορείας*. Όπως φαίνεται και από την εικόνα 3.5.1, θεωρούμε δεδομένο ότι ένα ρομπότ με προσανατολισμό $\theta = 0$ είναι στραμμένο προς την κατεύθυνση του άξονα x. Αντίστοιχα ένα ρομπότ με προσανατολισμό $\theta = \frac{\pi}{2}$ είναι στραμμένο προς την κατεύθυνση του άξονα y. Η στάση ενός ρομπότ χωρίς τον προσανατολισμό του ονομάζεται συχνά και *θέση* του ρομπότ.

Ένας στατικός χάρτης του περιβάλλοντος αποτελεί μία αναπαράσταση που αποτυπώνει τα σταθερά στοιχεία του περιβάλλοντος που δεν αλλάζουν με την πάροδο του χρόνου. Η αναπαράσταση αυτή περιλαμβάνει εμπόδια, τοίχους, ορόσημα και άλλες κατασκευές που παραμένουν ακίνητες. Ο στατικός χάρτης είναι ζωτικής σημασίας

προκειμένου να μπορέσει ένα κινητό ρομπότ να πλοηγηθεί και να σχεδιάσει αποτελεσματικά την διαδρομή του. Μία κοινή μέθοδος για την αναπαράσταση ενός στατικού χάρτη είναι η διαίρεση του περιβάλλοντος σε ένα πλέγμα. Κάθε κελί στο πλέγμα αντιστοιχεί σε μία συγκεκριμένη περιοχή του περιβάλλοντος και η παρουσία ή αποχή ενός εμποδίου στην συγκεκριμένη περιοχή καταγράφεται. Η αναπαράσταση αυτή οδηγεί σε ένα δυαδικό πλέγμα όπου τα εμπόδια αντιπροσωπεύονται από κατειλημμένα κελιά και ο ελεύθερος χώρος από μή κατειλημμένα κελιά. Ο χάρτης του περιβάλλοντος συμβολίζεται με m και αναφέρεται σε αυτήν την δυαδική αναπαράσταση πλέγματος που αναλύθηκε προηγουμένως.

Σχεδόν όλες οι εργασίες στην ρομποτική απαιτούν γνώση της στάσης του ρομπότ. Το πρόβλημα του εντοπισμού της στάσης ενός κινητού ρομπότ μπορεί να θεωρηθεί ως ένα πρόβλημα μετασχηματισμού συντεταγμένων. Οι χάρτες περιγράφονται σε ένα καθολικό σύστημα συντεταγμένων που είναι ανεξάρτητο από την στάση του ρομπότ. Ο εντοπισμός της στάσης είναι η διαδικασία με την οποία καθορίζεται η αντιστοιχία ανάμεσα στο σύστημα συντεταγμένων του χάρτη και στο τοπικό σύστημα συντεταγμένων του ρομπότ. Η γνώση του μετασχηματισμού συντεταγμένων δίνει την δυνατότητα στο ρομπότ να εκφράσει την θέση των αντικειμένων που το ενδιαφέρουν στο δικό του πλαίσιο συντεταγμένων, κάτι το οποίο είναι προαπαιτούμενο για την πλοήγηση του ρομπότ. Έτσι η γνώση της στάσης $x_t = (x, y, \theta)^\intercal$ του ρομπότ επαρκεί για τον προσδιορισμό του μετασχηματισμού συντεταγμένων, με την προϋπόθεση ότι η στάση εκφράζεται στο ίδιο πλαίσιο συντεταγμένων με τον χάρτη.

Δυστυχώς στο πρόβλημα εντοπισμού στάσης ενός κινητού ρομπότ η στάση δεν μπορεί να γίνει αισθητή άμεσα. Και αυτό γιατί τα περισσότερα ρομπότ δεν διαθέτουν κάποιον αισθητήρα χωρίς θόρυβο για την μέτρηση της στάσης τους. Κατά συνέπεια τα ρομπότ πρέπει να συμπεράνουν την στάση τους από τα δεδομένα που τους παρέχουν οι αισθητήρες τους.

3.5.2 Ταξινόμηση των προβλημάτων εντοπισμού στάσης

Τα προβλήματα εντοπισμού στάσης ενός κινητού ρομπότ δεν έχουν όλα την ίδια δυσκολία. Στην συνέχεια θα περιγράψουμε σύντομα μία ταξινόμηση των προβλημάτων εντοπισμού στάσης. Η συγκεκριμένη ταξινόμηση χωρίζει τα προβλήματα εντοπισμού στάσης με βάση μερικές σημαντικές διαστάσεις που αφορούν την φύση του περιβάλλοντος και την αρχική γνώση που ενδέχεται να έχει το ρομπότ σε σχέση με το πρόβλημα εντοπισμού στάσης.

Τοπικός και καθολικός εντοπισμός στάσης

Τα προβλήματα εντοπισμού στάσης μπορούν να χαρακτηριστούν από το είδος της γνώσης που είναι διαθέσιμη αρχικά και κατά τον χρόνο εκτέλεσης. Διακρίνουμε τρεις τύπους προβλημάτων εντοπισμού στάσης με αυξανόμενο βαθμό δυσκολίας:

- *Παρακολούθηση στάσης*: Η παρακολούθηση στάσης υποθέτει ότι η αρχική στάση του ρομπότ είναι γνωστή. Ο εντοπισμός της στάσης του ρομπότ μπορεί να επιτευχθεί με την ενσωμάτωση του θορύβου στην κίνηση του ρομπότ. Η αβεβαιότητα της στάσης προσεγγίζεται συχνά από μία μονοκόρυφη κατανομή (π.χ. μία Γκαουσιανή κατανομή). Το πρόβλημα της παρακολούθησης στάσης είναι τοπικό πρόβλημα, αφού η αβεβαιότητα είναι τοπική και περιορισμένη στην περιοχή κοντά στην πραγματική στάση του ρομπότ.
- *Καθολικός εντοπισμός στάσης*: Στον καθολικό εντοπισμό στάσης η αρχική στάση του ρομπότ είναι άγνωστη. Το ρομπότ βρίσκεται αρχικά σε κάποιο σημείο του περιβάλλοντος του, αλλά δεν γνωρίζει την ακριβή στάση του. Οι μέθοδοι για τον καθολικό εντοπισμό στάσης δεν μπορούν να υποθέσουν ότι το σφάλμα της στάσης είναι φραγμένο. Έτσι οι μονοκόρυφες κατανομές για την προσέγγιση της αβεβαιότητας είναι συνήθως ακατάλληλες. Ο καθολικός εντοπισμός στάσης είναι πιο δύσκολος από την παρακολούθηση στάσης (στην πραγματικότητα εμπεριέχει το πρόβλημα της παρακολούθησης στάσης).
- *Πρόβλημα απαχθέντος ρομπότ*: Το πρόβλημα του απαχθέντος ρομπότ είναι μία ακόμα δυσκολότερη παραλλαγή του προβλήματος του καθολικού εντοπισμού στάσης. Κατά την διάρκεια της λειτουργίας του το ρομπότ μπορεί να «απαχθεί» και να τηλεμεταφερθεί σε κάποια άλλη θέση. Το συγκεκριμένο πρόβλημα είναι πιο δύσκολο από το πρόβλημα εντοπισμού στάσης με την έννοια ότι το ρομπότ ενδέχεται να πιστεύει ότι γνωρίζει που βρίσκεται ενώ αυτό δεν ισχύει. Στον καθολικό εντοπισμό στάσης το ρομπότ ξέρει ότι δεν γνωρίζει που βρίσκεται. Θα μπορούσε κάποιος να ισχυριστεί ότι στην πράξη η απαγωγή του ρομπότ συμβαίνει πολύ σπάνια. Ωστόσο η πρακτική σημασία του παραπάνω προβλήματος προκύπτει από την παρατήρηση ότι η απόλυτη επιτυχία δεν είναι εγγυημένη για τους περισσότερους κορυφαίους αλγορίθμους

εντοπισμού στάσης. Η δυνατότητα της ανάκαμψης από αποτυχίες είναι θεμελιώδης για ένα πραγματικό αυτόνομο ρομπότ. Η απαγωγή του ρομπότ ελέγχει τον αλγόριθμο εντοπισμού στάσης και μετράει την δυνατότητα του να ανακάμπτει από αποτυχίες του καθολικού εντοπισμού στάσης.

Στατικά και Δυναμικά Περιβάλλοντα

Μία δεύτερη διάσταση που επηρεάζει σημαντικά την δυσκολία του εντοπισμού στάσης είναι το περιβάλλον. Τα περιβάλλοντα μπορούν να είναι στατικά ή δυναμικά:

- *Στατικά Περιβάλλοντα*: Στα στατικά περιβάλλοντα η μοναδική μεταβλητή κατάσταση είναι η στάση του ρομπότ. Με άλλα λόγια, μόνο το ρομπότ κινείται στο στατικό περιβάλλον. Όλα τα υπόλοιπα αντικείμενα του περιβάλλοντος παραμένουν πάντα στην ίδια θέση.
- *Δυναμικά Περιβάλλοντα*: Στα δυναμικά περιβάλλοντα υπάρχουν και άλλα αντικείμενα εκτός από το ρομπότ, με θέση ή κατεύθυνση που μεταβάλλεται με τον χρόνο. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι αλλαγές που διατηρούνται σε σχέση με τον χρόνο, οι οποίες επηρεάζουν τις ενδείξεις περισσότερων από ένα αισθητήρων. Φυσικά οι αλλαγές που δεν είναι μετρήσιμες δεν έχουν σχέση με τον εντοπισμό στάσης, και εκείνες που επηρεάζουν μόνο μία μέτρηση αισθητήρα θεωρούνται θόρυβος. Ο εντοπισμός της στάσης σε δυναμικά περιβάλλοντα είναι αρκετά πιο δύσκολος από ότι σε στατικά περιβάλλοντα.

3.5.3 Πιθανοτική Εκτίμηση Στάσης

Όπως αναφέρθηκε και προηγουμένως, η στάση ενός κινητού ρομπότ δεν αποτελεί μια άμεσα μετρήσιμη ποσότητα. Αντίθετα, ένα ρομπότ πρέπει να στηρίζεται στους αισθητήρες του για να συλλέγει τέτοιες πληροφορίες. Οι αισθητήρες όμως δεν παρέχουν πλήρεις πληροφορίες για την στάση του και οι μετρήσεις τους αλλοιώνονται από θόρυβο όντας πολλές φορές ανακριβείς και αβέβαιες.

Λόγο της αβεβαιότητας αυτής, η εξέλιξη της στάσης x του ρομπότ αλλά και των μετρήσεων z που αυτό πραγματοποιεί μέσω των αισθητήρων του διέπεται από πιθανοτικούς νόμους. Πιο συγκεκριμένα, η στάση x_t την χρονική στιγμή t παράγεται στοχαστικά από την στάση x_{t-1} την χρονική στιγμή t-1. Επομένως είναι απαραίτητο να ορισθεί η κατανομή πιθανότητας από την οποία παράγεται η x_t. Στην πιο γενική της μορφή, η εμφάνιση της στάσης x_t εξαρτάται από όλες τις προηγούμενες στάσεις, μετρήσεις και ενέργειες ελέγχου που έχουν πραγματοποιηθεί. Άρα, ο πιθανοτικός νόμος που χαρακτηρίζει την εξέλιξη της στάσης μπορεί να δίνεται από μία κατανομή πιθανότητας της ακόλουθης μορφής:

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) \tag{3.5.2}$$

Μία υπόθεση που πραγματοποιείται συχνά προκειμένου οι αλγόριθμοι εκτίμησης στάσης να είναι υπολογιστικά εφικτοί είναι αυτή της υπό συνθήκης ανεξαρτησίας. Σύμφωνα με την συνθήκη αυτή, η στάση x_{t-1} θεωρείτε ένα επαρκές στατιστικό στοιχείο για όλες τις προηγούμενες ενέργειες ελέγχου και όλες τις προηγούμενες μετρήσεις μέχρι το συγκεκριμένο χρονικό σημείο. Με άλλα λόγια η στάση x_{t-1} αποτελεί μία επαρκή περίληψη όλων όσων έχουν συμβεί σε προηγούμενα χρονικά βήματα. Έτσι μόνο η ενέργεια ελέγχου u_t μας δίνει περεταίρω πληροφορίες αν γνωρίζουμε την στάση x_{t-1}. Ως αποτέλεσμα, η κατανομή πιθανότητας που περιγράψαμε παραπάνω μπορεί να απλοποιηθεί στην ακόλουθη μορφή:

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{3.5.3}$$

Η παραπάνω πιθανότητα ονομάζεται *πιθανότητα μεταβολής κατάστασης* και ορίζει τον τρόπο με τον οποίο εξελίσσεται η στάση του ρομπότ στον χρόνο ως συνάρτηση των ενεργειών ελέγχου u_t του ρομπότ.

Με τον ίδιο τρόπο μπορούμε να μοντελοποιήσουμε και την διαδικασία με την οποία παράγονται οι μετρήσεις. Θεωρώντας δηλαδή με την ίδια λογική την στάση x_t ως ένα πλήρες στατιστικό στοιχείο, η κατανομή πιθανότητας της μέτρησης z_t μπορεί να ορισθεί ως εξής:

$$p(z_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \tag{3.5.4}$$

Η πιθανότητα αυτή ονομάζεται *πιθανότητα μέτρησης* και ορίζει τον πιθανοτικό νόμο σύμφωνα με τον οποίο παράγονται οι μετρήσεις z από την στάση x του ρομπότ. Με άλλα λόγια δηλαδή θεωρούμε τις μετρήσεις ως θορυβώδεις προβολές της στάσης του ρομπότ.

Η πιθανότητα μεταβολής κατάστασης και η πιθανότητα μέτρησης περιγράφουν μαζί το δυναμικό στοχαστικό σύστημα του ρομπότ και του περιβάλλοντος του. Συμπερασματικά η στάση του ρομπότ x_t στην χρονική στιγμή t εξαρτάται στοχαστικά από την στάση x_{t-1} στη χρονική στιγμή t-1 και την ενέργεια ελέγχου u_t ενώ η μέτρηση z_t στην χρονική στιγμή t εξαρτάται στοχαστικά μόνο από την στάση του ρομπότ στην χρονική στιγμή t.

3.5.4 Πιθανότητα Μεταβολής Κατάστασης

Η πιθανότητα μεταβολής κατάστασης $p(x_t|u_t, x_{t-1})$ παίζει βασικό ρόλο στους αλγορίθμους εκτίμησης της στάσης του ρομπότ που θα αναλυθούν στην συνέχεια. Η κινηματική του ρομπότ έχει μελετηθεί κατά κύριο λόγο με αιτιοκρατική μορφή. Στην πραγματικότητα η πιθανότητα μεταβολής κατάστασης γενικεύει τις εξισώσεις της κινηματικής ενός ρομπότ ώστε να λαμβάνουν υπόψη τους το γεγονός ότι το αποτέλεσμα μίας ενέργειας ελέγχου είναι αβέβαιο, επειδή υπάρχει θόρυβος ελέγχου ή μη μοντελοποιημένα εξωγενή φαινόμενα. Με άλλα λόγια, το αποτέλεσμα μίας ενέργειας ελέγχου περιγράφεται με μία εκ των υστέρων πιθανότητα. Έτσι, τα αιτιοκρατικά μοντέλα των μηχανισμών δράσης ενός ρομπότ «πιθανοτικοποιούνται» με την προσθήκη μεταβλητών θορύβου, οι οποίες χαρακτηρίζουν τους τύπους της αβεβαιότητας που υπάρχει στην ρομποτική δράση. Τα μοντέλα που προκύπτουν υπόκεινται στις πιθανοτικές τεχνικές εκτίμησης κατάστασης όπως θα φανεί στους αλγορίθμους που θα αναλυθούν στην συνέχεια.

Όπως φαίνεται και από την εξίσωση 3.5.3, η πιθανότητα μεταβολής κατάστασης ορίζεται ως η υπό συνθήκη πυκνότητα πιθανότητας $p(x_t|u_t, x_{t-1})$, όπου τα x_t, x_{t-1} αποτελούν τις στάσεις του ρομπότ την χρονική στιγμή t και t-1 αντίστοιχα και το u_t αποτελεί μία εντολή κίνησης (ή ενέργεια ελέγχου). Το συγκεκριμένο μοντέλο περιγράφει την εκ των υστέρων κατανομή για τις κινηματικές καταστάσεις στις οποίες βρίσκεται το ρομπότ όταν εκτελέσει την εντολή κίνησης u_t στην κατάσταση x_{t-1}. Στις πρακτικές εφαρμογές το u_t παρέχεται κυρίως από την οδομετρία ενός ρομπότ. Έτσι στην συνέχεια αναλύεται ένα μοντέλο κίνησης που βασίζεται στην οδομετρία, το οποίο είναι και αυτό που χρησιμοποιείται στις περισσότερες πραγματικές ρομποτικές εφαρμογές για τον υπολογισμό της πιθανότητας μεταβολής κατάστασης.

Μοντέλο Κίνησης της Οδομετρίας

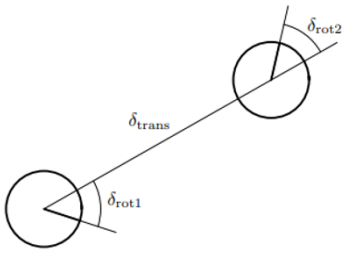

Figure 3.5.2: Μοντέλο κίνησης Οδομετρίας: Η κίνηση του ρομπότ στο χρονικό διάστημα $(t-1, t]$ προσεγγίζεται από μια περιστροφή δ_{rot1}, ακολουθούμενη από μια ευθεία μετατόπιση δ_{trans} και μια δεύτερη περιστροφή δ_{rot2}.

Το συγκεκριμένο μοντέλο κάνει χρήση των μετρήσεων της οδομετρίας ως βάση για τον υπολογισμό της κίνησης του ρομπότ ως προς τον χρόνο. Η οδομετρία προκύπτει συνήθως με την ενσωμάτωση πληροφοριών κωδικοποίησης για τους τροχούς και είναι διαθέσιμη εκ των υστέρων, δηλαδή αφού κινηθεί το ρομπότ. Πιο συγκεκριμένα, στο χρονικό διάστημα $(t-1, t]$ το ρομπότ μετακινείται από την στάση x_{t-1} στην στάση x_t. Όπως προκύπτει και από την σχέση 3.4.14 που είδαμε προηγουμένως, μέσω της οδομετρίας υπολογίζεται μία σχετική κίνηση από την $\overline{x}_{t-1}$ στην $\overline{x}_t$. Εδώ το σύμβολο "-" δείχνει ότι έχουμε μετρήσεις οδομετρίας ενσωματωμένες σε ένα εσωτερικό σύστημα συντεταγμένων του ρομπότ, για το οποίο δεν ξέρουμε την σχέση που έχει με τις καθολικές συντεταγμένες του κόσμου. Το βασικό στοιχείο των συγκεκριμένων πληροφοριών που εκμεταλλευόμαστε στην εκτίμηση της κίνησης είναι ότι η σχετική διαφορά μεταξύ των $\overline{x}_{t-1}$ και $\overline{x}_t$ αποτελεί έναν καλό εκτιμητή

για την διαφορά των πραγματικών στάσεων x_{t-1} και x_t. Επομένως η πληροφορία κίνησης u_t δίνεται από το ακόλουθο διάνυσμα:

$$u_t = \begin{bmatrix} \overline{x}_{t-1} \\ \overline{x}_t \end{bmatrix} = \begin{bmatrix} \overline{x} \\ \overline{y} \\ \overline{\theta} \\ \overline{x}' \\ \overline{y}' \\ \overline{\theta}' \end{bmatrix} \tag{3.5.5}$$

Για να εξάγουμε την σχετική οδομετρία, μετασχηματίζουμε την πληροφορία κίνησης u_t σε μία ακολουθία τριών βημάτων: μία περιστροφή, ακολουθούμενη από μία κίνηση σε ευθεία γραμμή (μετατόπιση), και ακόμα μία περιστροφή όπως φαίνεται και στην εικόνα 3.5.2

Η αρχική περιστροφή ονομάζεται δ_{rot1}, η ευθεία μετατόπιση δ_{trans} και η δεύτερη περιστροφή δ_{rot2}. Έτσι κάθε ζεύγος στάσεων (x, x') έχει ένα μοναδικό διάνυσμα παραμέτρων $(\delta_{rot1}\ ,\ \delta_{trans}\ ,\ \delta_{rot2})$, και οι παράμετροι αυτές είναι αρκετές για την ανακατασκευή της σχετικής κίνησης μεταξύ των x και x'. Άρα, οι δ_{rot1}, δ_{trans}, δ_{rot2}, αποτελούν επαρκή στατιστικά στοιχεία για την σχετική κίνηση που κωδικοποιείται από την οδομετρία. Το πιθανοτικό μοντέλο κίνησης υποθέτει ότι οι παραπάνω τρεις παράμετροι έχουν αλλοιωθεί από ανεξάρτητο θόρυβο. Ο αλγόριθμος υπολογισμού του μοντέλου κίνησης, δηλαδή της πιθανότητας μεταβολής κατάστασης $p(x_t|u_t, x_{t-1})$, με χρήση των μετρήσεων που προσφέρει η οδομετρία δίνεται στην συνέχεια:

Algorithm 1: Αλγόριθμος Υπολογισμού Μοντέλου Κίνησης μέσω της Οδομετρίας

Input: x_{t-1}, x_t, u_t

1 $\delta_{rot1} = atan2(\overline{y}' - \overline{y}, \overline{x}' - \overline{x}) - \overline{\theta}$;

2 $\delta_{trans} = \sqrt{(\overline{y} - \overline{y}')^2 + (\overline{x} - \overline{x}')^2}$;

3 $\delta_{rot2} = \overline{\theta}' - \overline{\theta} - \delta_{rot1}$;

4 $\hat{\delta}_{rot1} = atan2(y' - y, x' - x) - \theta$;

5 $\hat{\delta}_{trans} = \sqrt{(y - y')^2 + (x - x')^2}$;

6 $\hat{\delta}_{rot2} = \theta' - \theta - \delta_{rot1}$;

7 $p_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, a_1 \cdot \hat{\delta}_{rot1} + a_2 \cdot \hat{\delta}_{trans})$;

8 $p_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, a_3 \cdot \hat{\delta}_{trans} + a_4 \cdot (\hat{\delta}_{rot1} + \hat{\delta}_{rot2})$;

9 $p_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, a_1 \cdot \hat{\delta}_{rot2} + a_2 \cdot \hat{\delta}_{trans})$;

10 **return** $p_1 \cdot p_2 \cdot p_3$;

Όπως μπορούμε να παρατηρήσουμε ο παραπάνω αλγόριθμος δέχεται ως είσοδο μία αρχική στάση x_{t-1}, ένα ζεύγος στάσεων u_t που προέρχονται από την οδομετρία του ρομπότ και μία υποθετική τελική στάση x_t και παράγει ως έξοδο την αριθμητική πιθανότητα $p(x_t|u_t, x_{t-1})$. Πιο αναλυτικά:

- *Βήματα 1-3*: Στα συγκεκριμένα βήματα ο αλγόριθμος ανακτά τις παραμέτρους της σχετικής κίνησης από τις ενδείξεις της οδομετρίας μέσω ενός αντιστρόφου μοντέλου κίνησης.
- *Βήματα 4-6*: Στα συγκεκριμένα βήματα υπολογίζονται οι αντίστοιχες παράμετροι της σχετικής κίνησης για τις δεδομένες στάσεις x_{t-1}, x_t.
- *Βήματα 7-9*: Στα συγκεκριμένα βήματα υπολογίζονται οι πιθανότητες σφάλματος για τις μεμονωμένες παραμέτρους της κίνησης. Για τον σκοπό αυτό χρησιμοποιείται η συνάρτηση $prob(a, b^2)$ η οποία αναπαριστά μία κανονική κατανομή σφάλματος για την παράμετρο a με μέση τιμή 0 και διακύμανση b^2.
- *Βήμα 10*: Το βήμα αυτό επιστέφει την συνδυασμένη πιθανότητα σφάλματος, η οποία προκύπτει από τον πολλαπλασιασμό των μεμονωμένων πιθανοτήτων σφάλματος p_1, p_2 και p_3. Το βήμα αυτό υποθέτει την

ανεξαρτησία των διαφορετικών πηγών σφάλματος.

Οι μεταβλητές a_1 έως a_4 που χρησιμοποιούνται από τον παραπάνω αλγόριθμο είναι παράμετροι που έχουν ορισθεί ειδικά για το ρομπότ και καθορίζουν τον θόρυβο στην κίνηση του.

Όμως σε πολλούς πρακτικούς αλγορίθμους, όπως είναι το φίλτρο σωματιδίων που θα αναλυθεί στην συνέχεια, αυτό που επιθυμούμε δεν είναι μία παράσταση κλειστής μορφής για τον υπολογισμό της $p(x_t|u_t, x_{t-1})$ για κάθε x_t , u_t , x_{t-1}. Αντίθετα αυτό που απαιτείται είναι δείγματα από την κατανομή $p(x_t|u_t, x_{t-1})$ μέσω της δειγματοληψίας της. Έτσι προκύπτει ο ακόλουθος αλγόριθμος ο οποίος ουσιαστικά εκτελεί δειγματοληψία της κατανομής $p(x_t|u_t, x_{t-1})$ για δεδομένα u_t και x_{t-1}:

Algorithm 2: Αλγόριθμος Δειγματοληψίας Μοντέλου Κίνησης μέσω της Οδομετρίας

Input: x_{t-1}, u_t

$\delta_{rot1} = atan2(\overline{y}' - \overline{y}, \overline{x}' - \overline{x}) - \overline{\theta}$;

$\delta_{trans} = \sqrt{(\overline{y} - \overline{y}')^2 + (\overline{x} - \overline{x}')^2}$;

$\delta_{rot2} = \overline{\theta}' - \overline{\theta} - \delta_{rot1}$;

$\hat{\delta}_{rot1} = \delta_{rot1} - sample(a_1 \cdot \hat{\delta}_{rot1} + a_2 \cdot \hat{\delta}_{trans})$;

$\hat{\delta}_{trans} = \delta_{trans} - sample(a_3 \cdot \hat{\delta}_{trans} + a_4 \cdot (\hat{\delta}_{rot1} + \hat{\delta}_{rot2}))$;

$\hat{\delta}_{rot2} = \delta_{rot2} - sample(a_1 \cdot \hat{\delta}_{rot2} + a_2 \cdot \hat{\delta}_{trans})$;

$x' = x + \hat{\delta}_{trans} \cdot cos(\theta + \hat{\delta}_{rot1})$;

$y' = y + \hat{\delta}_{trans} \cdot sin(\theta + \hat{\delta}_{rot1})$;

$\theta' = \theta + \hat{\delta}_{rot1} + \delta_{rot2}$;

return $x_t = (x', y', \theta')^\intercal$;

Όπως μπορούμε να παρατηρήσουμε ο συγκεκριμένος αλγόριθμος δέχεται ως είσοδο μία στάση x_{t-1} και μία ένδειξη οδομετρίας u_t και παράγει ως έξοδο μία τυχαία στάση x_t σύμφωνα με την κατανομή $p(x_t|u_t, x_{t-1})$. Ο αλγόριθμος αυτός επαναλαμβάνεται για κάθε δείγμα που αποτελεί πιθανή στάση του ρομπότ την χρονική στιγμή $t-1$. Η διαφορά του σε σχέση με τον προηγούμενο αλγόριθμο έγκειται στο γεγονός πως ουσιαστικά μαντεύει τυχαία μία στάση x_t μέσω της στάσης x_{t-1} και των δεδομένων της οδομετρίας u_t αντί να υπολογίζει την πιθανότητα μίας δεδομένης στάσης x_t του ρομπότ.

3.5.5 Πιθανότητα Μέτρησης

Τα μοντέλα μέτρησης του περιβάλλοντος περιγράφουν την διαδικασία με την οποία παράγονται οι μετρήσεις των αισθητήρων στον φυσικό κόσμο. Οι λεπτομέρειες του μοντέλου εξαρτώνται από τον εκάστοτε αισθητήρα που χρησιμοποιείται. Για παράδειγμα οι αισθητήρες λήψης εικόνων μοντελοποιούνται καλύτερα με προβολική γεωμετρία, ενώ οι αισθητήρες σόναρ με την περιγραφή του ηχητικού κύματος και την ανάκλαση του σε επιφάνειες του περιβάλλοντος. Τα μοντέλα αυτά λαμβάνουν υπόψιν τους την εγγενή αβεβαιότητα στις μετρήσεις των αισθητήρων του ρομπότ. Από μαθηματική άποψη, το μοντέλο μέτρησης ενός αισθητήρα ορίζεται ως μια κατανομή υπό συνθήκη πιθανότητας $p(z_t|x_t, m)$, όπου x_t είναι η στάση του ρομπότ, z_t είναι η μέτρηση στην χρονική στιγμή t, και m είναι ο χάρτης του περιβάλλοντος.

Πολλές φορές τα χαρακτηριστικά απόκρισης ενός αισθητήρα εξαρτώνται από μεταβλητές που δεν αναφέρονται αναλυτικά σε έναν πιθανοτικό αλγόριθμο ρομποτικής (όπως για παράδειγμα το υλικό της επιφάνειας των τοίχων). Αντίθετα η έλλειψη ακρίβειας που παρουσιάζουν τα μοντέλα των αισθητήρων ενσωματώνονται στις στοχαστικές πτυχές των αλγορίθμων. Με τη μοντελοποίηση της διαδικασίας μέτρησης ως μιας υπό συνθήκη πιθανότητας$p(z_t|x_t, m)$, αντί μιας αιτιοκρατικής συνάρτησης $z_t = f(x_t)$, η αβεβαιότητα στο μοντέλο του αισθητήρα μπορεί να ενσωματωθεί στις μη αιτιοκρατικές πτυχές του μοντέλου. Ωστόσο χρειάζεται προσοχή κατά την σχεδίαση ενός πιθανοτικού μοντέλου ώστε να ληφθούν υπόψιν οι διαφορετικοί τύποι αβεβαιότητας που μπορεί να επηρεάσουν μία μέτρηση του αισθητήρα.

Πολλοί αισθητήρες παράγουν περισσότερες από μία αριθμητικές μετρήσεις όταν τους ζητηθεί. Για παράδειγμα,

ένας αισθητήρας μέτρησης απόστασης λέιζερ, ο οποίος αποτελεί και τον αισθητήρα που χρησιμοποιείται στην παρούσα εργασία, παράγει συνήθως ολόκληρες σαρώσεις απόστασης. Συμβολίζοντας το πλήθος τέτοιων τιμών μίας μέτρησης z_t με K, προκύπτει ότι:

$$z_t = \{z_t^1, ..., z_t^K\} \tag{3.5.6}$$

όπου το z_t^k αναφέρεται σε μία μεμονωμένη μέτρηση (π.χ. μία τιμή απόστασης).

Η πιθανότητα $p(z_t|x_t, m)$ προκύπτει από το γινόμενο των πιθανοτήτων των μεμονωμένων μετρήσεων:

$$p(z_t|x_t, m) = \prod_{k=1}^{K} p(z_t^k|x_t, m) \tag{3.5.7}$$

Η παραπάνω σχέση υποθέτει πως υπάρχει ανεξαρτησία του θορύβου σε κάθε μεμονωμένη δέσμη μετρήσεων.

Στην συνέχεια θα αναλύσουμε δύο μοντέλα αισθητήρων απόστασης που χρησιμοποιούνται συχνά σε πολλές πρακτικές εφαρμογές.

Μοντέλο Δέσμης

Το συγκεκριμένο μοντέλο ενσωματώνει τέσσερις τύπους σφαλμάτων μέτρησης, που είναι όλοι βασικοί για την σωστή λειτουργία του μοντέλου:

- *Τον τοπικό θόρυβο μέτρησης*
- *Τα σφάλματα που οφείλονται σε μη αναμενόμενα αντικείμενα*
- *Τα σφάλματα που οφείλονται σε αποτυχίες εντοπισμού αντικειμένων*
- *Τον τυχαίο ανεξήγητο θόρυβο*

Τοπικός Θόρυβος Μέτρησης Σε έναν ιδανικό κόσμο, ένας αισθητήρας μέτρησης απόστασης θα μετρούσε πάντα την σωστή απόσταση από το πλησιέστερο αντικείμενο στο πεδίο μετρήσεων του. Στην πράξη όμως η τελική μέτρηση της απόστασης ενδέχεται να μην είναι ακριβής. Έτσι συμβολίζουμε με z_t^{k*} την πραγματική απόσταση του αντικειμένου και με z_t^k την μέτρηση της απόστασης του αντικειμένου σύμφωνα με τον αισθητήρα. Έχοντας τον χάρτη m του περιβάλλοντος, η πραγματική απόσταση z_t^{k*} του αντικειμένου σε σχέση με το ρομπότ μπορεί εύκολα να προσδιοριστεί καθώς γνωρίζοντας την θέση των αντικειμένων που βρίσκονται στον χώρο αλλά και την θέση του ρομπότ μπορούμε να προσδιορίσουμε ποια είναι η πραγματική του απόσταση από το πλησιέστερο αντικείμενο.

Όμως ακόμα και αν ο αισθητήρας μετράει σωστά την απόσταση από το πλησιέστερο αντικείμενο, η τιμή που επιστρέφει μπορεί να επηρεαστεί από κάποιο σφάλμα. Το τελευταίο ενδέχεται να οφείλεται στην περιορισμένη ανάλυση των αισθητήρων μέτρησης απόστασης, στις ατμοσφαιρικές επιδράσεις στο σήμα της μέτρησης κ.ο.κ. Αυτός ο θόρυβος μέτρησης μοντελοποιείται συνήθως με μία μικρού πλάτους Γκαουσιανή κατανομή την οποία συμβολίζουμε ως p_{hit} και η οποία έχει μέση τιμή z_t^{k*} και τυπική απόκλιση σ_{hit}.

Στην πράξη οι τιμές που μετρούνται από τον αισθητήρα μέτρησης απόστασης περιορίζονται σε ένα διάστημα $[0, z_{max}]$, όπου το z_{max} συμβολίζει την μέγιστη απόσταση που μπορεί να μετρήσει ο αισθητήρας.

Άρα τελικώς η κατανομή p_{hit} ορίζεται ως εξής:

$$p_{hit}(z_t^k|x_t, m) = \begin{cases} nN(z_t^k; z_t^{k*}, \sigma_{hit}^2) & \text{if } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.8}$$

όπου η απόσταση z_t^{k*} υπολογίζεται εύκολα γνωρίζοντας την θέση του ρομπότ και τον χάρτη μέσω ενός αλγορίθμου που ονομάζεται *ρίψη ακτινών*. Επιπλέον, το $N(z_t^k; z_t^{k*}, \sigma_{hit}^2)$ συμβολίζει την κανονική κατανομή μίας μεταβλητής z_t^k με μέση τιμή z_t^{k*} και τυπική απόκλιση σ_{hit}. Η τυπική απόκλιση σ_{hit} αποτελεί μία εγγενή παράμετρο του μοντέλου μέτρησης.

Μη αναμενώμενα αντικείμενα Τα περιβάλλοντα στα οποία λειτουργούν τα κινητά ρομπότ είναι συνήθως δυναμικά ενώ ο δεδομένος χάρτης του περιβάλλοντος είναι στατικός. Αυτό έχει ως αποτέλεσμα τα αντικείμενα που δεν περιέχονται στον χάρτη να αποτελούν την αιτία για την οποία οι αισθητήρες μέτρησης απόστασης παράγουν αναπάντεχα μικρές αποστάσεις σε σύγκριση με τον χάρτη. Ένας τρόπος για να αντιμετωπίσουμε τέτοια αντικείμενα είναι να τα θεωρήσουμε ως θόρυβο του αισθητήρα. Σε μία τέτοια περίπτωση οι αποστάσεις που μετρώνται είναι μικρότερες από την z_t^{k*} και όχι μεγαλύτερες. Η πιθανότητα ανίχνευσης μη αναμενόμενων αντικειμένων μειώνεται με την απόσταση. Και αυτό γιατί εάν για παράδειγμα θεωρήσουμε ότι έχουμε δύο αντικείμενα τα οποία εμφανίζονται ανεξάρτητα και με την ίδια σταθερή πιθανότητα μπροστά από το ρομπότ, προκειμένου ο αισθητήρας να μετρήσει το αντικείμενο που βρίσκεται πιο μακριά θα πρέπει το αντικείμενο που βρίσκεται πιο κοντά να είναι απών. Έτσι η πιθανότητα να μετρήσει το πιο μακρινό αντικείμενο είναι μικρότερη.

Με βάση το παραπάνω παράδειγμα γίνεται εύκολα αντιληπτό ότι μετρήσεις του αισθητήρα που οφείλονται σε μη αναμενώμενα αντικείμενα μπορούν να μοντελοποιηθούν ως μία εκθετική κατανομή της ακόλουθης μορφής:

$$p_{short}(z_t^k|x_t,m) = \begin{cases} n\lambda_{short}e^{-\lambda_{short}z_t^k} & \text{if } 0 \le z_t^k \le z_t^{k*} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.9}$$

όπου η παράμετρος λ_{short} της εκθετικής κατανομής αποτελεί μια εγγενή παράμετρο του μοντέλου μέτρησης.

Αποτυχίες εντοπισμού αντικειμένων Μερικές φορές ο αισθητήρας μέτρησης απόστασης αποτυγχάνει να εντοπίσει τα αντικείμενα που υπάρχουν στον χώρο. Για παράδειγμα στους αισθητήρες λέιζερ που χρησιμοποιούνται για μέτρηση απόστασης, κάτι τέτοιο είναι συχνό όταν ανιχνεύονται μαύρα αντικείμενα που απορροφούν το φως ή σε ορισμένα συστήματα λέιζερ κατά την μέτρηση αντικειμένων σε έντονο ηλιακό φωτισμό. Ένα τυπικό παράδειγμα της αποτυχίας ενός αισθητήρα είναι η *μέτρηση μέγιστης απόστασης*: Ο αισθητήρας δηλαδή επιστρέφει τη μέγιστη επιτρεπόμενη τιμή z_{max}. Εφόσον αυτό το φαινόμενο συμβαίνει αρκετά συχνά, είναι απαραίτητη η μοντελοποίηση των μετρήσεων μέγιστης απόστασης στο μοντέλο μέτρησης.

Η παραπάνω περίπτωση μοντελοποιείται με μία κατανομή σημειακής μάζας που είναι συγκεντρωμένη στην τιμή z_{max}:

$$p_{max}(z_t^k|x_t,m) = I(z=z_{max}) = \begin{cases} 1 & \text{if } z = z_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.10}$$

Εδώ το $I(z=z_{max})$ συμβολίζει την συνάρτηση-δείκτη που παίρνει την τιμή 1 αν το όρισμα της είναι αληθές και την τιμή 0 διαφορετικά.

Τυχαίος Θόρυβος Μέτρησης Οι αισθητήρες μέτρησης απόστασης παράγουν περιστασιακά εντελώς ανεξήγητες μετρήσεις. Οι μετρήσεις αυτές μπορούν να μοντελοποιηθούν με μία ομοιόμορφη κατανομή σε ολόκληρο το εύρος μετρήσεων $[0, z_{max}]$ του αισθητήρα:

$$p_{rand}(z_t^k|x_t,m) = \begin{cases} \frac{1}{z_{max}} & \text{if } z = z_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.11}$$

Στην εικόνα 3.5.3 φαίνονται οι κατανομές για τον κάθε τύπο σφάλματος που αναλύθηκε προηγουμένως.

Οι τέσσερις αυτές κατανομές 3.5.8, 3.5.9, 3.5.10, 3.5.11 τελικώς συνδιάζονται με την χρήση ενός σταθμισμένου μέσου όρου ώστε να προκύψει η τελική πιθανότητα μέτρησης όπως ορίζεται από το μοντέλο δέσμης. Ο ορσιμός της πιθανότητας αυτής δίνεται από την ακόλουθη σχέση:

$$p(z_t|x_t,m) = \begin{bmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{bmatrix}^{\intercal} \cdot \begin{bmatrix} p_{hit}(z_t^k|x_t,m) \\ p_{short}(z_t^k|x_t,m) \\ p_{max}(z_t^k|x_t,m) \\ p_{rand}(z_t^k|x_t,m) \end{bmatrix} \tag{3.5.12}$$

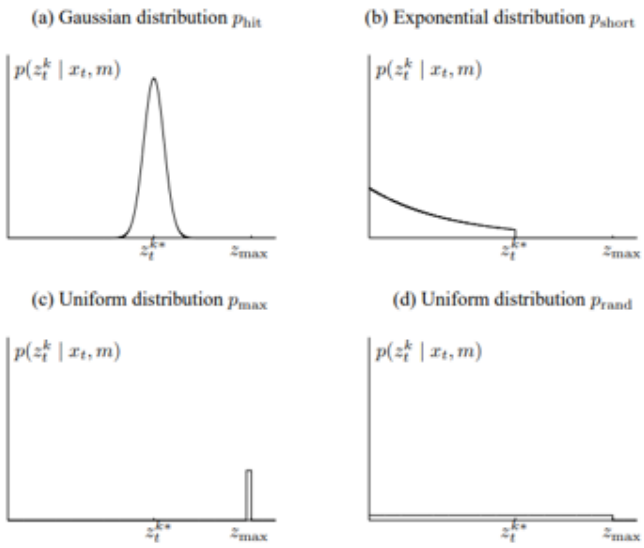

Figure 3.5.3: Κατανομές σφαλμάτων για το Μοντέλο Δέσμης

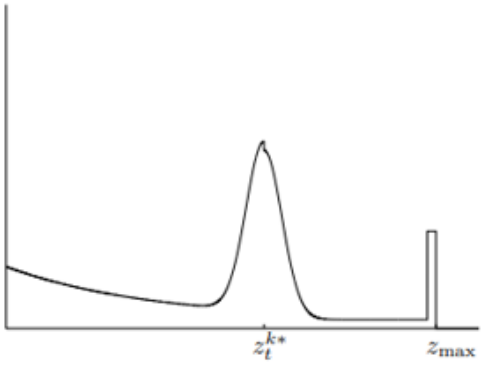

Figure 3.5.4: Πιθανότητα Μέτρησης όπως προκύπτει από το Μοντέλο Δέσμης του αισθητήρα μέτρησης απόστασης

Όπως μπορούμε να παρατηρήσουε, η πιθανότητα μέτρησης με βάση το μοντέλο δέσμης έχει ως παραμέτρους τις z_{hit}, z_{short}, z_{max} και z_{rand} με $z_{hit} + z_{short} + z_{max} + z_{rand} = 1$.

Στην εικόνα 3.5.4 μπορούμε να δούμε σχηματικά μία τυπική κατανομή που προκύπτει από τον γραμμικό συνδιασμό των τεσσάρων μεμονωμένων πυκνοτήτων. Όπως μπορούμε να παρατηρήσουμε τα βασικά χαρακτηριστικά και των τεσσάρων μοντέλων που περιγράψαμε προηγουμένως διατηρούνται στην συνδυασμένη πυκνότητα πιθανότητας.

Έχοντας λοιπόν ορίσει το μοντέλο μέτρησης $p(z_t|x_t, m)$ σύμφωνα με την σχέση 3.5.12, προκύπτει και ο αντίστοιχος αλγόριθμος υπολογισμού της πιθανότητας μίας σάρωσης απόστασης z_t που μας δίνεται σε κάθε χρονική στιγμή t από τον αισθητήρα μέτρησης απόστασης του ρομπότ για δεδομένη στάση x_t και χάρτη m του περιβάλλοντος:

Algorithm 3: Αλγόριθμος υπολογισμού Πιθανότητας Μέτρησης με βάση το Μοντέλο Δέσμης

Input: Στάση του ρομπότ: x_t, Μέτρηση απόστασης: z_t, Χάρτης: m

1 $q = 1$;

2 **for** $k = 1$ *to* K **do**

3 compute z_t^{k*} for the measurement z_t^k using ray-casting
$p = z_{hit} \cdot p_{hit}(z_t^k|x_t, m) + z_{short} \cdot p_{short}(z_t^k|x_t, m) + z_{max} \cdot p_{max}(z_t^k|x_t, m) + z_{rand} \cdot p_{rand}(z_t^k|x_t, m)$;

4 $q = q \cdot p$;

5 **return** q;

Όπως μπορούμε να παρατηρήσουμε η είσοδος του αλγορίθμου είναι μα πλήρης σάρωση απόστασης z_t, μία στάση x_t του ρομπότ και ένας δεδομένος χάρτης m. Ο εξωτερικός βρόγχος του αλγορίθμου πολλαπλασιάζει την πιθανότητα μεμονωμένων δεσμών z_t^k του αισθητήρα θεωρώντας τες όπως αναφέρθηκε και προηγουμένως στατιστικά ανεξάρτητες. Στο βήμα 3 εφαρμόζεται η ρίψη ακτινών για να υπολογισθεί η πραγματική απόσταση χωρίς θόρυβο για μια συγκεκριμένη μέτρηση του αισθητήρα. Η πιθανότητα κάθε μεμονωμένης μέτρησης υπολογίζεται στο βήμα 5, η οποία υλοποιεί τον κανόνα ανάμειξης των πυκνοτήτων που ορίσθηκε στην εξίσωση 3.5.12. Αφού η διαδικασία επαναληφθεί για όλες τις μεμονωμένες δέσμες z_t^k του αισθητήρα που αφορούν την μέτρηση z_t, ο αλγόριθμος επιστρέφει τη ζητούμενη πιθανότητα $p(z_t|x_t, m)$.

Μοντέλο Πιθανοφάνειας

Παρότι το βασισμένο σε δέσμη μοντέλο του αισθητήρα είναι στενά συνδεδεμένο με την γεωμετρία και την φυσική των αισθητήρων μέτρησης απόστασης, έχει δύο σημαντικά μειονεκτήματα:

- *Έλλειψη Ομαλότητας*: Σε «ακατάστατα» περιβάλλοντα με πολύ μικρά εμπόδια η κατανομή $p(z_t^k|x_t, m)$ μπορεί να είναι πολύ ανώμαλη ως προς την στάση x_t του ρομπότ. Για παράδειγμα σε ένα περιβάλλον με πολλές καρέκλες και τραπέζια, ένα κινητό ρομπότ θα εντοπίσει μέσω των αισθητήρων του τα πόδια των εμποδίων. Σε μία τέτοια περίπτωση, μικρές αλλαγές στην στάση x_t ενός ρομπότ ενδέχεται να έχουν τρομακτική επίδραση στη σωστή απόσταση μιας δέσμης αισθητήρα. Κατά συνέπεια το μοντέλο $p(z_t^k|x_t, m)$ είναι εξαιρετικά ασυνεχές ως προς την στάση x_t. Η κατεύθυνση θ_t του ρομπότ επηρεάζεται ιδιαίτερα επειδή οι μικρές αλλαγές της πορείας μπορούν να προκαλέσουν μεγάλες μετατοπίσεις της απόστασης στον χώρο x-y. Η έλλειψη ομαλότητας έχει δύο προβληματικές συνέπειες. Πρώτον, οποιαδήποτε κατά προσέγγιση αναπαράσταση της εκ των υστέρων κατανομής κινδυνεύει να μην βρει την σωστή στάση του ρομπότ, καθώς κοντινές καταστάσεις ενδέχεται να έχουν δραστικά διαφορετικές εκ των υστέρων πιθανότητες. Κάτι τέτοιο θέτει περιορισμούς στην ακρίβεια της προσέγγισης, οι οποίοι, αν δεν ικανοποιούνται, αυξάνουν το σφάλμα που προκύπτει στην εκ των υστέρων πιθανότητα. Δεύτερον οι μέθοδοι αναρρίχησης λόφων για την εύρεση των πιθανότερων καταστάσεων είναι επιρρεπής στα τοπικά ελάχιστα, επειδή υπάρχει μεγάλος αριθμός τοπικών ελαχίστων σε τέτοια ανώμαλα μοντέλα.

- *Υπολογιστική Πολυπλοκότητα*: Το βασισμένο σε δέσμη μοντέλο είναι και υπολογιστικά περίπλοκο. Ο υπολογισμός της $p(z_t^k|x_t, m)$ για κάθε μέτρηση του αισθητήρα z_t^k περιλαμβάνει τη ρίψη ακτινών, που είναι υπολογιστικά δαπανηρή.

Προκειμένου να ξεπερασθούν οι παραπάνω περιορισμοί συχνά χρησιμοποιείται ένα διαφορετικό μοντέλο το οποίο ονομάζεται *Μοντέλο Πιθανοφάνειας*. Η βασική ιδέα του συγκεκριμένου μοντέλου είναι να προβάλουμε πρώτα τα τελικά σημεία μιας σάρωσης z_t του αισθητήρα στο χώρο των καθολικών συντεταγμένων του χάρτη. Έτσι προκειμένου να προβάλουμε μία μεμονωμένη μέτρηση του αισθητήρα z_t^k στο καθολικό σύστημα συντεταγμένων του χάρτη, χρειάζεται να γνωρίζουμε ποια είναι η θέση του τοπικού συστήματος συντεταγμένων του ρομπότ σε σχέση με το καθολικό πλαίσιο συντεταγμένων, από που ξεκινάει η δέσμη z_k του αισθητήρα στο ρομπότ, και που σημαδεύει ο αισθητήρας. Έστω ότι η στάση του ρομπότ στην χρονική στιγμή t συμβολίζεται με $x_t = (x, y, \theta)$. Επιπλέον συμβολίζουμε τη σχετική θέση του αισθητήρα ως προς το σταθερό, τοπικό σύστημα συντεταγμένων του ρομπότ με $(x_{k,sens}, y_{k,sens})$, και το γωνιακό προσανατολισμό της δέσμης του αισθητήρα σε σχέση με την κατεύθυνση πορείας του ρομπότ με $\theta_{k,sens}$. Οι τιμές αυτές είναι συγκεκριμένες για κάθε αισθητήρα. Έτσι πλέον το τελικό σημείο της δέσμης z_t^k μπορεί να αντιστοιχιστεί στο καθολικό σύστημα συντεταγμένων του χάρτη μέσω του ακόλουθου τριγωνικού μετασχηματισμού:

$$\begin{bmatrix} x_{z_t^k} \\ y_{z_t^k} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_{k,sens} \\ y_{k,sens} \end{bmatrix} + z_t^k \cdot \begin{bmatrix} cos(\theta + \theta_{k,sens}) \\ sin(\theta + \theta_{k,sens}) \end{bmatrix} \tag{3.5.13}$$

Οι συντεταγμένες που προκύπτουν από τον τριγωνομετρικό μετασχηματισμό 3.5.13 έχουν νόημα μόνο όταν ο αισθητήρας εντοπίζει κάποιο εμπόδιο. Αν ο αισθητήρας μέτρησης απόστασης επιστρέψει τη μέγιστη αποδεκτή τιμή του $z_t^k = z_{max}$, οι συντεταγμένες δεν έχουν νόημα στο φυσικό κόσμο. Το μοντέλο μέτρησης με πεδίο πιθανοφάνειας απορρίπτει απλώς τις ενδείξεις μέγιστης απόστασης.

Σε αντιστοιχία με το μοντέλο δέσμης, υποθέτουμε ότι υπάρχουν τρία είδη πηγών θορύβου και αβεβαιότητας:

- *Θόρυβος μέτρησης*: Ο θόρυβος που προέρχεται από την διαδικασία μέτρησης μοντελοποιείται με Γκαουσιανές κατανομές. Στο επίπεδο x-y αυτό περιλαμβάνει την εύρεση του πλησιέστερου εμποδίου στον χάρτη. Έστω ότι η Ευκλείδεια απόσταση μεταξύ των συντεταγμένων $(x_{z_t^k}, y_{z_t^k})^\intercal$ της μέτρησης και του πλησιέστερου αντικειμένου στο χάρτη m συμβολίζεται με $dist$. Άρα η πιθανότητα μιας μέτρησης του αισθητήρα δίνεται από μία Γκαουσιανή κατανομή κεντραρισμένη στο μηδέν, η οποία αποτυπώνει τον θόρυβο του αισθητήρα:

$$p_{hit}(z_t^k|x_t, m) = \varepsilon_{\sigma_{hit}^2}(dist^2) \tag{3.5.14}$$

- *Αποτυχία μέτρησης*: Όπως και στην περίπτωση του μοντέλου δέσμης, υποθέτουμε ότι οι ενδείξεις μέγιστης απόστασης μοντελοποιούνται με μια κατανομή σημειακής μάζας p_{max} η οποία προκύπτει από την σχέση 3.5.15.

$$p_{max}(z_t^k|x_t, m) = I(z = z_{max}) = \begin{cases} 1 & \text{if } z = z_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.15}$$

- *Ανεξήγητη τυχαία μέτρηση*: 'Οπως και στην περίπτωση του μοντέλου δέσμης χρησιμοποιείται μία ομοιόμορφη κατανομή p_{rand} για την μοντελοποιηση του τυχαίου θορύβου στην αντίληψη του ρομπότ η οποία δίνεται από την σχέση 3.5.16.

$$p_{rand}(z_t^k|x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{if } z = z_{max} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.16}$$

Σε αντιστοιχία με το βασισμένο σε δέσμη μοντέλο αισθητήρα, η ζητούμενη πιθανότητα $p(z_t^k|x_t, m)$ ενσωματώνει και τις τρεις παραπάνω κατανομές σε έναν ενιαίο γραμμικό μετασχηματισμό ο οποιος δίνεται από την ακόλουθη σχέση και ο οποίος χρησιμοποιεί τους γνωστούς συντελεστές στάθμισης z_{hit}, z_{max} και z_{rand}:

$$z_{hit} \cdot p_{hit}(z_t^k|x_t, m) + z_{max} \cdot p_{max}(z_t^k|x_t, m) + z_{rand} \cdot p_{rand}(z_t^k|x_t, m) \tag{3.5.17}$$

Έχοντας λοιπόν ορίσει το μοντέλο μέτρησης $p(z_t|x_t, m)$ σύμφωνα με την σχέση 3.5.17, προκύπτει και ο αντίστοιχος αλγόριθμος υπολογισμού της πιθανότητας μίας σάρωσης απόστασης z_t που μας δίνεται σε κάθε χρονική στιγμή t από τον αισθητήρα μέτρησης απόστασης του ρομπότ για δεδομένη στάση x_t και χάρτη m του περιβάλλοντος:

Algorithm 4: Αλγόριθμος υπολογισμού Πιθανότητας Μέτρησης με βάση το Μοντέλο Πιθανοφάνειας

Input: Στάση του ρομπότ: x_t, Μέτρηση απόστασης: z_t, Χάρτης: m

1 $q = 1$;

2 **for** $k = 1 \ to \ K$ **do**

3 **if** $z_t^k \neq z_{max}$ **then**

4 $x_{z_t^k} = x + x_{k,sens} \cdot cos\theta - y_{k,sens} \cdot sin\theta + z_t^k \cdot cos(\theta + \theta_{k,sens})$;

5 $y_{z_t^k} = y + y_{k,sens} \cdot cos\theta - x_{k,sens} \cdot sin\theta + z_t^k \cdot sin(\theta + \theta_{k,sens})$;

6 $dist^2 = min_{x',y'}\{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2 | < x', y' > \ \ occupied \ in \ m\}$;

7 $q = q \cdot (z_{hit} \cdot prob(dist^2 | \sigma_{hit}^2) + \frac{z_{random}}{z_{max}})$

8 **return** q;

Ο εξωτερικός βρόγχος, όπως και στον βασισμένο στο μοντέλο δέσμης αλγόριθμο, πολλαπλασιάζει τις μεμονωμένες τιμές της $p(z_t^k | x_t, m)$, με βάση την υπόθεση ότι ο θόρυβος είναι ανεξάρτητος στις διαφορετικές δέσμες του αισθητήρα. Το βήμα 3 ελέγχει αν η ένδειξη του αισθητήρα είναι ένδειξη μέγιστης απόστασης και σε περίπτωση που κάτι τέτοιο ισχύει, η συγκεκριμένη ένδειξη απλώς αγνοείται. Στα βήματα που ακολουθούν υπολογίζεται η απόσταση από το πλησιέστερο εμπόδιο στο χώρο x-y (βήματα 4-6), και η αντίστοιχη πιθανοφάνεια προκύπτει στο βήμα 7 με την ανάμειξη μιας κανονικής και μίας ομοιόμορφης κατανομής. Η συνάρτηση $prob(dist, \sigma_{hit})$ υπολογίζει την πιθανότητα της απόστασης $dist$ σύμφωνα με μια κεντραρισμένη στο μηδέν Γκαουσιανή κατανομή με τυπική απόκλιση σ_{hit}.

3.5.6 Κατανομές Πεποίθησης

Μια άλλη βασική έννοια που χρησιμοποιείται συχνά στην εκτίμηση της στάσης του ρομπότ είναι η πεποίθηση. Για μία στάση x_t, η πεποίθηση μπορεί να συμβολιστεί ως $bel(x_t)$ και ουσιαστικά αποτελεί μία σύντμηση για την εκ των υστέρων πιθανότητα:

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t}) \tag{3.5.18}$$

Η συγκεκριμένη εκ των υστέρων πιθανότητα είναι η κατανομή πιθανότητας για την στάση x_t στη χρονική στιγμή t, η οποία ορίζεται με βάση όλες τις προηγούμενες μετρήσεις $z_{1:t}$ και ενέργειες ελέγχου $u_{1:t}$.

Παρατηρώντας την σχέση 3.5.18 μπορούμε να παρατηρήσουμε πως η πεποίθηση λαμβάνεται μετά την ενσωμάτωση της μέτρησης z_t. Πολλές φορές όμως είναι χρήσιμος ο υπολογισμός μίας εκ των υστέρων πιθανότητας πριν από την ενσωμάτωση της μέτρησης z_t, δηλαδή αμέσως μετά της ενέργειας ελέγχου u_t. Μία τέτοια εκ των υστέρων πιθανότητα συμβολίζεται ως εξής:

$$\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t}) \tag{3.5.19}$$

Η παραπάνω κατανομή πιθανότητας αναφέρεται συχνά ως *πρόβλεψη*. Αυτή η ορολογία αντικατοπτρίζει το γεγονός ότι η πεποίθηση προβλέπει την κατάσταση στη χρονική στιγμή t με βάση την εκ των υστέρων πιθανότητα της προηγούμενης στάσης, πριν από την ενσωμάτωση της μέτρησης στη χρονική στιγμή t. Ο υπολογισμός της $bel(x_t)$ από την $\overline{bel}(x_t)$ ονομάζεται *διόρθωση ή ενημέρωση μέτρησης*.

3.5.7 Φίλτρο Bayes

Ο πιο γενικός αλγόριθμος για τον υπολογισμό πεποιθήσεων δίνεται από τον αλγόριθμο του φίλτρου Bayes. Ο συγκεκριμένος αλγόριθμος υπολογίζει σε κάθε χρονική στιμγή t την κατανομή πεποίθησης $bel(x_t)$ από δεδομένα μετρήσεων και ενεργειών ελέγχου.

Το φίλτρο είναι αναδρομικό, δηλαδή η πεποίθηση $bel(x_t)$ στη χρονική στιγμή t υπολογίζεται από την πεποίθηση $bel(x_{t-1})$ στην χρονική στιγμή t-1. Σε κάθε επανάληψη, η είσοδος του φίλτρου είναι η πεποίθηση $bel(x_{t-1})$ στην προηγούμενη χρονική στιγμή t-1, μαζί με την τελευταία ενέργεια ελέγχου u_t και την τελευταία μέτρηση z_t από τους αισθητήρες. Η έξοδος του αλγορίθμου μετά την ολοκλήρωση της κάθε επανάληψης είναι η πεποίθηση $bel(x_t)$ στην χρονική στιγμή t. Η λειτουργία του αλγορίθμου φαίνεται στην συνέχεια:

Algorithm 5: Αλγόριθμος Φίλτρου Bayes

Input: Αρχική Πεποίθηση $bel(x_0)$, Μοντέλο Κίνησης $p(x_t|u_t, x_{t-1})$, Μοντέλο Μέτρησης $p(z_t|x_t)$
Output: Ενημερωμένη πεποίθηση $bel(x_t)$

for *each* $t = 1, 2, \ldots, T$ **do**
  // Βήμα Πρόβλεψης
  $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) \cdot bel(x_{t-1})\, dx_{t-1}$;
  // Βήμα Ενηνέρωσης
  $bel(x_t) = \eta \cdot p(z_t|x_t) \cdot \overline{bel}(x_t)$;
return $bel(x_T)$;

Σε κάθε επανάληψη του αλγορίθμου υπολογίζεται αναδρομικά η πεποίθηση $bel(x_t)$ από την πεποίθηση $bel(x_{t-1})$, η οποία έχει υπολογισθεί στο προηγούμενο βήμα.

Τα δύο σημαντικά βήματα στον αλγόριθμο του φίλτρου Bayes είναι τα ακόλουθα:

- *Βήμα 2*: Στο συγκεκριμένο βήμα ο αλγόριθμος επεξεργάζεται την ενέργεια ελέγχου u_t. Για τον σκοπό αυτό, χρησιμοποιεί την πιθανότητα μεταβολής κατάστασης $p(x_t|u_t, x_{t-1})$. Ειδικότερα. η πεποίθηση $\overline{bel}(x_t)$ που αντιστοιχίζεται από το ρομπότ στην κατάσταση x_t προκύπτει ως το ολοκλήρωμα (άθροισμα) του γινομένου δύο κατανομών: της εκ των προτέρων πεποίθησης $bel(x_{t-1})$ που έχει αντιστοιχιστεί στην στάση x_{t-1} του ρομπότ και της πιθανότητας να προκαλεί η ενέργεια ελέγχου u_t μια μεταβολή από την κατάσταση x_{t-1} στην x_t.
- *Βήμα 4*: Στο συγκεκριμένο βήμα εφαρμόζεται το δεύτερο βήμα του φίλτρου Bayes που ονομάζεται *βήμα ενημέρωσης*. Σύμφωνα με το βήμα αυτό ο αλγόριθμος πολλαπλασιάζει την πεποίθηση $\overline{bel}(x_t)$ με την πιθανότητα μέτρησης $p(z_t|x_t)$.

Τα δύο αυτά βήματα επαναλαμβάνονται για κάθε υποθετική στάση x_t του ρομπότ την χρονική στιγμή t.

3.5.8 Φίλτρο Σωματιδείων

Το φίλτρο σωματιδίων αποτελεί μία μη-παραμετρική υλοποίηση του φίλτρου Bayes. Η βασική ιδέα του φίλτρου σωματιδίων είναι η αναπαράσταση της εκ των υστέρων πεποίθησης $bel(x_t)$ με ένα σύνολο τυχαίων δειγμάτων στάσεων του ρομπότ που έχουν ληφθεί από την συγκεκριμένη κατανομή. Με άλλα λόγια τα φίλτρα σωματιδίων δεν αναπαριστούν την κατανομή με παραμετρική μορφή αλλά με ένα σύνολο δειγμάτων που έχουν ληφθεί από την κατανομή. Μια τέτοια αναπαράσταση είναι προσεγγιστική αλλά όχι παραμετρική, και μπορεί επομένως να αναπαραστήσει ένα ευρύτερο χώρο κατανομών από τις Γκαουσιανές κατανομές, για παράδειγμα.

Στα φίλτρα σωματιδίων, τα δείγματα μίας εκ των υστέρων κατανομής ονομάζονται σωματίδια και συμβολίζονται με:

$$X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{3.5.20}$$

Κάθε σωματίδιο $x_t^{[i]}$ $(με 1 \leq i \leq M)$ είναι ένα συνεπτυγμένο στιγμιότυπο της στάσης του ρομπότ την χρονική στιγμή t, όπου M ο συνολικός αριθμός των σωματιδίων. Με άλλα λόγια, ένα σωματίδιο αναπαριστά μία υπόθεση σχετικά με το ποια ενδέχεται να είναι η πραγματική στάση του ρομπότ την χρονική στιγμή t.

Η ιδέα πίσω από τα φίλτρα σωματιδίων είναι η προσέγγιση της εκ των υστέρων πεποίθησης $bel(x_t)$ με το σύνολο των σωματιδίων X_t. Ιδανικά, η πιθανότητα να συμπεριληφθεί μία υπόθεση στάσης x_t στο σύνολο σωματιδίων X_t θα πρέπει να είναι ανάλογη της πεποίθησης $bel(x_t)$:

$$x_t^{[i]} \sim p(x_t|z_{1:t}, u_{1:t}) = bel(x_t) \tag{3.5.21}$$

Με βάση την παραπάνω σχέση γίνεται κατανοητό ότι όσο περισσότερα είναι τα δείγματα με τα οποία γεμίζει μία υποπεριοχή του χώρου καταστάσεων, τόσο πιθανότερο είναι να περιέχεται η πραγματική κατάσταση στην συγκεκριμένη περιοχή.

Όπως και κάθε αλγόριθμος που αποτελεί πρακτική υλοποίηση του φίλτρου Bayes έτσι και ο αλγόριθμος του φίλτρου σωματιδίων κατασκευάζει την εκ των υστέρων κατανομή $bel(x_t)$ αναδρομικά από την εκ των υστέρων κατανομή $bel(x_{t-1})$ της προηγούμενης χρονικής στιγμής. Αφού στην συγκεκριμένη περίπτωση οι εκ των υστέρων κατανομές αναπαρίστανται με σύνολα σωματιδίων, αυτό σημαίνει ότι τα φίλτρα σωματιδίων κατασκευάζουν το σύνολο σωματιδίων X_t αναδρομικά από το σύνολο σωματιδίων X_{t-1}.

Ο αλγόριθμος του φίλτρου σωματιδίων δίνεται στην συνέχεια:

Algorithm 7: Αλγόριθμος Φίλτρου Σωματιδίων

Data: Σύνολο σωματιδίων X_{t-1}, Ενέργεια ελέγχου u_t, Μέτρηση z_t
Result: Ενημερωμένο σύνολο σωματιδίων X_t

1 **foreach** *particle* x_i *in* X_{t-1} **do**
  // Βήμα Πρόβλεψης
2 Sample a new particle $x'_i \sim p(x_t|u_t, x_i)$;
3 Add the sampled particle to the temporary set of particles $\overline{X}_t$;
4 Assign weight $w_t^{[i]} = p(z_t|x'_i)$;
5 Normalize weights w_i;
6 **foreach** *particle* x_i *in* $\overline{X}_t$ **do**
  // Βήμα Αναδειγματοληψίας
7 Sample particle x_i with probability proportional to weight $w_t^{[i]}$;
8 Add the sampled particle to the set of updated particles X_t;

Όπως μπορούμε να παρατηρήσουμε, η είσοδος του αλγορίθμου είναι το σύνολο σωματιδίων X_{t-1} μαζί με την τελευταία ενέργεια ελέγχου u_t και την τελευταία μέτρηση z_t. Στην συνέχεια ο αλγόριθμος κατασκευάζει πρώτα ένα προσωρινό σύνολο σωματιδίων $\overline{X}_t$ που αναπαριστά την πεποίθηση $\overline{bel}(x_t)$. Για να το επιτύχει, επεξεργάζεται συστηματικά κάθε σωματίδιο $x_{t-1}^{[i]}$ στο σύνολο σωματιδίων X_{t-1} της εισόδου. Έπειτα μετασχηματίζει τα σωματίδια αυτά στο σύνολο X_t το οποίο προσεγγίζει την εκ των υστέρων κατανομή $bel(x_t)$. Πιο συγκεκριμένα:

- *Βήμα 2*: Στο συγκεκριμένο βήμα παράγεται μία υποθετική κατάσταση x'_i για την χρονική στιγμή t με βάση το σωματίδιο $x_{t-1}^{[i]}$ και την ενέργεια ελέγχου u_t. Ουσιαστικά το βήμα αυτό περιλαμβάνει την δειγματοληψία από την κατανομή μεταβολής κατάστασης. Έτσι, για να υλοποιήσει κανείς το συγκεκριμένο βήμα πρέπει να είναι σε θέση να κάνει δειγματοληψία από την κατανομή μεταβολής κατάστασης $p(x_t|u_t, x_{t-1})$. Το σύνολο των σωματιδίων που προκύπτουν μετά από M επαναλήψεις αποτελεί την αναπαράσταση του φίλτρου για την $\overline{bel}(x_t)$.

- *Βήμα 4*: Στο συγκεκριμένο βήμα υπολογίζεται για κάθε σωματίδιο ο αποκαλούμενος συντελεστής σπουδαιότητας, που συμβολίζεται με $w_t^{[i]}$. Οι συντελεστές σπουδαιότητας χρησιμοποιούνται για την ενσωμάτωση της μέτρησης z_t στο σύνολο των σωματιδίων. Επομένως, ο συντελεστής σπουδαιότητας εκφράζει την πιθανότητα της μέτρησης z_t σύμφωνα με το σωματίδιο x'_i και δίνεται από την σχέση $w_t^{[i]} = p(z_t|x'_i)$. Αν ερμηνεύσουμε το $w_t^{[i]}$ ως το συντελεστή στάθμισης (ή βάρος) ενός σωματιδίου το σύνολο των σταθμισμένων σωματιδίων αναπαριστά (κατά προσέγγιση) την εκ των υστέρων πεποίθηση $bel(x_t)$ του φίλτρου.

- *Βήματα 6-8*: Το πραγματικό τέχνασμα του αλγορίθμου βασίζεται στα συγκεκριμένα βήματα όπου εφαρμόζεται μία διαδικασία γνωστή ως αν δειγματοληψία ή δειγματοληψία σπουδαιότητας. Ο αλγόριθμος επιλέγει με αντικατάσταση M σωματίδια από το προσωρινό σύνολο $\overline{X}_t$. Η πιθανότητα επιλογής κάθε σωματιδίου δίνεται από τον συντελεστή στάθμισης της σπουδαιότητας του. Η αναδειγματοληψία μετασχηματίζει ένα σύνολο σωματιδίων με M σωματίδια σε ένα άλλο σύνολο σωματιδίων του ίδιου μεγέθους. Αφού ενσωματώνονται οι συντελεστές στάθμισης της σπουδαιότητας στην διαδικασία της αναδειγματοληψίας, η κατανομή των σωματιδίων αλλάζει: ενώ πριν από το βήμα της αναδειγματοληψίας τα σωματίδια ήταν κατανεμημένα σύμφωνα με την $\overline{bel}(x_t)$, μετά την αναδειγματοληψία κατανέμονται (κατά προσέγγιση) σύμφωνα με την εκ των υστέρων πεποίθηση $bel(x_t)$. Στην πραγματικότητα το σύνολο που προκύπτει έχει πολλά διπλότυπα αφού τα σωματίδια επιλέγονται με αντικατάσταση. Η σημαντική λειτουργία του βήματος αναδειγματοληψίας είναι ότι επιβάλει την επιστροφή των σωματιδίων στην εκ των υστέρων πε-

ποίθηση $bel(x_t)$. Με άλλα λόγια, επανασυγκεντρώνει το σύνολο των σωματιδίων σε περιοχές του χώρου καταστάσεων με μεγάλη εκ των υστέρων πιθανότητα. Με αυτόν τον τρόπο εστιάζει τους υπολογιστικούς πόρους που χρησιμοποιούνται από τον αλγόριθμο του φίλτρου στις περιοχές του χώρου καταστάσεων που έχουν την μεγαλύτερη σημασία.

Στην πράξη, οποιαδήποτε συγκεκριμένη υλοποίηση του φίλτρου σωματιδίων απαιτεί τον ορισμό τριών κατανομών πιθανότητας:

- *Την αρχική πεποίθηση* $p(x_0)$
- *Την πιθανότητα μέτρησης* $p(z_t|x_t)$
- *Την πιθανότητα μεταβολής κατάστασης* $p(x_t|u_t, x_{t-1})$

3.5.9 Εντοπισμός Θέσης Monte Carlo

Ο εντοπισμός θέσης Monte Carlo αναπαριστά την πεποίθηση $bel(x_t)$ με σωματίδια. Ο αλγόριθμος ονομάζεται εντοπισμός θέσης Μόντε Κάρλο (Monte Carlo Localization ή MCL) και εφαρμόζεται σε προβλήματα τοπικού και καθολικού εντοπισμού θέσης. Ουσιαστικά προκύπτει από τον αλγόριθμο του φίλτρου σωματιδίων (7) με αντικατάσταση των κατάλληλων πιθανοτικών μοντέλων κίνησης (10) και μέτρησης (5 ή 8). Έτσι προκύπτει ο ακόλουθος αλγόριθμος:

Algorithm 8: Αλγόριθμος Εντοπισμού Θέσης Monte Carlo

Input: Σύνολο σωματιδίων: X_{t-1}, Ενέργεια Ελέγχου: u_t, Μέτρηση απόστασης: z_t, Χάρτης: m

1 $\overline{X}_t = X_t = \emptyset$;
2 **for** $m = 1$ *to* M **do**
3 $\quad x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$;
4 $\quad w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$;
5 $\quad \overline{X}_t = \overline{X}_t + < x_t^{[m]}, w_t^{[m]} >$;
6 **for** $m = 1$ *to* M **do**
7 $\quad$ Sample a particle $x_t^{[m]}$ from the set $\overline{X}_t$ with probability proportional to weight $w_t^{[m]}$;
8 $\quad$ add $x_t^{[m]}$ to X_t;
9 $x_t^{final} = \frac{\sum_{m=1}^{M} x_t^{[m]} \cdot w_t^{[m]}}{\sum_{m=1}^{M} w_t^{[m]}}$;
10 **return** X_t, x_t^{final};

Τα βασικά βήματα του συγκεκριμένου αλγορίθμου είναι τα ακόλουθα:

- *Βήμα 3*: Στο συγκεκριμένο βήμα, σε κάθε σωματίδιο που ανήκει στο σύνολο σωματιδίων X_{t-1}, εφαρμόζεται ο αλγόριθμος δειγματοληψίας του μοντέλου κίνησης (10) με βάση δεδομένα που προέχρονται από την οδομετρία του ρομπότ που αναλύθηκε προηγουμένως.
- *Βήμα 4*: Στο συγκεκριμένο βήμα, σε κάθε σωματίδιο που ανήκει στο σύνολο σωματιδίων X_{t-1}, εφαρμόζεται ο αλγόριθμος υπολογισμού της πιθανότητας μέτρησης, είτε με βάση το μοντέλο δέσμης (5) είτε το μοντέλο πιθανοφάνειας (8) που αναλύθηκαν προηγουμένως, για να προσδιοριστεί ο συντελεστής στάθμισης σπουδαιότητας του.
- *Βήμα 5*: Στο συγκεκριμένο βήμα, κάθε νέο σωματίδιο που προκύπτει από το μοντέλο κίνησης μαζί με τον συντελεστή σπουδαιότητας του που προκύπτει από το μοντέλο μέτρησης προστίθενται στο προσωρινό σύνολο σωματιδίων $\overline{X}_t$.
- *Βήματα 6-8*: Στα συγκεκριμένα βήματα εκτελείτε η μέθοδος αναδειγματοληψίας όπως εξηγήθηκε στην ανάλυση του αλγορίθμου του φίλτρου σωματιδίων (7).
- *Βήμα 9*: Στο συγκεκριμένο βήμα υπολογίζεται η τελική στάση του ρομπότ x_t^{final} την χρονική στιμγή t ως ο σταθμισμένος μέσος όρο των στάσεων όλων των σωματιδίων κανονικοποιημένος με βάση τον

συντελεστή σπουδαιότητας των σωματιδίων.

Η αρχική πεποίθηση $bel(x_0)$ υπολογίζεται με την τυχαία παραγωγή M σωματιδίων από την εκ των προτέρων κατανομή $p(x_0)$. Σε περίπτωση που η στάση του ρομπότ είναι αρχικά γνωστή η συγεκριμένη κατανομή μοντελοποιείται ως μία πολύ στενή Γκαουσιανή κατανομή γύρω από την θέση αυτή. Επιπλέον σε κάθε τέτοιο σωματίδιο $x_0^{[m]}$ αντιστοιχιζέται ένας συντελεστής στάθμισης με αρχική τιμή $w_0^{[m]} = \frac{1}{M}$.

3.5.10 Αλγόριθμος MCL με τυχαία σωματίδια: Ανάκαμψη από αποτυχίες

Με την παραπάνω μορφή ο αλγόριθμος εντοπισμού θέσης Monte Carlo δεν μπορεί να ανακάμψει σε περίπτωση αποτυχίας του καθολικού εντοπισμού θέσης. Αυτό συμβαίνει γιατί καθώς προκύπτει η στάση του ρομπότ, τα σωματίδια σε σημεία διαφορετικά από την πιο πιθανή στάση εξαφανίζονται σταδιακά και ο αλγόριθμος δεν μπορεί να ανακάμψει αν η συγκεκριμένη στάση δεν είναι σωστή. Στην πράξη οποιοσδήποτε στοχαστικός αλγόριθμος, όπως ο MCL, ενδέχεται να απορρίψει τυχαία όλα τα σωματίδια που βρίσκονται κοντά στην σωστή στάση κατά την διάρκεια του βήματος της αναδειγματοληψίας. Το πρόβλημα είναι ιδιαίτερα μεγάλο όταν ο αριθμός των σωματιδίων είναι μικρός (π.χ. M=50) καθώς και όταν τα σωματίδια είναι διεσπαρμένα σε μεγάλο χώρο (π.χ. κατά τον καθολικό εντοπισμό θέσης).

Το συγκεκριμένο πρόβλημα μπορεί να επιλυθεί με μία αρκετά απλή ευριστική τεχνική. Η ιδέα της τεχνικής αυτής είναι η προσθήκη τυχαίων σωματιδίων στο σύνολο των σωματιδίων. Μια τέτοια εισήγηση τυχαίων σωματιδίων μπορεί να αιτιολογηθεί μαθηματικά εάν υποθέσουμε ότι το ρομπότ ενδέχεται να απαχθεί με μικρή πιθανότητα, παράγοντας έτσι ένα ποσοστό τυχαίων καταστάσεων στο μοντέλο της κίνησης. Ωστόσο ακόμα και αν το ρομπότ δεν απαχθεί, τα επιπλέον σωματίδια προσθέτουν ένα επιπλέον επίπεδο ανθεκτικότητας. Το ερώτημα είναι πόσα σωματίδια πρέπει να προστεθούν σε κάθε επανάληψη του αλγορίθμου και από ποια κατανομή θα πρέπει να παραχθούν τα σωματίδια αυτά. Θα μπορούσε κάποιος να προσθέσει ένα σταθερό αριθμό σωματιδίων σε κάθε επανάληψη. Μια καλύτερη ιδέα είναι η προσθήκη σωματιδίων με βάση κάποια εκτίμηση της απόδοσης του αλγορίθμου εντοπισμού θέσης. Ένας τρόπος υλοποίησης της παραπάνω ιδέας είναι η παρακολούθηση της πιθανότητας μετρήσεων των αισθητήρων $p(z_t|z_{t-1}, u_t, m)$ και η συσχέτιση τους με την μέση πιθανότητα μέτρησης η οποία μπορεί πολύ εύκολα να προκύψει μέσα από τα δεδομένα. Στα φίλτρα σωματιδίων, μια προσέγγιση για τη συγκεκριμένη ποσότητα προκύπτει εύκολα από το συντελεστή στάθμισης της σπουδαιότητας, επειδή ο τελευταίος είναι εξ ορισμού μια στοχαστική εκτίμηση αυτής της πιθανότητας. Έτσι λοιπόν μέσω της ακόλουθης εξίσωσης μπορούμε να προσεγγίσουμε την ζητούμενη ποσότητα:

$$p(z_t|z_{t-1}, u_t, m) = \frac{1}{M} \sum_{m=1}^{M} w_t^{[m]} \tag{3.5.22}$$

Ένας από τους λόγους που η παραπάνω ποσότητα μπορεί να εμφανίσει χαμηλές τιμές είναι η αποτυχία του εντοπισμού θέσης. Βέβαια υπάρχουν και άλλοι λόγοι για τους οποίους η πιθανότητα μέτρησης μπορεί να είναι μικρή, όπως το γεγονός ότι η ποσότητα του θορύβου του αισθητήρα ενδέχεται να είναι αφύσικα μεγάλη ή τα σωματίδια μπορεί να εξακολουθούν να είναι διασκορπισμένα κατά τη διάρκεια μίας φάσης καθολικού εντοπισμού θέσης. Για τον λόγο αυτό είναι καλή ιδέα να εξομαλύνουμε την παραπάνω εκτίμηση υπολογίζοντας το μέσο όρο της για πολλά χρονικά βήματα. Πιο συγκεκριμένα, συχνά διατηρούμε έναν βραχυπρόθεσμο μέσο όρο της πιθανότητας μέτρησης και τον συσχετίζουμε με τον μακροπρόθεσμο μέσο όρο όταν προσδιορίζουμε τον αριθμό των τυχαίων σωματιδίων.

Ο αλγόριθμος που ακολουθεί την παραπάνω λογική ονομάζεται Προσαρμοστικός αλγόριθμος εντοπισμού θέσης Monte Carlo (Adaptive MCL). Ο αλγόριθμος αυτός είναι προσαρμοστικός, με την έννοια ότι παρακολουθεί τον βραχυπρόθεσμο και τον μακροπρόθεσμο μέσο όρο της πιθανότητας $p(z_t|z_{1:t-1}, u_{1:t}, m)$.

Algorithm 9: Προσαρμοστικός Αλγόριθμος Εντοπισμού Θέσης Monte Carlo

Input: Σύνολο σωματιδίων: X_{t-1}, Ενέργεια Ελέγχου: u_t, Μέτρηση απόστασης: z_t, Χάρτης: m

1 $\overline{X}_t = X_t = \emptyset$;
2 static w_{slow}, w_{fast};
3 **for** $m = 1$ *to* M **do**
4 $\quad x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$;
5 $\quad w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$;
6 $\quad \overline{X}_t = \overline{X}_t + < x_t^{[m]}, w_t^{[m]} >$;
7 $\quad w_{avg} = w_{avg} + \frac{1}{M} w_t^{[m]}$
8 $w_{slow} = w_{slow} + a_{slow}(w_{avg} - w_{slow})$;
9 $w_{fast} = w_{fast} + a_{fast}(w_{avg} - w_{fast})$;
10 **for** $m = 1$ *to* M **do**
11 $\quad$ With probability $max(0.0, 1 - \frac{w_{fast}}{w_{slow}})$ add a random particle to the set X_t;
12 $\quad$ **if** *the addition of a random particle failed* **then**
13 $\quad\quad$ Sample a particle $x_t^{[m]}$ from the set $\overline{X}_t$ with probability proportional to weight $w_t^{[m]}$;
14 $\quad\quad$ Add $x_t^{[m]}$ to X_t;
15 $x_t^{final} = \frac{\sum_{m=1}^{M} x_t^{[m]} \cdot w_t^{[m]}}{\sum_{m=1}^{M} w_t^{[m]}}$;
16 **return** X_t, x_t^{final};

Όπως μπορούμε να παρατηρήσουμε το πρώτο μέρος του αλγορίθμου (βήματα 1-6) είναι πανομοιότυπο με τον απλό αλγόριθμο MCL (10): η δειγματοληψία νέων στάσεων γίνεται από προηγούμενα σωματίδια με χρήση του μοντέλου κίνησης και ο συντελεστής στάθμισης της σπουδαιότητα τους ορίζεται σύμφωνα με το μοντέλο μέτρησης.

Η διαφορά του συγκεκριμένου αλγορίθμου γίνεται αντιλυπτή στα ακόλουθα βήματα:

- *Βήμα 7*: Στο συγκεκριμένο βήμα ο αλγόριθμος υπολογίζει επανληπτικά την προσεγγιστική πιθανότητα μέτρησης όπως προκύπτει από την σχέση 3.5.22 που αναφέρθηκε προηγουμένως.

- *Βήματα 8-9*: Στα συγκεκριμένα βήματα ο αλγόριθμος διατηρεί έναν βραχυπρόθεσμο και έναν μακροπρόθεσμο μέσο όρο για την πιθανότητα μέτρησης. Προκειμένου ο αλγόριθμος να λειτουργήσει σωστά απαιτείται να ισχύει η σχέση $0 \leq a_{slow} \ll a_{fast}$. Οι παράμετροι a_{slow} και a_{fast} είναι ρυθμοί μείωσης για τα εκθετικά φίλτρα που υπολογίζουν την εκτίμηση του βραχυπρόθεσμου και του μακροπρόθεσμου μέσου όρου αντίστοιχα.

- *Βήμα 13*: Στο συγκεκριμένο βήμα, κατά την διαδικασία της αναδειγματοληψίας, προστίθεται στο σύνολο X_t ένα τυχαίο σωματίδιο με πιθανότητα $max(0.0, 1.0 - \frac{w_{fast}}{w_{slow}})$. Στην περίπτωση που δεν προστεθεί κάποιο τυχαίο σωματίδιο, η αναδειγματοληψία προχωράει με τον γνωστό τρόπο όπως και στην περίπτωση του απλού MCL αλγορίθμου (10).

Η πιθανότητα προσθήκης ενός τυχαίου σωματιδίου λαμβάνει υπόψη την απόκλιση μεταξύ του βραχυπρόθεσμου και του μακροπρόθεσμου μέσου όρου της πιθανότητας μέτρησης. Αν η βραχυπρόθεσμη πιθανότητα είναι μεγαλύτερη ή ίση με την μακροπρόθεσμη πιθανότητα ($w_{fast} \geq w_{slow}$), δεν προστίθεται κάποιο τυχαίο δείγμα. Όμως, αν η βραχυπρόθεσμη πιθανότητα είναι μικρότερη από την μακροπρόθεσμη ($w_{fast} < w_{slow}$), τότε προστίθενται τυχαία δείγματα σε αναλογία με το πηλίκο των συγκεκριμένων τιμών ($\frac{w_{fast}}{w_{slow}}$). Με αυτόν τον τρόπο μια ξαφνική μείωση της πιθανότητας μέτρησης οδηγεί σε αύξηση των τυχαίων σωματιδίων. Η εκθετική εξομάλυνση εξουδετερώνει τον κίνδυνο να θεωρήσουμε εσφαλμένο τον στιγμιαίο θόρυβο του αισθητήρα ως κακό αποτέλεσμα του εντοπισμού θέσης.

3.5.11 Δειγματοληψία KLD: προσαρμογή του μεγέθους των συνόλων δειγμάτων

Το μέγεθος που έχουν τα σύνολα των δειγμάτων που χρησιμοποιούνται για την αναπαράσταση των πεποιθήσεων είναι μία σημαντική παράμετρος για την αποδοτικότητα του φίλτρου σωματιδίων. Για να αποφύγει κάποιος την απόκλιση λόγω εξάντλησης των δειγμάτων στον MCL, πρέπει να επιλέξει μεγάλα σύνολα δειγμάτων έτσι ώστε να επιτρέψει σε ένα ρομπότ να αντιμετωπίσει τόσο το πρόβλημα του καθολικού εντοπισμού θέσης όσο και το πρόβλημα της παρακολούθησης θέσης. Κάτι τέτοιο όμως σε ορισμένες περιπτώσεις μπορεί να ισοδυναμεί με σπατάλη υπολογιστικών πόρων. Παρότι ένας μεγάλος αριθμός σωματιδίων ενδέχεται να είναι απαραίτητος για να αναπαρασταθεί με ακρίβεια η πεποίθηση κατά τα πρώτα στάδια του καθολικού εντοπισμού θέσης, ένα μικρό ποσοστό των σωματιδίων είναι αρκετό για την παρακολούθηση της στάσης του ρομπότ, από την στιγμή που το τελευταίο γνωρίζει που βρίσκεται.

Η δειγματοληψία KLD είναι μια παραλλαγή του Προσαρμοστικού αλγορίθμου εντοπισμού θέσης Monte Carlo που προσαρμόζει τον αριθμό των σωματιδίων σε σχέση με τον χρόνο. Το όνομα *δειγματοληψία KLD* προέρχεται από την απόκλιση *Kullback-Leibler* η οποία αποτελεί ένα μέτρο της διαφοράς μεταξύ δύο κατανομών πιθανότητας. Η ιδέα πίσω από την δειγματοληψία KLD είναι να προσδιορίσουμε τον αριθμό των σωματιδίων που παράγονται σε κάθε βήμα του αλγορίθμου με βάση ένα στατιστικό φράγμα n_χ για την ποιότητα της βασισμένης σε δείγματα προσέγγισης της πραγματικής εκ των υστέρων κατανομής $bel(x_t)$. Πιο συγκεκριμένα, σε κάθε επανάληψη του φίλτρου σωματιδίων, η δειγματοληψία KLD προσαρμόζει τον αριθμό των δειγμάτων έτσι ώστε, με πιθανότητα $1-\delta$, το σφάλμα μεταξύ της πραγματικής εκ των υστέρων κατανομής και της βασισμένης σε δείγματα προσέγγισης (δηλαδή του συνόλου δειγμάτων X_t που παράγεται σε κάθε βήμα του αλγορίθμου) να είναι μικρότερο από ε. Η ιδέα αυτή υλοποιείται με τον ακόλουθο αλγόριθμο:

Algorithm 10: Προσαρμοστικός Αλγόριθμος Εντοπισμού Θέσης Monte Carlo με χρήση δειγματοληψίας KLD

Input: Σύνολο: X_{t-1} το οποίο αντιπροσωπεύει την πεποίθηση $bel(x_{t-1})$, Μέτρηση Ελέγχου: u_t, Μέτρηση απόστασης: z_t, Χάρτης: m, Όρια: ε,δ , Μέγεθος κάδου: Δ, Αριθμός σωματιδίων στο χρονικό βήμα t-1: N_{t-1}, Ελάχιστος αριθμός σωματιδίων: $n_{\chi_{min}}$

1 $\overline{X}_t := \emptyset, X_t := \emptyset, n = 0, n_\chi = 0, k = 0;$

2 static w_{slow}, w_{fast};

3 **for** $n = 1$ *to* N_{t-1} **do**

  // Sampling: Predict next state

4 $x_t^{[n]} = sample\_motion\_model(u_t, x_{t-1}^{[n]});$

  // Compute importance weight

5 $w_t^{[n]} = measurement\_model(z_t, x_t^{[n]}, m)$;

  // Insert sample into temporary sample set

6 $\overline{X}_t = \overline{X}_t + < x_t^{[m]}, w_t^{[m]} >;$

  // Update normalization factor

7 $w_{avg} = w_{avg} + \frac{1}{M} w_t^{[m]};$

8 $w_{slow} = w_{slow} + a_{slow}(w_{avg} - w_{slow});$

9 $w_{fast} = w_{fast} + a_{fast}(w_{avg} - w_{fast});$

10 **for** $n = 1$ *to* N_{t-1} **do**

  // Resampling process

11 With probability $max(0.0, 1 - \frac{w_{fast}}{w_{slow}})$ add a random particle $x_t^{[n]}$ to the set X_t;

12 **if** *the addition of a random particle failed* **then**

13 Sample a particle $x_t^{[n]}$ from the set $\overline{X}_t$ with probability proportional to weight $w_t^{[m]}$;

14 Add $x_t^{[n]}$ to X_t;

15 **if** $x_t^{[n]}$ *falls into empty bin* b **then**

  // Update number of bins with support

16 $k = k + 1;$

  // Mark bin

17 $b :=$ non-empty;

18 **if** $n \geq n_{\chi_{min}}$ **then**

  // Update number of desired sampled

19 $n_\chi = \frac{k-1}{2\varepsilon}\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta}\}^3;$

20 $n = n + 1;$

21 **if** $n \geq n_\chi$ *and* $n \geq n_{\chi_{min}}$ **then**

  // Set the final number of particles generated

22 $N_t = n_\chi;$

  // Stop the resampling process

23 break;

24 $x_t^{final} = \frac{\sum_{m=1}^{M} x_t^{[m]} \cdot w_t^{[m]}}{\sum_{m=1}^{M} w_t^{[m]}};$

25 **return** X_t, x_t^{final};

'Οπως μπορούμε να παρατηρήσουμε, ο συγκεκριμένος αλγόριθμος δέχεται ως είσοδο το προηγούμενο σύνολο δειγμάτων X_{t-1}, τον στατικό χάρτη m, την τελευταία μέτρηση ελέγχου u_t και την τελευταία μέτρηση απόστασης z_t. Επιπλέον ο αλγόριθμός απαιτεί τις στατικές παραμέτρους ε και δ που αφορούν το σφάλμα της προσέγγισης αλλά και τον ελάχιστο αριθμό σωματιδίων $n_{\chi_{min}}$ που απαιτείται να παράγονται σε κάθε χρονικό βήμα και ο οποίος ορίζεται από τον χρήστη.

Τα πρώτα βήματα του αλγορίθμου (βήματα 1-9) είναι πανομοιότυπα με τα βήματα του Προσαρμοστικού Αλγορίθμου εντοπισμού θέσης Monte Carlo 16 που αναλύσαμε προηγουμένως. Ουσιαστικά ο αλγόριθμος διαφοροποιείται στα βήματα 10 εως 23 όπου πραγματοποιείτα η διαδικασία της αναδειγματοληψίας και η εισαγωγή των νέων

σωματίδιων στο σύνολο X_t. Η διαδικασία της αναδειγματοληψίας υλοποιείται με τον ίδιο τρόπο όπως ακριβώς την υλοποιεί και ο Προσαρμοστικός Αλγόριθμος εντοπισμού θέσης Monte Carlo 16 με μία σημαντική διαφορά. Η διαφορά αυτή έγγυται στο γεγονός πως νέα σωματίδια παράγονται μέχρι ότου να ικανοποιηθεί το στατιστικό φράγμα n_χ . Το φράγμα αυτό βασίζεται στον όγκο του χώρο καταστάσεων που καλύπτεται από σωματίδια. Ο όγκος αυτός μετριέται με ένα ιστόγραμμα ή πλέγμα H με υπέρθεση πάνω από τον τρισδιάστατο χώρο των καταστάσεων. Κάθε κλιμάκιο b στο ιστόγραμμα H είτε είναι κενό είτε περιέχει τουλάχιστον ένα σωματίδιο. Αρχικά κάθε κλιμάκιο b είναι κενό. Στα βήματα 15-23 υλοποιείται η βασική ιδέα της δειγματοληψίας KLD. Αν το σωματίδιο που μόλις έχει παραχθεί μέσω της διαδικασίας αναδειγματοληψίας (βήματα 11-14) αντιστοιχεί σε κενό κλιμάκιο του ιστογράμματος, τότε αυξάνεται ο αριθμός k των κλιμακίων που δεν είναι κενά και το κλιμάκιο b σημειώνεται ως μη κενό. Επομένως, το k μετράει τον αριθμό των κλιμακίων του ιστογράμματος που περιέχουν τουλάχιστον ένα σωματίδιο. Ο αριθμός αυτός παίζει καίριο ρόλο στο στατιστικό φράγμα που προσδιορίζεται στο βήμα 19. Η ποσότητα n_χ παρέχει ουσιαστικά τον αριθμό των σωματιδίων που απαιτούνται για να μην ξεπεραστεί το στατιστικό φράγμα. Αξίζει να σημειωθεί ότι για δεδομένο ε, το n_χ είναι κατά κύριο λόγο γραμμικό ως προς τον αριθμό k των μη κενών κλιμακίων (ο δεύτερος μη γραμμικός όρος γίνεται αμελητέος καθώς το k αυξάνεται). Ο όρος $z_{1-\delta}$ βασίζεται στην παράμετρο δ. Αντιπροσωπεύει το ανώτερο 1-δ ποσοστημόριο της τυπικής κανονικής κατανομής. Οι τιμές του $z_{1-\delta}$ για τυπικές τιμές του δ μπορούν να βρεθούν εύκολα σε τυποποιημένους στατιστικούς πίνακες. Ο αλγόριθμος παράγει νέα σωματίδια μέχρι ο αριθμός τους n να υπερβεί το n_χ και ένα οριζόμενο από το χρήστη ελάχιστο $n_{\chi_{min}}$. Όπως μπορεί να διαπιστωθεί, το κατώφλιο n_χ χρησιμεύει ως κινούμενος στόχος για το n. Όσο περισσότερα δείγματα n παράγονται, τόσο περισσότερα κλιμάκια k στο ιστόγραμμα δεν είναι κενά και τόσο μεγαλύτερο είναι το κατώφλιο n_x. Στην πράξη ο αλγόριθμος τερματίζει με βάση τον ακόλουθο συλλογισμό: στα πρώτα στάδια της αναδειγματοληψίας το k αυξάνεται σχεδόν με κάθε νέο δείγμα επειδή ουσιαστικά όλα τα κλιμάκια είναι κενά. Η αύξηση του k έχει ως αποτέλεσμα μία αύξηση του κατωφλίου n_χ. Ωστόσο, με την πάροδο του χρόνου, ολοένα και περισσότερα κλιμάκια δεν είναι κενά και το n_χ αυξάνεται μόνο περιστασιακά. Αφού το n αυξάνεται με κάθε νέο δείγμα, τελικά θα γίνει ίσο με το n_χ και η δειγματοληψία θα σταματήσει. Η χρονική στιγμή που θα συμβεί αυτό εξαρτάται από την ποιότητα της προσέγγισης της εκ των υστέρων κατανομής. Όσο πιο διασκορπισμένα είναι τα σωματίδια, τόσο περισσότερα κλιμάκια γεμίζουν και τόσο μεγαλύτερο είναι το κατώφλιο n_χ. Κατά την διάρκεια της παρακολούθησης θέσης, η δειγματοληψία KLD παράγει λιγότερα δείγματα αφού τα σωματίδια είναι συγκεντρωμένα σε ένα μικρό αριθμό διαφορετικών κλιμακίων. Σημαντική παρατήρηση είναι πως το ιστόγραμμα δεν έχει καμία επίδραση στην ίδια την κατανομή των σωματιδίων. Ο μοναδικός σκοπός του είναι να μετρήσει την πολυπλοκότητα, ή όγκο της πεποίθησης. Το πλέγμα απορρίπτεται στο τέλος κάθε επανάληψης του αλγορίθμου.

3.6 Παρακολούθηση Διαδρομής κινητού ρομπότ

3.6.1 Αλγόριθμος Pure Pursuit

Ο Pure Pursuit είναι ένας ευρέως χρησιμοποιούμενος αλγόριθμος για την παρακολούθηση διαδρομής σε κινητά ρομπότ. Θεωρεί μία διαδρομή P_t ως μία διατεταγμένη λίστα σημείων $P_t = \{p_0, p_1, p_2, \dots\}$ όπου $p_i = (x_i, y_i) \in P$. Σκοπός του αλγόριθμου είναι να υπολογίσει σε κάθε χρονική στιγμή t την κατάλληλη γραμμική και γωνιακή ταχύτητα που χρειάζεται να έχει το κινητό ρομπότ προκειμένου να κινηθεί πάνω στην διαδρομή P_t. Κάτι τέτοιο αποτυπώνεται μαθηματικά με μια συνάρτηση f η οποία αντιστοιχίζει την διαδρομή P_t σε κατάλληλη γραμμική και γωνιακή ταχύτητα (u_t, w_t) του κινητού ρομπότ.

$$(u_t, w_t) = f(P_t) \tag{3.6.1}$$

Τα βήματα του συγκεκριμένου αλγορίθμου είναι τα εξής:

- Ο αλγόριθμος αρχικά καθορίζει το σημείο $p_r = (x_r, y_r)$ της διαδρομής P_t που έχει την μικρότερη απόσταση σε σχέση με την τρέχουσα θέση του ρομπότ. Στην συνέχεια χρησιμοποιώντας μία δοθείσα απόσταση L η οποία ονομάζεται *Lookahead Distance* υπολογίζει την θέση ενός νέου σημείου p_l το οποίο ονομάζεται *lookahead point*. Το σημείο αυτό ορίζεται ως το πρώτο σημείο $p_i = (x_i, y_i)$ της διαδρομής P_t το οποίο βρίσκεται μακριά από το p_r σε απόσταση τουλάχιστον ίση με L. Κάτι τέτοιο εκφράζεται μαθηματικά μέσω της ακόλουθης σχέσης:

$$dist(p_i) = \sqrt{(x_r - x_i)^2 + (y_r - y_i)^2}$$
$$p_l = p_i \in P_t, s.t. \begin{cases} dist(p_{i-1}) < L \\ dist(p_i) \geq L \end{cases} \tag{3.6.2}$$

- Ο αλγόριθμος στην συνέχεια υπολογίζει την καμπυλότητα k ενός τόξου το οποίο έχει τις εξής ιδιότητες:
  - Το τόξο διέρχεται από το σημείο p_l που υπολογίσθηκε προηγουμένως.
  - Το κέντρο του τόξου βρίσκεται στον πλευρικό άξονα y του ρομπότ σε απόσταση $g_y + d$ από την θέση του ρομπότ. Η απόσταση g_y ορίζεται ως η απόσταση μεταξύ της θέσης του ρομπότ και του σημείου p_l κατά μήκος του πλευρικού άξονα y του ρομπότ ενώ η απόσταση d ορίζεται ως η εναπομείνασα απόσταση μέχρι το κέντρο του τόξου.

    Μαθηματικά, οι παραπάνω ιδιότητες περιγράφονται με τις ακόλουθες εξισώσεις:

$$r = |gy| + d \tag{3.6.3}$$

$$d^2 + gx^2 = r^2 \tag{3.6.4}$$

    Λύνοντας την εξίσωση 3.6.3 ως προς d και αντικαθιστώντας την στην εξίσωση 3.6.4 προκύπτει η ακόλουθη σχέση για την ακτίνα r του τόξου.

$$\begin{gathered} (r - |gy|)^2 + gx^2 = r^2 \\ r^2 + |gy|^2 - 2 \cdot r \cdot |gy| + gx^2 = r^2 \\ r^2 + L^2 - 2 \cdot r \cdot gy = r^2 \\ r = \frac{L^2}{2 \cdot |gy|} \end{gathered} \tag{3.6.5}$$

    Έτσι η καμπυλότητα του τόξου η οποία ορίζεται ως η αντίστροφη ποσότητα της ακτίνας του προκύπτει από την ακόλουθη εξίσωση:

$$k = \frac{2 \cdot |gy|}{L^2} \tag{3.6.6}$$

    όπου:

    * k είναι η καμπυλότητα του τόξου που απαιτείται ώστε το ρομπότ, κινούμενο πάνω στο συγκεκριμένο τόξο, να φτάσει από την τωρινή του στάση στο σημείο p_l.
    * gy είναι η συντεταγμένη του σημείου p_l στον πλευρικό άξονα y του ρομπότ θεωρώντας ως σύστημα συντεταγμένων το τοπικό σύστημα συντεταγμένων του ρομπότ.
    * L είναι η επιθυμητή απόσταση μεταξύ των σημείων p_r και p_l η οποία είναι σταθερή και ορίζεται από τον χρήστη.

- Τελικώς, ο αλγόριθμος μέσω της καμπυλότητας k και την γραμμικής ταχύτητας u_t του ρομπότ την χρονική στιγμή t, υπολογίζει την γωνιακή ταχύτητα w_t που πρέπει να έχει το το ρομπότ ώστε να το κινηθεί πάνω στο συγκεκριμένο τόξο καμπυλότητας k και να φτάσει στο σημείο p_l. Η τελική γωνιακή ταχύτητα του ρομπότ δίνεται από την ακόλουθη σχέση:

$$w_t = k \cdot u_t \tag{3.6.7}$$

Όπως μπορούμε να παρατηρήσουμε οι κύριες παράμετροι του αλγορίθμου είναι το *lookahead distance* L και η επιθυμητή γραμμική ταχύτητα u_t του ρομπότ σε κάθε χρονική στιμγή t. Γνωρίζοντας τις δύο αυτές παραμέτρους ο Pure Pursuit αλγόριθμος μπορεί να υπολογίσει σε κάθε χρονική στιγμή την γωνιακή ταχύτητα που πρέπει να έχει το ρομπότ ώστε να φτάσει στο επόμενο *lookahead point* p_l με βάση τις σχέσεις 3.6.6, 3.6.7 που αναλύσαμε

προηγουμένως. Αξίζει να σημειωθεί ότι βρίσκοντας το *lookahead point* p_l η ποσότητα gy στην σχέση 3.6.6 μπορεί πολύ εύκολα να υπολογισθεί με την γνώση της τρέχουσας θέσης του ρομπότ χωρίς την χρήση κάποιας επιπλεόν πληροφορίας.

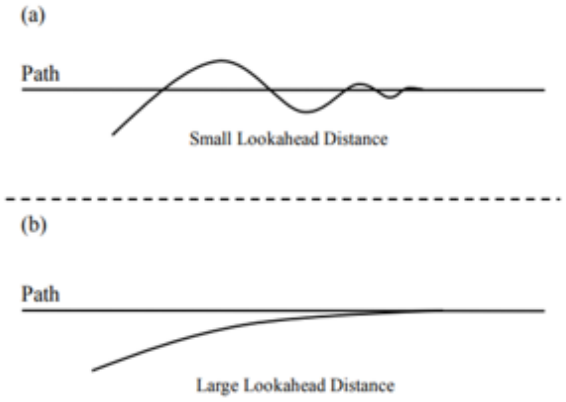

Figure 3.6.1: Επίδραση της απόστασης L στην συμπεριφορά του ρομπότ

Στην πράξη, η απόσταση L μεταξύ των σημείων p_r και p_l ρυθμίζεται κατάλληλα ώστε να επιτευχθεί ένα ικανοποιητικό αντιστάθμισμα μεταξύ των ταλαντώσεων του ρομπότ γύρω από το μονοπάτι που προκαλούνται όταν η απόσταση L έχει χαμηλές τιμές και την αργή σύγκλιση η οποία οδηγεί σε αποκλίσεις του ρομπότ από την διαδρομη όταν η απόσταση L έχει υψηλές τιμές. Αυτή η επίδραση της απόστασης L στην συμπεριφορά του ρομπότ μπορεί να φανεί και από την εικόνα 3.6.1.

Μειονεκτήματα του Αλγορίθμου

Τα δύο κύρια μειονεκτήματα του παραπάνω αλγορίθμου είναι τα ακόλουθα:

- Σε περιοχές όπου η διαδρομή παρουσιάζει υψηλή καμπυλότητα παρατηρείται αρκετές φορές overshoot ή undershoot συμπεριφορά με αποτέλεσμα να υπάρχουν αποκλίσεις από την διαδρομή ακόμα και με κατάλληλη ρύθμιση της απόστασης L.
- Ο αλγόριθμος δεν καθορίζει το πως θα πρέπει να υπολογίζεται η γραμμική ταχύτητα u_t του ρομπότ σε κάθε χρονικό βήμα. Ενώ κάτι τέτοιο μπορεί να φαίνεται ως πλεονέκτημα καθώς επιτρέπει μεγάλη ευελιξία επιλέγοντας διαφορετικά προφίλ για την γραμμική ταχύτητα, στην πραγματικότητα χωρίς κάποια συγκεκριμένη μέθοδο όλες οι παραλλαγές του αλγορίθμου *Pure Pursuit* που έχουν υλοποιηθεί στην πράξη χρησιμοποιούν σταθερή επιθυμητή γραμμική ταχύτητα. Κάτι τέτοιο δεν είναι απόλυτα ασφαλές και απαιτείται ένας προσαρμοστικός μηχανισμός που θα μεταβάλλει την γραμμική ταχύτητα με γνώμονα την ασφάλεια του ρομπότ ανάλογα με τις συνθήκες.

Στην συνέχεια παρουσιάζεται μία παραλλαγή του παραπάνω αλγορίθμου η οποία έχει ως στόχο να ξεπεράσει τους περιορισμούς που αναφέρθηκαν.

3.6.2 Αλγόριθμος Regulated Pure Pursuit

Ο Regulated Pure Pursuit αλγόριθμος έχει σχεδιαστεί για υπηρεσιακά και βιομηχανικά κινητά ρομπότ που λειτουργούν σε περιορισμένα και μερικώς παρατηρήσιμα περιβάλλοντα του πραγματικού κόσμου. Παρέχει μεθόδους για την προσαρμογή της γραμμικής ταχύτητας του ρομπότ ανάλογα με τις τρέχουσες συνθήκες. Και αυτό γιατί τα ρομποτικά συστήματα δεν μπορούν απλά να περνούν από διαδρόμους και εγκαταστάσεις έχοντας μέγιστη γραμμική ταχύτητα χωρίς έλεγχο. Οι μέθοδοι αυτοί αποτελούν γραμμικές συναρτήσεις ρύθμισης κόστους που παρέχουν συμπεριφορά υψηλής ποιότητας σε πολλά πραγματικά περιβάλλοντα στα οποία λειτουργούν τα κινητά ρομπότ. Επιπλέον, προέρχονται από κοινές απαιτήσεις για μείωση της ταχύτητας του ρομπότ όταν υπάρχουν απότομες στροφές ή όταν το ρομπότ λειτουργεί σε περιορισμένης χωρητικότητας περιοχές όπως για παράδειγμα περιοχές με πολλά εμπόδια.

Τα βήματα του συγκεκριμένου αλγορίθμου είναι τα εξής:

- Ο αλγόριθμος σε κάθε χρονική στιμγή t λαμβάνει ως είσοδο μία επιθυμητή διαδρομή P_t την οποία πρέπει να ακολουθήσει το ρομπότ καθώς και μία επιθυμητή γραμμική ταχύτητα $u_t^{desired}$ η οποία ορίζεται από τον χρήστη. Η επιθυμητή διαδρομή δίνεται ως μια διατεταγμένη λίστα διαδοχικών σημείων $\{p_0, ..., p_j, p_r, ..., p_n\}$ στο επίπεδο x-y όπου κινείται το ρομπότ.

- Στην συνέχεια ο αλγόριθμος εντοπίζει το κοντινότερο σημείο p_r της διαδρομής σε σχέση με την τρέχουσα θέση του ρομπότ και αφαιρεί από το μονοπάτι όλα τα σημεία $\{p_0, ..., p_j\}$ που βρίσκονται πριν από το σημείο p_r αφού πλέον αυτά δεν χρησιμεύουν.

- Για τα εναπομείναντα σημεία της διαδρομής, μετατρέπει τις συντεταγμένες του κάθε σημείου ώστε να αναφέρονται στο τοπικό σύστημα συντεταγμένων του ρομπότ. Έτσι προκύπτει το μονοπάτι $\{p'_r, ..., p'_n\}$:

- Έπειτα ο αλγόριθμος υπολογίζει το *lookahead distance* L_t μέσω της ακόλουθης σχέσης:

$$L_t = u_t \cdot l_t \tag{3.6.8}$$

όπου u_t είναι η γραμμική ταχύτητα που έχει το ρομπότ την χρονική στιγμή t και l_t είναι ένα κέρδος.

Σύμφωνα με την παραπάνω σχέση το L_t είναι ανάλογο της τρέχουσας γραμμικής ταχύτητας του ρομπότ την χρονική στιγμή t. Αυτό μπορεί να ερμηνευθεί από το γεγονός πως όταν η γραμμική ταχύτητα του ρομπότ είναι μεγάλη, τότε είναι πιο ασφαλές να θέτουμε μεγάλο L_t έτσι ώστε το ρομπότ να τείνει να κινείται ευθεία. Αντίθετα όταν η γραμμική ταχύτητα του ρομπότ είναι μικρή, όπως για παράδειγμα σε περιπτώσεις απότομων στροφών, τότε είναι προτιμότερο να θέτουμε μικρό L_t έτσι ώστε το ρομπότ να τείνει να στρίβει και να έχει την δυνατότητα να ακολουθήσει κλειστές στροφές.

- Έχοντας υπολογίσει το *lookahead distance* L_t ο αλγόριθμος, όπως και στην περίπτωση του απλού Pure Pursuit αλγορίθμου, εντοπίζει το *lookahead point* p_l μέσω της σχέσης 3.6.2.

- Στην συνέχεια ο αλγόριθμος υπολογίζει την καμπυλότητα k του τόξου χρησιμοποιώντας την σχέση 3.6.6, όπως και στην περίπτωση του απλού Pure Pursuit αλγορίθμου.

- Έπειτα ο αλγόριθμος χρησιμοποιεί δύο ευρετικές συναρτήσεις για την επεξεργασία της επιθυμητής γραμμικής ταχύτητας $u_t^{desired}$. Οι συναρτήσεις αυτές ονομάζονται: *ευρετική συνάρτης καμπυλότητας* και *ευρετική συνάρτηση εγγύτητας*. Στην συνέχεια αναλύεται η κάθε συνάρτηση ξεχωριστά:

  - *Ευρετική καμπυλότητας*: Ο σκοπός της ευρετικής συνάρτησης καμπυλότητας είναι να επιβραδύνει το ρομπότ μειώνοντας την γραμμική του ταχύτητα κατά την διάρκεια απότομων στροφών σε μερικώς παρατηρήσιμα περιβάλλοντα όπως κατά την είσοδο ή έξοδο του από διαδρόμους. Κάτι τέτοιο επιτρέπει σημαντικά πιο ασφαλή κίνηση του ρομπότ κατά την πραγματοποίηση «τυφλών» στροφών. Ο τύπος της συνάρτησης αυτής δίνεται στην συνέχεια:

$$u'_t = \begin{cases} u_t^{desired} & k > T_k \\ \frac{u_t^{desired}}{r_{min} \cdot k} & k \leq T_k \end{cases} \tag{3.6.9}$$

    Η ευρετική συνάρτηση καμπυλότητας εφαρμόζεται στην επιθυμητή γραμμική ταχύτητα $u_t^{desired}$ του ρομπότ όταν η αλλαγή στην καμπυλότητα k είναι πάνω από ένα κατώτατο όριο T_κ.

  - *Ευρετική εγγύτητας*: Η ευρετική συνάρτηση εγγύτητας εφαρμόζεται στην επιθυμητή γραμμική ταχύτητα $u_t^{desired}$ του ρομπότ όταν αυτό πλησιάζει σε δυναμικά εμπόδια ή σταθερές υποδομές. Ο σκοπός της είναι να επιβραδύνει το ρομπότ όταν αυτό κινείται σε μερικώς παρατηρήσιμα περιβάλλοντα όπου η πιθανότητα σύγκρουσης είναι ιδιαίτερα υψηλή. Η μείωση της ταχύτητας του ρομπότ όταν αυτό βρίσκεται κοντά σε σταθερή υποδομή μειώνει την πιθανότητα σύγκρουσης καθώς μειώνεται ο αντίκτυπος που μπορεί να έχουν μικρές αποκλίσεις από το μονοπάτι σε στενές περιοχές. Παράλληλα, η μείωση της ταχύτητας των ρομπότ όταν βρίσκονται κοντά σε δυναμικές οντότητες, όπως για παράδειγμα άνθρωποι που κινούνται στον χώρο, είναι μια κοινή απαίτηση ασφάλειας που επιτρέπει στο ρομπότ να σταματά γρηγορότερα ώστε να αποτρέψει κάποια πιθανή σύγκρουση. Ο τύπος της ευρετικής συνάρτησης εγγύτητας δίνεται στην συνέχεια:

$$u'_t = \begin{cases} u_t^{desired} \cdot \frac{a \cdot d_0}{d_{prox}} & d_0 \leq d_{prox} \\ u_t^{desired} & d_0 \geq d_{prox} \end{cases} \tag{3.6.10}$$

Η ευρετική συνάρτηση εγγύτητας μειώνει την επιθυμητή γραμμική ταχύτητα $u_t^{desired}$ του ρομπότ κατά τον λόγο $\frac{a \cdot d_o}{d_{max}}$ όπου d_{prox} είναι η απόσταση εγγύτητας και d_o είναι η τρέχουσα απόσταση του ρομπότ από το κοντινότερο εμπόδιο. Όταν η τρέχουσα απόσταση του ρομπότ από το κοντινότερο εμπόδιο γίνει μικρότερη ή ίση της απόστασης εγγύτητας τότε εφαρμόζεται η παραπάνω ευρετική συνάρτηση η οποία προσπαθεί να επιβραδύνει το ρομπότ. Η τιμή της απόστασης εγγύτητας d_{prox} καθορίζεται με βάση τις απαιτήσεις του εκάστοτε συστήματος ανάλογα με το πόσο κοντά στο ρομπότ μπορεί να βρίσκεται ένα εμπόδιο πριν το ρομπότ αρχίσει να επιβραδύνει μειώνοντας την ταχύτητα του. Επιπλέον το κέρδος $a \leq 1$ ορίζεται με τέτοιο τρόπο έτσι ώστε να ρυθμίζεται η απόκριση ανάλογα με το εκάστοτε σύστημα. Η τιμή του a καθορίζει με άλλα λόγια το πόσο επιθετική θα είναι η μείωση της ταχύτητας του ρομπότ. Μεγαλύτερη τιμή του α προκαλεί πιο γρήγορη μείωση της ταχύτητας του ρομπότ κοντά σε εμπόδια.

- Ο αλγόριθμος στην συνέχεια επιλέγει ως τελική επιθυμητή γραμμική ταχύτητα u'_t για το ρομπότ την ελάχιστη από τις δύο ταχύτητες που έχουν προκύψει από τις παραπάνω ευρετικές συναρτήσεις.
- Έπειτα ο αλγόριθμος υπολογίζει την γωνιακή ταχύτητα w_t του ρομπότ χρησιμοποιώντας την καμπυλότητα του τόξου k καθώς και την ρυθμισμένη γραμμική ταχύτητα u'_t και όχι την αρχική επιθυμητή γραμμική ταχύτητα $u_t^{desired}$. Η σχέση που δίνει την γωνιακή ταχύτητα w_t δίνεται στην συνέχεια:

$$w_t = k \cdot u'_t \tag{3.6.11}$$

- Οι ταχύτητες w_t και u'_t μεταφράζονται τελικώς μέσω των σχέσεων 3.4.5, 3.4.6, 3.4.7, 3.4.8 σε επιθυμητές γωνιακές ταχύτητες περιστροφής των δύο τροχών του ρομπότ.
- Το τελευταίο βήμα του αλγορίθμου είναι ο έλεγχος τρέχων ή επικείμενων συγκρούσεων. Έχοντας την γωνιακή ταχύτητα w_t και την γραμμική ταχύτητα u'_t ο αλγόριθμος προβάλει την κίνηση του ρομπότ στο μέλλον η οποία πραγματοποιείται πάνω σε ένα κυκλικό τόξο το οποίο προκύπτει από τις δύο παραπάνω ταχύτητες. Το κυκλικό τόξο προβάλλεται για συγκεκριμένη χρονική διάρκεια και τα σημεία του κυκλικού τόξου δειγματοληπτούνται ώστε να αντιστοιχούν σε κελιά του χάρτη m. Έπειτα γίνεται ο έλεγχος αν κάποιο από αυτά τα σημεία αυτά αντιστοιχεί σε κελί του χάρτη στο οποίο υπάρχει εμπόδιο και άρα έχει επισυμανθεί ως κατειλημμένο. Σε μία τέτοια περίπτωση το ρομπότ σταματάει την κίνηση του καθώς υπάρχει μεγάλος κίνδυνος σύγκρουσης με εμπόδιο.

3.7 Αλγόριθμος Σχεδιασμού Διαδρομής Dijkstra

Ο αλγόριθμος του Dijkstra είναι ένας ευρέως χρησιμοποιούμενος αλγόριθμος εύρεσης διαδρομής στην ρομποτική για την εύρεση της συντομότερης διαδρομής μεταξύ δύο κορυφών σε έναν γράφο.

Ο γράφος, ο οποίος στην περίπτωση μας αποτελεί τον χώρο λειτουργίας του ρομπότ, αναπαρίσταται ως μία συλλογή κορυφών και ακμών. Κάθε ακμή ενώνει δύο κορυφές και έχει ένα σχετικό βάρος ή κόστος. Το πρόβλημα της συντομότερης διαδρομής αναφέρεται στην εύρεση της συντομότερης διαδρομής από μία δεδομένη αρχική κορυφή προς μία άλλη τελική κορυφή. Στόχος είναι η ελαχιστοποίηση του συνολικού συσσωρευμένου βάρους ή κόστους της διαδρομής μεταξύ της αρχικής και της τελικής κορυφής. Στην περίπτωση μας η αρχική κορυφή αποτελεί την τρέχουσα θέση του ρομπότ ενώ η τελική κορυφή αντιστοιχεί στην θέση-στόχο στην οποία επιθυμεί τελικώς να φτάσει το ρομπότ.

Ο αλγόριθμος του Dijkstra χρησιμοποιεί μία άπληστη προσέγγιση για την επαναληπτική εξερεύνηση του γραφήματος, δημιουργώντας σταδιακά το δέντρο των συντομότερων διαδρομών. Η βασική ιδέα πίσω από τον αλγόριθμο του Dijkstra είναι πως το συντομότερο μονοπάτι προς μία κορυφή είναι γνωστό μόλις ο αλγόριθμος την επισκεφθεί εξασφαλίζοντας έτσι ότι τα μονοπάτια που ανακαλύπτονται αποτελούν την βέλτιστη λύση. Αυτή η ιδιότητα επιτυγχάνεται με την επιλογή της κορυφής με την μικρότερη τρέχουσα απόσταση σε κάθε επανάληψη, διασφαλίζοντας ότι ο αλγόριθμος εξερευνά τα μονοπάτια κατά σειρά αυξανόμενης απόστασης από την αρχική

κορυφή. Έτσι ο αλγόριθμος του Dijkstra εγγυάται την βέλτιστη λύση καθώς μόλις μία κορυφή χαρακτηριστεί ως επισκέψιμη, η τρέχουσα της απόσταση αποτελεί την μικρότερη διαδρομή από την αρχική κορυφή στην κορυφή αυτή.

Ο αλγόριθμος Dijkstra δίνεται στην συνέχεια:

Algorithm 11: Αλγόριθμος Dijkstra

Data: Γράφος $G(V, E, w)$, αρχική κορυφή s
Result: Μικρότερες αποστάσεις $D[]$ από την αρχική κορυφή s σε κάθε κορυφή

```
1  D[s] ← 0;
2  S ← ∅;
3  foreach u ∈ V do
4  |   D[u] ← ∞;
5  |   p[u] ← NULL;
6  while |S| < |V| do
7  |   u ∉ S : D[u] = min_{v∉S}{D[v]};
8  |   S ← S ∪ {u};
9  |   foreach neighbor v of u do
10 |   |   if D[v] > D[u] + w(u, v) then
11 |   |   |   D[v] ← D[u] + w(u, v);
12 |   |   |   p[v] ← u;
13 return dist[], p[];
```

Όπως μπορούμε να παρατηρήσουμε τα βήματα του αλγορίθμου είναι τα ακόλουθα:

- *Βήματα 1-5*: Στα συγκεκριμένα βήματα πραγματοποιείται η αρχικοποίηση του αλγορίθμου. Πιο αναλυτικά:
  - Ορίζεται η απόσταση της αρχικής κορυφής s στο μηδέν ($D[s] \leftarrow 0$) και όλων των υπόλοιπων κορυφών u του γραφήματος στο άπειρο ($D[u] \leftarrow \infty$).
  - Κατασκευάζεται ένα σύνολο S το οποίο αρχικοποιείται ως κενό ($S \leftarrow \emptyset$) και στο οποίο θα αποθηκεύονται οι κορυφές που ο αλγόριθμος έχει ήδη επισκεφθεί.
  - Κατασκευάζεται ένας πίνακας p όπου κάθε θέση του πίνακα αντιστοιχεί σε μία συγκεκριμένη κορυφή του γραφήματος. Σε κάθε θέση u του πίνακα αποθηκεύεται ένας δείκτης ο οποίος παραπέμπει στην κορυφή μέσω της οποίας ο αλγόριθμος επισκέφθηκε την συγκεκριμένη κορυφή u. Αρχικά κάθε θέση του πίνακα αρχικοποιείται στην τιμή NULL ($p[u] \leftarrow NULL$).
- *Βήματα 7-8*: Στα συγκεκριμένα βήματα πραγματοποιείται η εύρεση της κορυφής με την μικρότερη τρέχουσα απόσταση. Πιο αναλυτικά:
  - Επιλέγεται η κορυφή u με την μικρότερη τρέχουσα απόσταση $D[u]$ από τις κορυφές που ακόμα δεν έχει επισκεφθεί ο αλγόριθμος, δηλαδή τις κορυφές που δεν ανήκουν στο σύνολο S (στην πρώτη επανάληψη του αλγορίθμου η κορυφή αυτή θα είναι η αρχική κορυφή s).
  - Η κορυφή u προστίθεται στο σύνολο S ($S \leftarrow S \cup \{u\}$) αφού πλεόν αποτελεί μία επισκέψιμη κορυφή.
- *Βήματα 9-12*: Στα συγκεκριμένα βήματα πραγματοποιείται η ενημέρωση της τρέχουσας αποστάσης για κάθε γειτονική κορυφή v της τρέχουσας κορυφής u. Πιο αναλυτικά:
  - Υπολογίζεται η απόσταση προκειμένου ο αλγόριθμος να φτάσει στην κορυφή v μέσω της κορυφής u. Η απόσταση αυτή ισούται με το άθροισμα της απόστασης της τρέχουσας κορυφής $D[u]$ και το βάρος της ακμής $w(u, v)$ που συνδέει την τρέχουσα κορυφή u με την γειτονική της κορυφή v.
  - Εάν η υπολογιζόμενη απόσταση $D[u] + w(u, v)$ είναι μικρότερη από την μέχρι πρότινος ελάχιστη απόσταση $D[v]$ της γειτονικής κορυφής v τότε η απόσταση της γειτονικής κορυφής ενημερώνεται ($D[v] \leftarrow D[u] + w(u, v)$) ενώ επιπλέον ενημερώνεται και η αντίστοιχη θέση του πίνακα p ($p[v] = u$).

Τα βήματα 7-8 και 9-12 επαναλαμβάνονται μέχρις ότου ο αλγόριθμος επισκεφθεί όλες τις κορυφές τους γραφήματος και έτσι ισχύει η σχέση $|S| = |V|$. Ξεκινώντας από την τελική κορυφή και ακολουθώντας προς τα πίσω τους δείκτες όπως υπαγορεύονται από τον πίνακα p μέχρι την αρχική κορυφή s μπορεί κανείς να εξάγει την συντομότερη διαδρομή από την αρχική στην τελική κορυφή η οποία αποτελείται από την αρχική και τελική κορυφή αλλά και όλες τις ενδιάμεσες κορυφές. Έτσι το σύνολο των κορυφών αυτών αποτελεί την συντομότερη διαδρομή την οποία στην συνέχεια το ρομπότ πρέπει να ακολουθήσει προκειμένου να φθάσει στην τελική θέση στόχο που επιθυμεί.

Chapter 4

Υλοποίηση και Πειράματα

4.1 Σενάριο Υλοποίησης

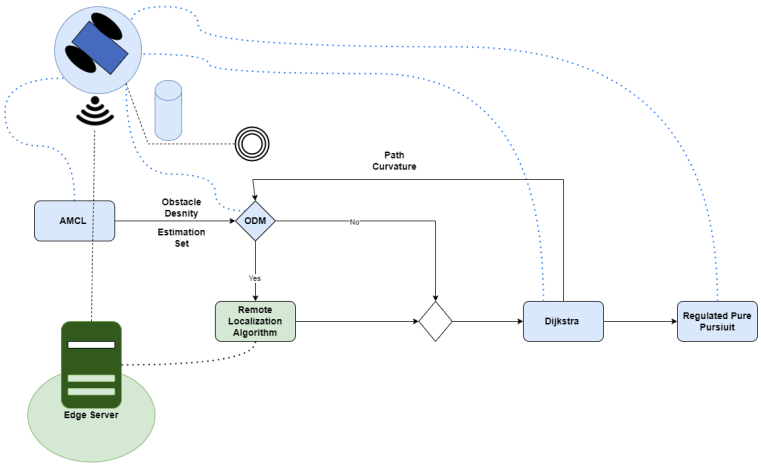

Figure 4.1.1: Επισκόπηση αρχιτεκτονικής. Τα στοιχεία που εκτελούνται τοπικά επισυνάπτονται με μπλέ χρώμα ενώ τα στοιχεία που εκτελούνται απομακρυσμένα με πράσινο χρώμα

Το σενάριο το οποίο μελετήθηκε στην παρούσα διπλωματική εργασία φαίνεται σχηματικά στην εικόνα 4.1.1. Το ρομπότ επιθυμεί να μετακινηθεί από μία γνωστή αρχική θέση A σε μια τελική θέση B. Τα βήματα που ακολουθεί προκειμένου να το επιτύχει αναλύονται στην συνέχεια:

- Αρχικά το ρομπότ εκτελεί τοπικά τον Προσαρμοστικό Αλγόριθμο Εντοπισμού Θέσης Monte Carlo με χρήση δειγματοληψίας KLD (25) προκειμένου σε κάθε χρονική στιγμή να έχει μία εκτίμηση της τρέχουσας στάσης του.
- Στην συνέχεια χρησιμοποιεί τον μηχανισμό απόφασης εκφόρτωσης (Offloading Decision Mechanism ή ODM) ,ο οποίος θα αναλυθεί στην συνέχεια, προκειμένου να αποφασίσει εάν είναι απαραίτητη η χρήση του απομακρυσμένου αλγορίθμου εντοπισμού θέσης. Σε περίπτωση που κάτι τέτοιο κριθεί αναγκαίο, το ρομπότ χρησιμοποιεί τον συγκεκριμένο αλγόριθμο προκειμένου να διορθώσει την εκτίμηση της στάσης του. Σε αντίθετη περίπτωση η εκτίμηση της στάσης του προέρχεται από τον τοπικά εκτελούμενο αλγόριθμο εντοπισμού θέσης. Ο απομακρυσμένος αλγόριθμος εντοπισμού θέσης επιλέχθηκε να είναι ο ORB-SLAM3 [3] ο οποίος εντοπίζει σε κάθε χρονική στιγμή την θέση και τον προσανατολισμό της κάμερας του ρομπότ. Έτσι το ρομπότ, γνωρίζωντας την σταθερή θέση της κάμερας σε σχέση με το κέντρικό του σημείο, μπορεί να έχει μία ακριβή εκτίμηση της στάσης του σε κάθε χρονική στιγμή. Στα πλαίσια της παρούσας διπλωματικής εργασίας, θεωρούμε ότι η εκτίμηση της στάσης του ρομπότ που προέρχεται από τον συγκεκριμένο αλγόριθμο είναι ακριβής και δεν περιέχει κάποιο σφάλμα.

- Η τελική εκτιμώμενη στάση του ρομπότ δίνεται στον αλγόριθμο Dijkstra (13) ο οποίος υπολογίζει την βέλτιστη διαδρομή από την τρέχουσα θέση του ρομπότ στην τελική θέση-στόχο. Σε αυτό το σημείο είναι σημαντικό να αναφερθεί πως λόγω υπολογιστικής πολυπλοκότητας, η βέλτιστη διαδρομή δεν επαναυπολογίζεται σε κάθε χρονικό βήμα. Αντίθετα ο επαναυπολογισμός της πραγματοποιείται με μία σταθερή συχνότητα η οποία μπορεί να ρυθμιστεί κατάλληλα από τον χρήστη. Όσο η βέλτιστη διαδρομή παραμένει η ίδια, το ρομπότ επιχειρεί να ακολουθήσει την συγκεκριμένη διαδρομή.
- Η τελική εκτιμόμενη στάση του ρομπότ μαζί με την βέλτιστη διαδρομή που υπολογίσθηκε στο προηγούμενο βήμα δίνεται στον Regulated Pure Pursuit αλγόριθμο ο οποίος υπολογίζει τις κατάλληλες ταχύτητες κίνησης w_t και u_t προκειμένου το ρομπότ να ακολουθήσει την συγκεκριμένη διαδρομή.

Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε χρονικό βήμα μέχρις ώτου το ρομπότ να καταφέρει να φτάσει στην τελική του θέση.

4.2 Μηχανισμός Απόφασης Εκφόρτωσης

Ο Μηχανισμός Απόφασης Εκφόρτωσης χρησιμοποιείται από το ρομπότ προκειμένου να αποφασίσει εάν υπάρχει η ανάγκη για την χρήση του πιο ακριβούς αλγορίθμου εντοπισμού θέσης ο οποίος εκτελείται απομακρυσμένα όντας πιο απαιτητικός υπολογιστικά. Ο απομακρυσμένος αλγόριθμος εντοπισμού θέσης ουσιαστικά χρησιμοποιείται από το ρομπότ για την διόρθωση της εκτίμησης της στάσης του η οποία έχει προκύψει από τον τοπικά εκτελούμενο αλγόριθμο εντοπισμού θέσης. Η λήψη μιάς τέτοιας απόφασης λαμβάνεται από τον μηχανισμό με βάση τις ακόλουθες τρεις μετρικές:

- Την αβεβαιότητα ως προς την ακριβή στάση του ρομπότ.
- Την «πυκνότητα» των εμποδίων.
- Την καμπυλότητα της βέλτιστης διαδρομής που πρέπει να ακολουθήσει το ρομπότ.

Στην συνέχεια αναλύεται ξεχωριστά η κάθε μία από τις τρεις αυτές μετρικές:

Αβεβαίοτητα ως προς την στάση του ρομπότ

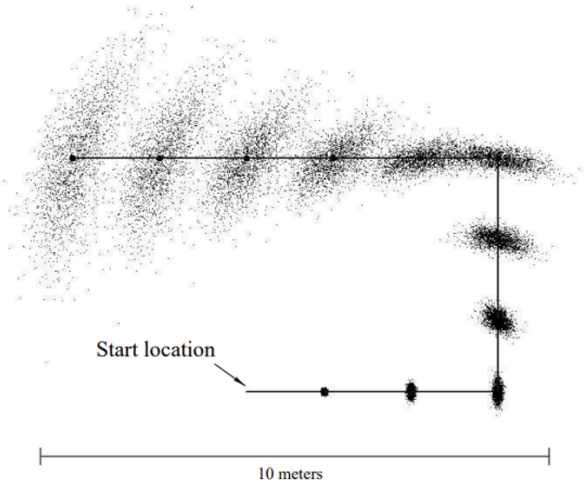

Figure 4.2.1: Αλγόριθμος δειγματοληψίας μοντέλου κίνησης με βάση την οδομετρία σε δράση

Θεωρούμε ως x_t την τρέχουσα εκτίμηση για την θέση του ρομπότ και ως $X_t^{est} \subset \mathbb{R}^2$ ένα σύνολο το οποίο ονομαζεται *estimation set* και το οποίο περιέχει όλες τις πιθανές θέσεις στις οποίες μπορεί να φτάσει το ρομπότ την χρονική στιγμή t ξεκινώντας από μία αρχική θέση x_0 με μία αρχική αβεβαιότητα X_0^{est}. Το πρόβλημα της εκτίμησης μίας συντηριτικής προσέγγισης για το X_t^{est} είναι ένα αρκετά δύσκολο υπολογιστικό πρόβλημα για συστήματα με γενική δυναμική που λειτουργούν σε περιορισμένα περιβάλλοντα όπως είναι τα ρομπότ. Έτσι, στην

περίπτωση μας, θεωρήσαμε πως μία καλή προσέγγιση για το σύνολο αυτό μπορεί να προκύψει από τον αλγόριθμο δειγματοληψίας του μοντέλου κίνησης μέσω της οδομετρίας (10) τον οποίο αναλύσαμε στο Κεφάλαιο 3. Ο συγκεκριμένος αλγόριθμος δέχεται ως είσοδο ένα σύνολο σωματιδίων και παράγει ένα νέο σύνολο σωματιδίων μέσω του μοντέλου κίνησης του ρομπότ το οποίο λαμβάνει υπόψιν του τις μετρήσεις της οδομετρίας προσομοιώνοντας τον θόρυβο που υπάρχει στις συγκεκριμένες μετρήσεις στις εξισώσεις του μοντέλου.

Στην εικόνα 4.2.1 μπορούμε να δούμε μία υπέρθεση των συνόλων δειγμάτων που έχουν παραχθεί με βάση των παραπάνω αλγόριθμο για πολλές χρονικές στιγμές ώστε να φανεί το μοντέλο κίνησης σε δράση. Μπορούμε εύκολα να παρατηρήσουμε πως η αβεβαιότητα αυξάνεται με την κίνηση του ρομπότ και τα δείγματα είναι διεσπαρμένα σε έναν αυξανόμενα μεγάλο χώρο.

Έτσι, περiκλύοντας το συγκεκριμένο σύνολο σωματιδίων μέσα σε ένα ορθογώνιο παραλληλόγραμμο μπορούμε να έχουμε μία προσέγγιση του πραγματικού συνόλου X_t^{est}. Όπως εξηγήσαμε και προηγουμένως, όσο μεγαλύτερη γίνεται η αβεβαιότητα ως προς την τρέχουσα στάση του ρομπότ, τόσο πιο διασκορπισμένα είναι τα σωματίδια στον επιπέδο x-y και άρα τόσο μεγαλύτερο είναι και το εμβαδόν του ορθογωνίου παραλληλόγραμμου που τα περικλύει. Έτσι μπορούμε να ορίσουμε το συγκεκριμένο εμβαδόν ως μία συνάρτηση κόστους $D_{uncertainty}(X_t)$ η οποία εκφράζει την αβεβαιότητα ως προς την ακριβή στάση του ρομπότ και η οποία προκύπτει με βάση το τρέχων σύνολο σωματιδίων X_t που έχει προκύψει από τον παραπάνω αλγόριθμο:

$$D_{uncertainty}(X_t) = E(X_t) \tag{4.2.1}$$

όπου η συνάρτηση E δέχεται ως είσοδο ένα σύνολο σωματιδίων και επιστρέφει το εμβαδόν του ορθογώνιου παραλληλογράμμου που τα περικλύει.

Διαισθητικά η τιμή του κόστους $D_{uncertainty}(X_t)$ αυξάνεται καθώς το ρομπότ κινείται λόγο της αβεβαιότητας που εισάγεται από τον θόρυβο που υπάρχει στην μέτρηση της στάσης του. Σε περιπτώσεις που η αβεβαιότητα αυτή ξεπεράσει ένα προκαθορισμένο ανεκτό όριο, απαιτείται η χρήση ενός πιο ακριβούς αλγορίθμου εντοπισμού θέσης προκειμένου το ρομπότ να μειώσει την αβεβαίοτητα που έχει για την ακριβή στάση του.

Πυκνότητα Εμποδίων

Η συγκεκριμένη μετρική εκτιμά την εγγύτητα του ρομπότ σε σχέση με τα εμπόδια που βρίσκονται στον χώρο. Ορίζουμε ως x_t την εκτίμηση της τρέχουσας θέσης του ρομπότ και ως Z_{obs} ένα σύνολο που περιέχει την θέση στο επίπεδο x-y των σημείων των εμποδίων τα οποια ανιχνεύονται από τον αισθητήρα απόστασης λέιζερ του ρομπότ. Κάθε τέτοιο σημείο συμβολίζεται ως z_{obs}. Έτσι ορίζουμε την ακόλουθη συνάρτηση κόστους $D_{ostbalce}(x_t)$ η οποία εκφράζει την "πυκνότητα" των εμποδίων σε σχέση με την τρέχουσα θέση x_t του ρομπότ:

$$D_{obstalce}(x_t) = \sum_{z_{obs} \in Z_{obs}} \exp\left(-||x_t - z_{obs}||\right) \tag{4.2.2}$$

Διαισθητικά η τιμή του κόστους $D_{obstalce}(x_t)$ λαμβάνει υψηλές τιμές όταν το ρομπότ κινείται σε περιοχές του περιβάλλοντος όπου υπάρχουν αρκετά εμπόδια. Σε τέτοιες περιπτώσεις απαιτείται η χρήση ενός πιο ακριβούς αλγορίθμου εντοπισμού θέσης προκειμένου το ρομπότ να αποφύγει την σύγκρουση με κάποιο εμπόδιο.

Καμπυλότητα Διαδρομής

Λαμβάνοντας σε κάθε χρονική στιγμή την διαδρομή P_t ως μία διατεταγμένη λίστα σημείων $P_t = \{p_0, p_1, p_2, \dots\}$ όπου $p_i = (x_i, y_i) \in P$ η οποία έχει προκύψει από τον αλγόριθμο σχεδιασμού διαδρομής Dijkstra και κρατώντας τα πρώτα M σημεία της διαδρομής αυτής μπορούμε να ορίσουμε το κόστος D_{path} το οποίο ουσιαστικά εκφράζει την καμπυλότητα της διαδρομής P_t, όπως αυτή προκύπτει από τα πρώτα M σημεία της:

$$D_{path}(P_t) = \sum_{i=0}^{M-1} (||p_{i+1} - p_i||) - ||p_M - p_0|| \tag{4.2.3}$$

Διαισθητικά όταν το κόστος $D_{path}(P_t)$ λαμβάνει υψηλές τιμές, αυτό σημαίνει πως το ρομπότ καλείται να ακολουθήσει μία πιο δύσκολη διαδρομή, γεγονός που απαιτεί την χρήση ενός πιο ακριβή αλγορίθμου εντοπισμού θέσης προκειμένου το ρομπότ να μην παρεκκλίνει από την διαδρομή αυτή.

Τελική Απόφαση

Η τελική απόφαση για την χρήση του απομακρυσμένου αλγορίθμου εντοπισμού θέσης ORB-SLAM3 λαμβάνεται με βάση τις τιμές των $D_{uncertainty}(X_t)$, $D_{obstalce}(x_t)$, $D_{path}(P_t)$. Έτσι, σε περίπτωση που κάποια από τις τιμές αυτές ξεπεράσει ένα προκαθορισμένο όριο, το οποίο δίνεται από τον χρήστη, τότε ο απομακρυσμένος αλγόριθμος εντοπισμού θέσης ενεργοποιείται διορθώνοντας την τρέχουσα στάση του ρομπότ.

4.3 Πειράματα

Πείραμα 1

Στην συνέχεια παρουσιάζεται ένα συγκεκριμένο σενάριο προκειμένου να φανεί η χρησιμότητα του μηχανισμού απόφασης εκφόρτωσης. Πιο συγκεκριμένα. το ρομπότ επιχειρεί να μεταβεί από μία γνωστή αρχική θέση σε μία γνωστή τελική θέση κινούμενο σε ένα χώρο διαστάσεων 6m x 6m ο οποίος είναι περιτριγυρισμένος από τοίχους και στο εσωτερικό του οποίου είναι τοποθετημένα στατικά εμπόδια. Υποθέτουμε πως το ρομπότ διαθέτει τον χάρτη του περιβάλλοντος με αποτέλεσμα να γνωρίζει με ακρίβεια την θέση τόσο των τοίχων όσο και των στατικών εμποδίων που βρίσκονται στον χώρο σε σχέση με ένα εξωτερικό σύστημα ανφοράς. Διακρίνουμε τις ακόλουθες περιπτώσεις:

- Το ρομπότ χρησιμοποιεί μόνο τον τοπικό αλγόριθμο εντοπισμού θέσης λειτουργώντας σε ένα ιδανικό περιβάλλον χωρίς την ύπαρξη θορύβου μέτρησης με αποτέλεσμα να έχει μία ακριβή εκτίμηση της στάσης του.
- Το ρομπότ χρησιμοποιεί μόνο τον τοπικό αλγόριθμο εντοπισμού θέσης λειτουργώντας όμως σε ένα ρεαλιστικό περιβάλλον όπου υπάρχει θόρυβος μέτρησης με αποτέλεσμα ο τοπικός αλγόριθμος εντοπισμού θέσης να μην είναι αρκετός για τον εντοπισμό της ακριβής στάσης του ρομπότ.
- Το ρομπότ χρησιμοποιεί τον μηχανισμό απόφασης εκφόρτωσης προκειμένου να αποφασίζει σε κάθε χρονική στιγμή εάν είναι απαραίτητη η χρήση του απομακρυσμένου αλγορίθμου εντοπισμού θέσης προκειμένου να διορθώσει την εκτίμηση που έχει για την στάση του. Δοκιμάζονται διαφορετικά σενάρια σχετικά με το ποιες από τις τρεις μετρικές που αναφέρθηκαν χρησιμοποιεί ο αλγόριθμος απόφασης εκφόρτωσης.

Όπως μπορούμε να παρατηρήσουμε, όταν δεν υπάρχει θόρυβος μέτρησης (Σχήμα 4.3.1) ο τοπικός αλγόριθμος εντοπισμού στάσης είναι αρκετός και το ρομπότ μπορεί να φθάσει με επιτυχία στον στόχο του ενώ ακολουθεί με ακρίβεια το βέλτιστο μονοπάτι από την αρχική στην τελική θέση. Σε περίπτωση όμως που υπάρχει θόρυβος (Σχήμα 4.3.2) ο τοπικός αλγόριθμος εκτίμησης στάσης αποτυγχάνει να εντοπίσει την ακριβή στάση του ρομπότ με την αβεβαιότητα του να αυξάνεται με την πάροδο του χρόνου. Έτσι το ρομπότ έχοντας λανθασμένη εκτίμηση για την ακριβή του στάση, συγκρούεται με ένα από τα εμπόδια που βρίσκονται στον χώρο και αποτυγχάνει να φτάσει στην τελική θέση. Στο συγκεκριμένο σημείο είναι σημαντικό να επισημάνουμε πως το estimation set που αναφέρθηκε παραπάνω υπολογίζεται μόνο μέσω του μοντέλου κίνησης και άρα μόνο μέσω της οδομετρίας του ρομπότ. Αντίθετα ο τοπικός αλγόριθμος εντοπισμού θέσης AMCL χρησιμοποιεί και το μοντέλο μέτρησης προκειμένου να διορθώσει την εκτίμηση της στάσης του. Γι αυτόν τον λόγο παρατηρούμε πως το estimation set αποκλίνει από την εκτίμηση της στάσης που προέρχεται από τον τοπικό αλγόριθμο εντοπισμού θέσης.

Στο σχήμα 4.3.3 φαίνεται η χρήση του μηχανισμού απόφασης εκφόρτωσης ο οποίος λαμβάνει υπόψιν του μόνο το κόστος $D_{uncertainty}(X_t)$ που περιγράφθηκε προηγουμένως. Παρατηρούμε ότι όταν η αβεβαιότητα ως προς την ακριβή στάση του ρομπότ γίνει αρκετά μεγάλη και άρα το κόστος $D_{uncertainty}(X_t)$ αυξηθεί αρκετά ξεπερνώντας ένα προκαθορισμένο όριο, τότε γίνεται η χρήση του απομακρυσμένου αλγορίθμου εντοπισμού θέσης ORB-SLAM3 ο οποίος διορθώνει την εκτίμηση της στάσης του ρομπότ. Γνωρίζοντας έτσι πλέον την ακριβή του στάση, το ρομπότ συνεχίζει την κίνηση του χρησιμοποιώντας ξανά τον τοπικό αλγόριθμο εντοπισμού στάσης μέχρις ότου η αβεβαιότητα του ως προς την ακριβή του στάση ξεπεράσει και πάλι το προκαθορισμένο όριο.

Στο σχήμα 4.3.4 φαίνεται η χρήση του μηχανισμού απόφασης εκφόρτωσης ο οποίος λαμβάνει υπόψιν του μόνο το κόστος $D_{obstalce}(x_t)$ που περιγράφθηκε προηγουμένως. Στο συγκεκριμένο πείραμα παρατηρούμε ότι ο μηχανισ-

μός απόφασης εκφόρτωσης επιλέγει να χρησιμοποιήσει τον απομακρυσμένο αλγόριθμο εντοπισμού θέσης ORB-SLAM3 όταν το ρομπότ φθάσει αρκετά κοντά σε ένα εμπόδιο ή σε κάποιον από τους τοίχους που περικλύουν το χώρο. Έτσι όταν υπάρχει μεγάλος κίνδυνος σύγκρουσης, το ρομπότ χρειάζεται μία ακριβή εκτίμηση της στάσης του προκειμένου να μπορέσει να σχεδιάσει σωστά την βέλτιστη διαδρομή του και να αποφύγει την επικείμενη αυτή σύγκρουση. Στην συγκεκριμένη βέβαια περίπτωση αγνοείται εντελώς η αβεβαιότητα ως προς την στάση του ρομπότ.

Στο σχήμα 4.3.5 φαίνεται η χρήση του μηχανισμού απόφασης εκφόρτωσης ο οποίος λαμβάνει υπόψιν τόσο το κόστος $D_{uncertainty}(X_t)$ όσο και το κόστος $D_{obstalce}(x_t)$ που περιγράφθηκαν προηγουμένως. Στην συγκεκριμένη περίπτωση το ρομπότ ακολουθεί μία πιο περίπλοκη διαδρομή καθώς επιθυμεί να φτάσει σε μία διαφορετική τελική θέση σε σχέση με τα προηγούμενα παραδείγματα.

Στο σχήμα 4.3.6 φαίνεται η χρήση του μηχανισμού απόφασης εκφόρτωσης ο οποίος λαμβάνει υπόψιν τόσο το κόστος $D_{uncertainty}(X_t)$ όσο και το κόστος $D_{path}(P_t)$ που περιγράφθηκαν προηγουμένως. Όπως μπορούμε να παρατηρήσουμε σε σημεία όπου η καμπυλότητα της διαδρομής που πρέπει να ακολουθήσει το ρομπότ είναι αρκετά υψηλή ενεργοποιείται ο απομακρυσμένος αλγόριθμος εντοπισμού στάσης για όσο χρονικό διάστημα απαιτείται μέχρι το ρομπότ να περάσει το κομμάτι εκείνο της διαδρομής που παρουσιάζει υψηλή καμπυλότητα.

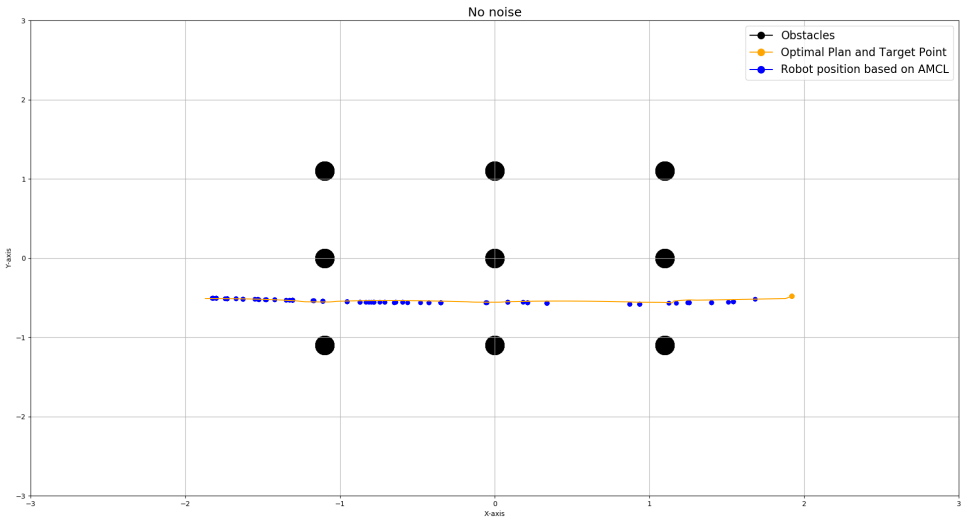

Figure 4.3.1: Χρήση μόνο του τοπικού αλγορίθμου εντοπισμού θέσης χωρίς την ύπαρξη θορύβου

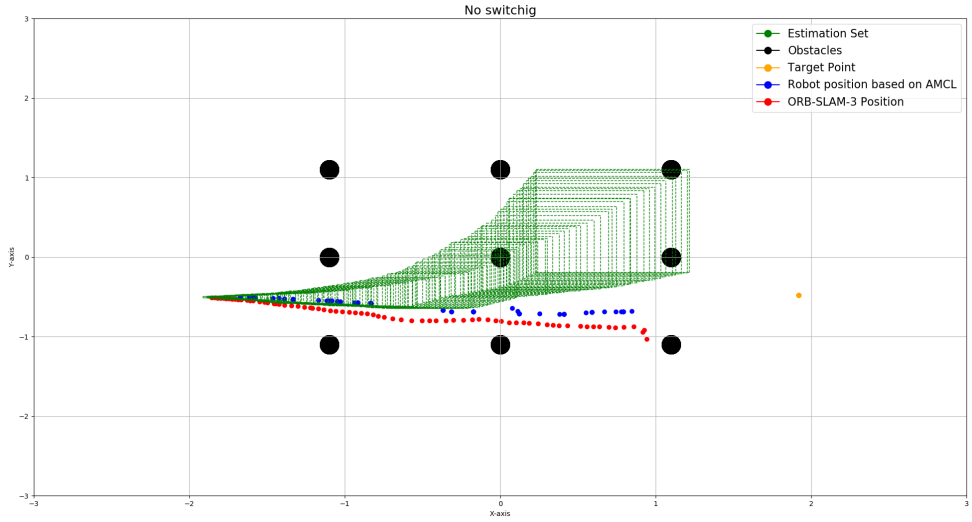

Figure 4.3.2: Χρήση μόνο του τοπικού αλγορίθμου εντοπισμού θέσης με ύπαρξη θορύβου

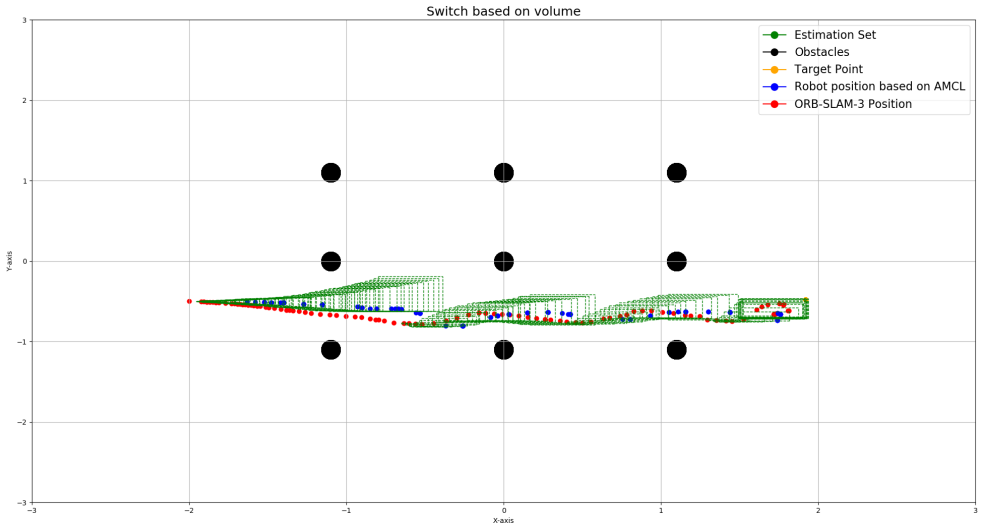

Figure 4.3.3: Χρήση του μηχανισμού απόφασης εκφόρτωσης λαμβάνοντας υπόψιν το κόστος $D_{uncertainty}(X_t)$

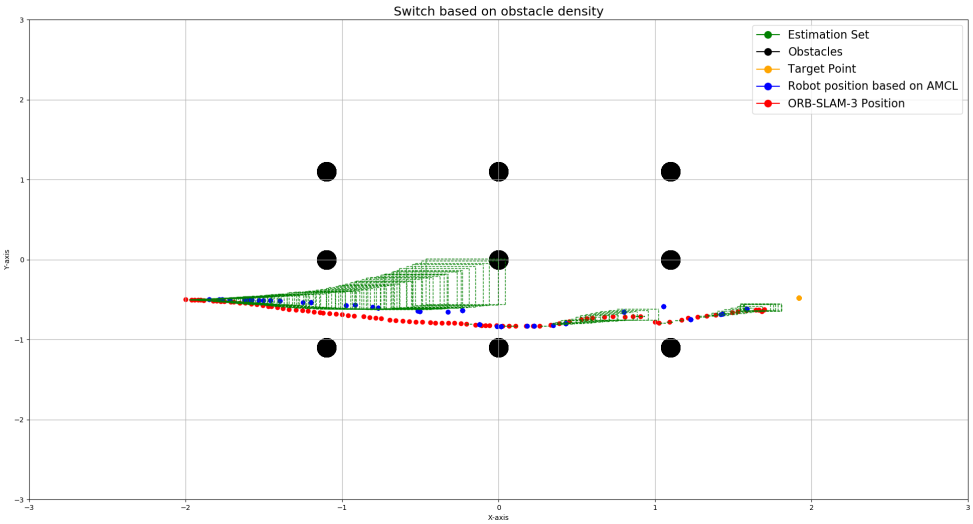

Figure 4.3.4: Χρήση του μηχανισμού απόφασης εκφόρτωσης λαμβάνοντας υπόψιν το κόστος $D_{obstalce}(x_t)$

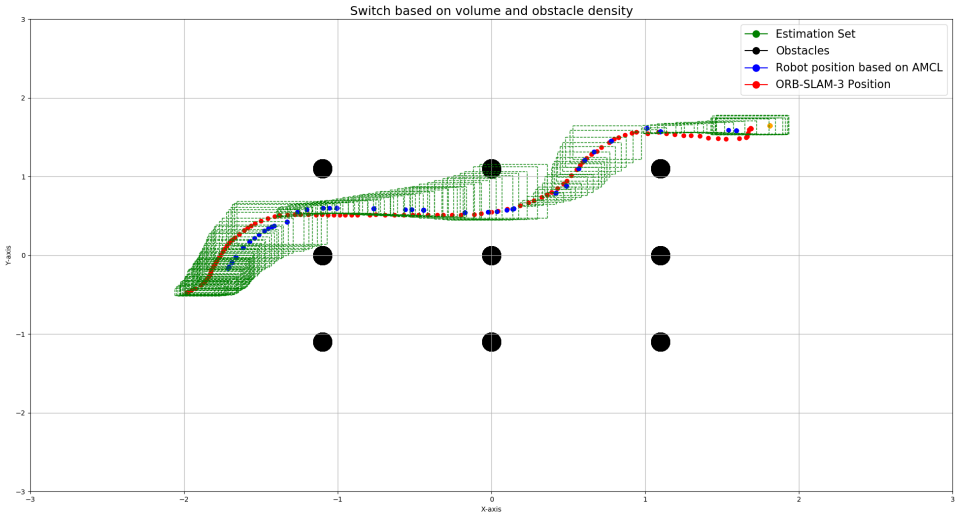

Figure 4.3.5: Χρήση του μηχανισμού απόφασης εκφόρτωσης λαμβάνοντας υπόψιν τα κόστοι $D_{uncertainty}(X_t)$ και $D_{obstalce}(x_t)$

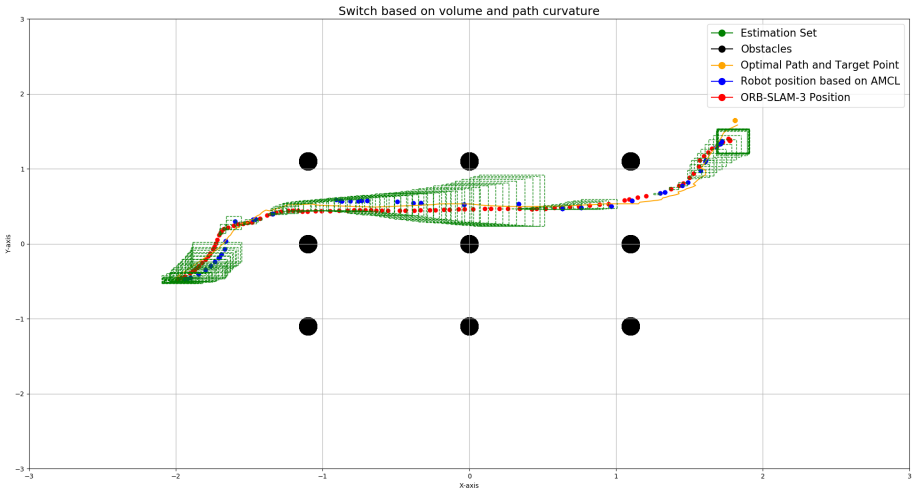

Figure 4.3.6: Χρήση του μηχανισμού απόφασης εκφόρτωσης λαμβάνοντας υπόψιν τα κόστοι $D_{uncertainty}(X_t)$ και $D_{path}(P_t)$

Πείραμα 2

Στο συγκεκριμένο πείραμα θέλαμε να αξιολογήσουμε την επίδραση που έχει το προκαθορισμένο όριο του κόστους $D_{uncertainty}(X_t)$ στην ακρίβεια με την οποία το ρομπότ ακολουθεί την βέλτιστη διαδρομή από την αρχική στην τελική του θέση. Έτσι πραγματοποιήθηκαν τρία πειράματα όπου το ρομπότ μεταβένει από την ίδια αρχική στην ίδια τελική θέση με το προκαθορισμένο όριο για το κόστος $D_{uncertainty}(X_t)$ να είναι διαφορετικό σε κάθε ένα από αυτά και έτσι ο αλγόριθμος απόφασης εκφόρτωσης να λαμβάνει σε διαφορετική στιγμή την απόφαση για την χρήση του απομακρυσμένου αλγορίθμου εντοπισμού θέσης.

Στην εικόνα 4.3.7 έχει επιλεχθεί μικρή τιμή για το προκαθορισμένο όριο του κόστους $D_{uncertainty}(X_t)$. Ως αποτέλεσμα, το ρομπότ χρησιμοποιεί αρκετά συχνά τον απομακρυσμένο αλγόριθμο εντοπισμού θέσης με αποτέλεσμα η απόκλιση του από την βέλτιστη διαδορμή να είναι αρκετά μικρή. Αντίθετα, στην εικόνα 4.3.8 η τιμή του προκαθορισμένου ορίου αυξάνεται με αποτέλεσμα το ρομπότ να αποκλύνει περισσότερο από την βέλτιστη διαδρομή καθώς απαιτείται μεγαλύτερη αβεβαιότητα ως προς την στάση του προκειμένου να ενεργοποιηθεί ο απομακρυσμένος αλγόριθμος εντοπισμού θέσης. Τέλος στην εικόνα 4.3.9 όπου το προκαθορισμένο όριο έχει αρκετά υψηλή τιμή, βλέπουμε μία μεγάλη απόκλιση μεταξύ της κίνησης του ρομπότ και της βέλτιστης διαδρομής από την αρχική στην τελική του θέση.

Μπορούμε λοιπόν να συμπεράνουμε πως όσο μικρότερο είναι το προκαθορισμένο όριο για το κόστος $D_{uncertainty}(X_t)$, τόσο μικρότερη θα είναι η απόκλιση μεταξύ της κίνησης του ρομπότ και της βέλτιστης διαδρομής από την αρχική στην τελική θέση. Βέβαια η επιλογή αρκετά χαμηλής τιμής για το προκαθορισμένο όριο ενδέχεται να οδηγήσει σε μεγάλες καθυστερήσεις καθώς θα ενεργοποιείται αρκετά συχνά ο απομακρυσμένος αλγόριθμος εντοπισμού θέσης. Έτσι η επιλογή του βέλτιστου ορίου εξαρτάται από τις ανάγκες της εκτάστοτε εφαρμογής. Για εφαρμογές που έχουν υψηλές απαιτήσεις σε ακρίβεια παρακολούθησης διαδρομής, απαιτείται μικρή τιμή για το όριο. Αντίθετα για εφαρμογές που έχουν ως πρωταρχικό στόχο να φτάσει το ρομπότ στον στόχο σε όσο μικρότερο χρονικό διάστημα, απαιτείται μεγαλύτερη τιμή.

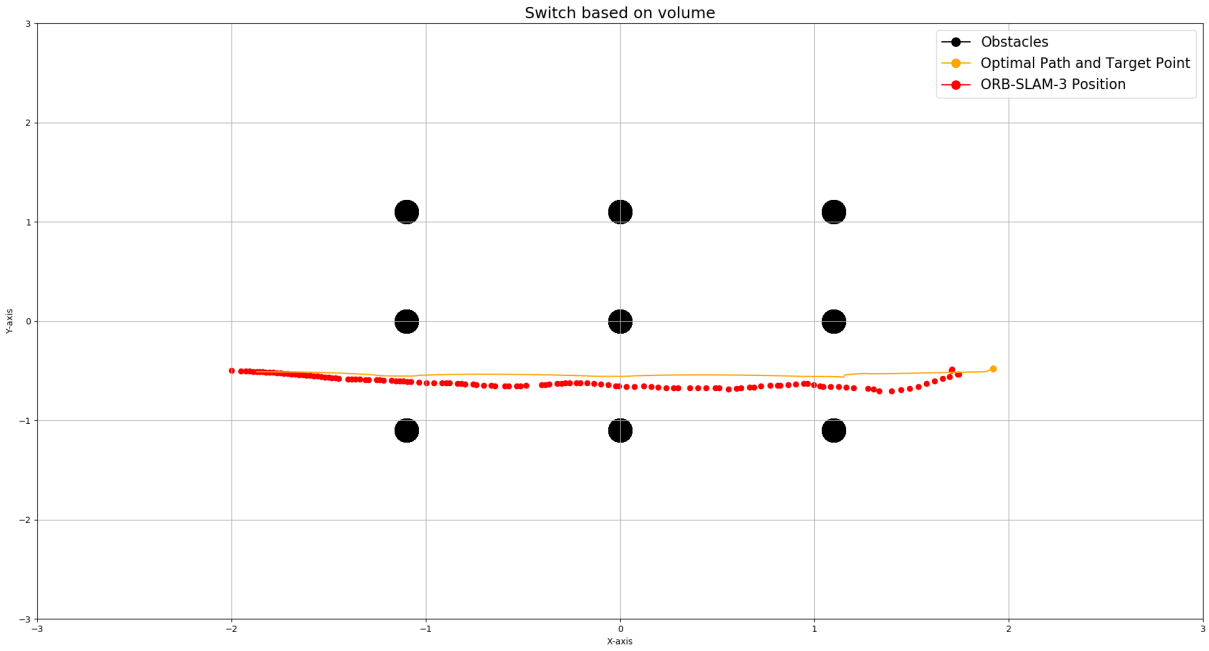

Figure 4.3.7: Χρήση του μηχανισμού απόφασης εκφόρτωσης θέτωντας μία αρκετά μικρή τιμή για το προκαθορισμένο όριο του κόστους $D_{uncertainty}(X_t)$

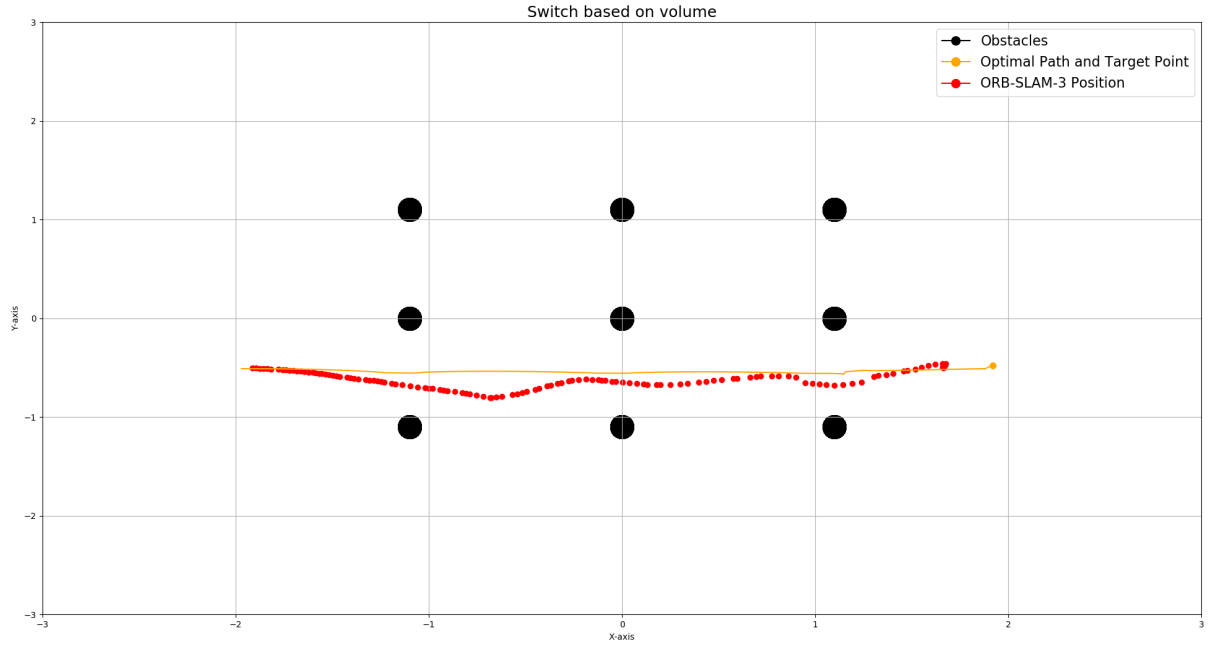

Figure 4.3.8: Χρήση του μηχανισμού απόφασης εκφόρτωσης θέτωντας μία μεσαία τιμή για το προκαθορισμένο όριο του κόστους $D_{uncertainty}(X_t)$

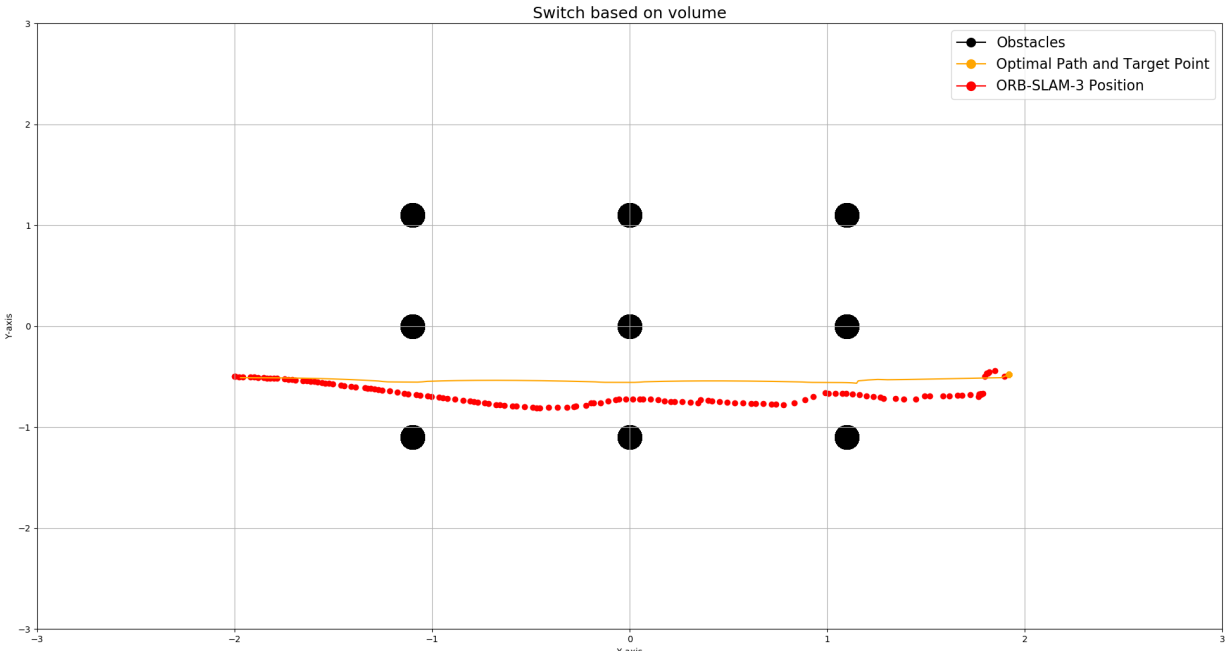

Figure 4.3.9: Χρήση του μηχανισμού απόφασης εκφόρτωσης θέτωντας μία αρκετά μεγάλη τιμή για το προκαθορισμένο όριο του κόστους $D_{uncertainty}(X_t)$

Chapter 5

Συμπεράσματα και Μελλοντικές Προεκτάσεις

5.0.1 Συμπεράσματα

Στο πλαίσιο της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε ο σχεδιασμός και η υλοποίηση ενός διακοπτικού μηχανισμού για την εναλλαγή μεταξύ δύο αλγορίθμων εντοπισμού στάσης ενός κινητού ρομπότ, ενός που εκτελείται τοπικά στο ρομπότ και ενός που απαιτείται να εκτελεστεί απομακρυσμένα σε έναν Edge Server. Ο τοπικός αλγόριθμος εντοπισμού στάσης επιλέχθηκε να είναι ο Adaptive Monte Carlo Localization Algorithm ο οποίος κάνει χρήση των δεδομένων που προέρχονται από την οδομετρία του ρομπότ αλλά και από έναν αισθητήρα μέτρησης απόστασης ο οποίος είναι ενσωματωμένος στο ρομπότ. Αντίστοιχα, ο απομακρυσμένος αλγόριθμος εντοπισμού στάσης επιλέχθηκε να είναι ο ORB-SLAM3 ο οποίος παρέχει μία ακριβή εκτίμηση της στάσης του ρομπότ με βάση μία κάμερα ενσωματωμένη στο ρομπότ. Η αξία του διακοπτικού μηχανισμού ελέγχθηκε σε ένα σενάριο πλοήγησης όπου ένα κινητό ρομπότ επιχειρεί να μεταβεί από μία γνωστή αρχική σε μία γνωστή τελική θέση μέσα σε ένα στατικό περιβάλλον. Προκειμένου να πραγματοποιηθεί το συγκεκριμένο σενάριο, το ρομπότ χρησιμοποιεί τον αλγόριθμο Dijkstra για τον σχεδιασμό της βέλτισης διαδρομής από την αρχική στην τελική του θέση ενώ ο Regulated Pure Pursuit αλγόριθμος χρησιμοποιείτα προκειμένου το ρομπότ να μπορέσει να ακολουθήσει την συγκεκριμένη διαδορμή.

Μέσα από τα πειράματα πο διεξήχθησαν διαπυστώθηκε πως η χρήση του παραπάνω διακοπτικού μηχανισμού επιτυγχάνει μεγαλύτερη ακρίβεια ως προς την εκτίμηση της στάσης του ρομπότ σε σχέση με την χρήση μόνο του τοπικού αλγορίθμου εντοπισμού στάσης, με αποτέλεσμα το ρομπότ να φτάνει επιτυχώς στην θέση-στόχο του χωρίς να συγκρούεται με τα στατικά εμπόδια που υπάρχουν στον χώρο, ενώ παράλληλα αποφεύγεται η καθυστέρηση που θα παρουσιαζόταν σε περίπτωση της χρήσης μόνο του απομακρυσμένου αλγορίθμου εντοπισμού στάσης. Επιπλέον, ο διακοπτικός αυτός μηχανισμός έχει την δυνατότητα, ανάλογα με τους στόχους της εκάστωτε εφαρμογής, να ρυθμιστεί κατάλληλα έτσι ώστε να ισορροπήσει την υπολογιστική ταχύτητα που παρέχεται από τον τοπικό αλγόριθμο εντοπισμού στάσης έναντι της ακρίβειας ως προς τον εντοπισμό της στάσης , και άρα της παρακολούθησης της διαδρομής, που παρέχεται από τον απομακρυσμένο αλγόριθμο εντοπισμού στάσης. Έτσι, σε εφαρμογές όπου η ασφάλεια της κίνησης είναι πρωταρχικής σημασίας και το ρομπότ πρέπει να ακολουθήσει με ακρίβεια μία σχεδιασμέμη διαδρομή, ο διακοπτικός μηχανισμός μπορεί να ρυθμιστεί κατάλληλα ώστε να εξυπηρετεί τον συγκεκριμένο σκοπό. Αντίστοιχα, σε εφαρμογές που έχουν ως πρωταρχικό τους στόχο να φτάσει το ρομπότ στον στόχο του σε όσο μικρότερο χρονικό διάστημα ακόμα και αν δεν ακολουθεί με μεγάλη ακρίβεια την βέλτιστη διαδρομή, ο διακοπτικός μηχανισμός μπορεί να ρυθμιστεί με κατάλληλο τρόπο ώστε να παρέχει την επιθυμητή αυτή συμπεριφορά.

5.0.2 Μελλοντικές Προεκτάσεις

Μία μελλοντική προέκταση που θα είχε ιδιαίτερο ενδιαφέρον θα ήταν η αξιολόγηση του παραπάνω διακοπτικού μηχανισμού σε ένα πιο ρεαλιστικό σενάριο με την χρήση ενός πραγματικού ρομπότ το οποίο επικοινωνεί μέσω του δικτύου με έναν Edge Server. Σε μία τέτοια περίπτωση, ο διακοπτικός μηχανισμός θα μπορούσε να λαμβάνει υπ-

όψιν του και επιπλέον μετρικές που αφορούν το δίκτυο, όπως είναι η καθυστέρηση που δημιουργείτε σε περίπτωση χρήσης του απομακρυσμένου αλγορίθμου εντοπισμού θέσης, αλλά και μετρικές σχετικές με την διαθεσιμότητα των πόρων της απομακρυσμένης υποδομής. Η χρήση τέτοιων μετρικών θεωρούμε πως θα δημιουργούσε ένα πιο ολοκληρωμένο διακωπτικό σύστημα το οποίο θα ήταν πιο ανθεκτικό σε συνθήκες που παρατηρούνται στον πραγματικό κόσμο.

Επιπλέον, μια πιο πολύπλοκη και προσαρμοζόμενη τεχνική η οποία θα συνδιάζει τις μετρικές που αναφέρθηκαν στο Κεφάλαιο 4 αλλά και στην προηγούμενη παράγραφο σε μία ενιαία συνθήκη απόφασης με χρήση συγκεκιρμένων συντελεστών, θα μπορούσε να οδηγησει σε ακόμα καλύτερα αποτελέσματα χωρίς να απαιτείται χειροκίνητα η δοκιμή διαφορετικών προκαθορισμένων ορίων προκειμένου να επιτευχθεί η επιθυμητή συμπεριφορά. Προς αυτήν την κατεύθυνση, θα μπορούσαν να διερευνηθούν τεχνικές βαθιάς μάθησης για τον υπολογισμό και την δυναμική προσαρμογή των συντελεστών της συνθήκης με βάση τις διαφορετικές απαιτήσεις της εκάστοτε εφαρμογής

Ακόμη, η χρήση και προσαρμογή του παραπάνω διακοπτικού μηχανισμού σε σενάρια όπου υπάρχουν δυναμικά εμπόδια στον χώρο αλλά και σε σενάρια όπου υπάρχουν πολλά κινητά ρομπότ τα οποία κινούνται μέσα σε ένα ενιαίο χώρο και μοιράζονται τους ίδιους απομακρυσμένους πόρους είναι κάτι που θα είχε αρκετό ενδιαφέρον να διερευνηθεί.

Τέλος, ο σχεδιασμός και η υλοποίηση ενός ελεγκτή στον οποίο θα έχει ενσωματωθεί ο προτεινόμενος διακοπτικός μηχανισμός και ο οποίος θα παρέχει θεωρητικές εγγυήσεις για την ευστάθεια και την σύγκλιση του δυναμικού συστήματος του ρομπότ θεωρούμε πως θα μπορούσε να οδηγήσει σε μεγαλύτερη ανθεκτικότητα και ευρωστία του συνολικού ρομποτικού συστήματος.

Chapter 6

Βιβλιογραφία

[1] Arumugam, R. et al. "DAvinCi: A Cloud Computing Framework for Service Robots". In: June 2010, pp. 3084–3089. DOI: 10.1109/ROBOT.2010.5509469.

[2] Bhattacharya, A. and De, P. "A Survey of Adaptation Techniques in Computation Offloading". In: *Journal of Network and Computer Applications* 78 (Nov. 2016). DOI: 10.1016/j.jnca.2016.10.023.

[3] Campos, C. et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM". In: *IEEE Transactions on Robotics* 37.6 (Dec. 2021), pp. 1874–1890. ISSN: 1941-0468. DOI: 10.1109/tro.2021.3075644. URL:

[4] Chen, W. et al. "QoS-Aware Robotic Streaming Workflow Allocation in Cloud Robotics Systems". In: *IEEE Transactions on Services Computing* PP (Feb. 2018), pp. 1–1. DOI: 10.1109/TSC.2018.2803826.

[5] Chinchali, S. et al. *Network Offloading Policies for Cloud Robotics: a Learning-based Approach*. 2019. arXiv: 1902.05703 [cs.RO].

[6] Guo, Y. et al. "An Energy Sensitive Computation Offloading Strategy in Cloud Robotic Network Based on GA". In: *IEEE Systems Journal* 13.3 (2019), pp. 3513–3523. DOI: 10.1109/JSYST.2018.2830395.

[7] Guo, Y. et al. "An energy sensitive system framework for cloud robotic network". In: *International Journal of Communication Systems* 32 (July 2019), e4028. DOI: 10.1002/dac.4028.

[8] Hong, Z. et al. "QoS-Aware Cooperative Computation Offloading for Robot Swarms in Cloud Robotics". In: *IEEE Transactions on Vehicular Technology* PP (Feb. 2019), pp. 1–1. DOI: 10.1109/TVT.2019.2901761.

[9] Hu, G., Tay, W. P., and Wen, Y. "Cloud Robotics: Architecture, Challenges and Applications". In: *IEEE Network - NETWORK* 26 (May 2012), pp. 21–28. DOI: 10.1109/MNET.2012.6201212.

[10] Jazayeri, F., Shahidinejad, A., and Ghobaei-Arani, M. "Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach". In: *Journal of Ambient Intelligence and Humanized Computing* 12 (Aug. 2021). DOI: 10.1007/s12652-020-02561-3.

[11] Jung, W.-S. et al. "ACODS: adaptive computation offloading for drone surveillance system". In: *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. 2017, pp. 1–6. DOI: 10.1109/MedHocNet.2017.8001647.

[12] Kattepur, A., Rath, H. K., and Simha, A. "A-Priori Estimation of Computation Times in Fog Networked Robotics". In: June 2017, pp. 9–16. DOI: 10.1109/IEEE.EDGE.2017.11.

[13] L., Q. et al. "Edge Computing for Mobile Robots: Multi-Robot Feature-Based Lidar Odometry with FPGAs". In: Nov. 2019. DOI: 10.23919/ICMU48249.2019.9006646.

[14] Lei, F. et al. "Joint Computation Offloading and URLLC Resource Allocation for Collaborative MEC assisted Cellular-V2X Networks". In: *IEEE Access* PP (Jan. 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.2970750.

[15] Li, S. et al. "Latency-Aware Task Assignment and Scheduling in Collaborative Cloud Robotic Systems". In: July 2018, pp. 65–72. DOI: 10.1109/CLOUD.2018.00016.

[16] Li, W. et al. "Subtask Scheduling for Distributed Robots in Cloud Manufacturing". In: *IEEE Systems Journal* 11 (June 2015), pp. 1–10. DOI: 10.1109/JSYST.2015.2438054.

[17] Mell, P. and Grance, T. *The NIST Definition of Cloud Computing*. en. 2011-09-28 2011. DOI: https://doi.org/10.6028/NIST.SP.800-145.

[18] Mohamed, N., Al-Jaroodi, J., and Jawhar, I. "Utilizing Fog Computing for Multi-robot Systems". In: Jan. 2018, pp. 102–105. DOI: 10.1109/IRC.2018.00023.

[19] Pandey, P., Pompili, D., and Yi, J. "Dynamic Collaboration Between Networked Robots and Clouds in Resource- Constrained Environments". In: *Automation Science and Engineering, IEEE Transactions on* 12 (Apr. 2015), pp. 471–480. DOI: 10.1109/TASE.2015.2406115.

[20] Penmetcha, M. and Min, B.-C. "A Deep Reinforcement Learning-Based Dynamic Computational Offloading Method for Cloud Robotics". In: *IEEE Access* PP (Apr. 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3073902.

[21] Qu, C., Wang, S., and Calyam, P. "DyCOCo: A Dynamic Computation Offloading and Control Framework for Drone Video Analytics". In: Oct. 2019, pp. 1–2. DOI: 10.1109/ICNP.2019.8888089.

[22] Rahman, A. et al. "A Cloud Robotics Framework of Optimal Task Offloading for Smart City Applications". In: Dec. 2016, pp. 1–7. DOI: 10.1109/GLOCOM.2016.7841487.

[23] Rahman, A. et al. "Communication-Aware Cloud Robotic Task Offloading With On-Demand Mobility for Smart Factory Maintenance". In: *IEEE Transactions on Industrial Informatics* PP (Oct. 2018), pp. 1–1. DOI: 10.1109/TII.2018.2874693.

[24] Rahman, A. et al. "Energy-Efficient Optimal Task Offloading in Cloud Networked Multi-Robot Systems". In: *Computer Networks* 160 (May 2019). DOI: 10.1016/j.comnet.2019.05.016.

[25] Sarker, V. et al. "Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing". In: May 2019. DOI: 10.1109/ICASERT.2019.8934466.

[26] Shahidinejad, A. and Ghobaei-Arani, M. "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach". In: *Software Practice and Experience* 50 (Sept. 2020). DOI: 10.1002/spe.2888.

[27] Spatharakis, D. et al. "A Switching Offloading Mechanism for Path Planning and Localization in Robotic Applications". In: Nov. 2020, pp. 77–84. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00031.

[28] Wang, X. and Guo, H. "Mobility-Aware Computation Offloading for Swarm Robotics using Deep Reinforcement Learning". In: *2021 IEEE 18th Annual Consumer Communications and Networking Conference (CCNC)*. IEEE, Jan. 2021. DOI: 10.1109/ccnc49032.2021.9369456. URL: