ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

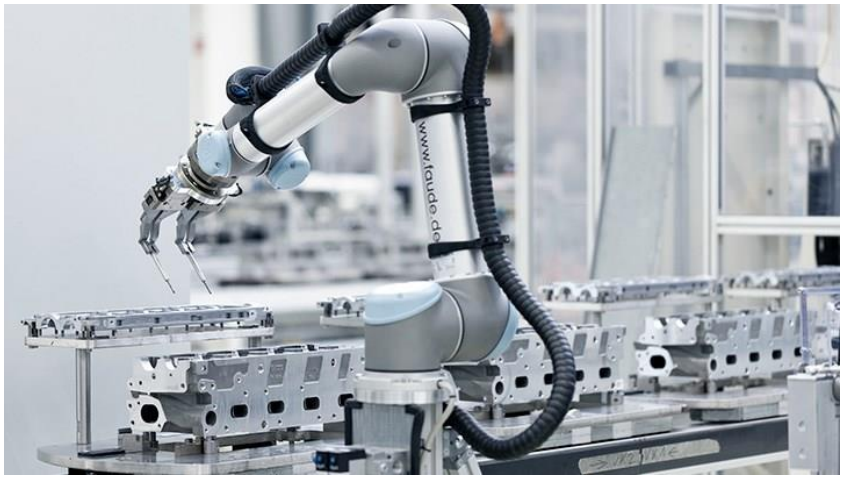
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ

ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΘΕΜΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΜΕΛΕΤΗ ΚΙΝΗΣΗΣ ΡΟΜΠΟΤ ΣΕ ΠΕΡΙΒΑΛΛΟΝ ΜΕ ΕΜΠΟΔΙΑ



ΟΝΟΜΑ ΦΟΙΤΗΤΗ: ΜΑΚΡΗΣ ΙΑΣΩΝ

Α.Μ ΦΟΙΤΗΤΗ: 71447503

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΖΑΧΑΡΙΑ ΠΑΡΑΣΚΕΥΗ

ΑΙΓΑΛΕΩ, ΙΟΥΝΙΟΣ 2023

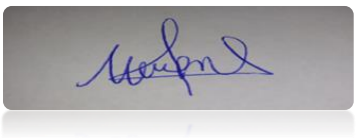
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Μακρής Ιάσων** του **Αντωνίου**, με αριθμό μητρώου 71447503 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής **ΜΗΧΑΝΙΚΩΝ** του Τμήματος **ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ**, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών



Επιτροπή αξιολόγησης διπλωματικής

No	Ονοματεπώνυμο	Ψηφιακή Υπογραφή
1	**Παρασκευή Ζαχαρία**	Paraskevi Zacharia Digitally signed by Paraskevi Zacharia Date: 2023.07.19 13:22:37 +03'00'
2	**Χαμηλοθώρης Γεώργιος**	Georgios Chamilothoris 2023.07.19 16:06:29 +03'00'
3	**Αβραάμ Χατζόπουλος**	AVRAAM CHATZOPOULOS Digitally signed by AVRAAM CHATZOPOULOS Date: 2023.07.19 13:41:27 +03'00'

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Αντικείμενο διπλωματικής εργασίας

Στον κλάδο της ρομποτικής τα τελευταία χρονιά παρατηρούμε μια ραγδαία εξέλιξη στον τομέα του σχεδιασμού και της κατασκευής των ρομποτικών συστημάτων. Τα παραδοσιακά βιομηχανικά ρομπότ εκτελούν πολλών ειδών εφαρμογές και είναι σημαντικά ακόμη και σήμερα σε πολλούς κλάδους όπως η αυτοκινητοβιομηχανία, όμως υστερούν στην συνεργασία με τον άνθρωπο, αφού βρίσκονται απομακρυσμένα από αυτόν σε ασφαλισμένους χώρους για την αποφυγή κάποιου ατυχήματος, εκτός αυτού υστερούν στην αποφυγή εμποδίων αφού τα περισσότερα δεν διαθέτει αισθητήρες κρούσης. Αυτό συμβαίνει διότι εκτελούν μια επαναληπτική εργασία σε καλά καθορισμένους χώρους και η εμφάνιση εμποδίων δεν αποτελεί ένα συχνό φαινόμενο. Η εξέλιξη την ρομποτικής ήρθε με τα συνεργατικά ρομπότ (collaborative robots, cobots), τα οποία είναι ρομπότ σχετικά μικρού μεγέθους και βάρους, οπού βασικός τους σκοπός είναι η συνεργασία τους με τον άνθρωπο. Είναι ευκολά στην τοποθέτηση τους σε διαφορά περιβάλλοντα είτε δομημένα, είτε αδόμητα, καθώς και ευκολά στην εγκατάσταση τους και τον προγραμματισμό αφού ένα cobot προγραμματίζεται κυρίως μέσω της μεθοδολογίας της διδασκαλίας από τον άνθρωπο. Διαθέτουν αισθητήρες κρούσης για την αποφυγή εμποδίων και μπορούν να επικοινωνήσουν με αλλά cobots μέσα στο περιβάλλον τους.

Άλλη μια εξέλιξη στον τομέα της ρομποτικής παρουσιάστηκε με την βελτιστοποίηση στον τομέα του σχεδιασμού κίνησης και ιδιαίτερα στις στρατηγικές αποφυγής εμποδίων. Ένα υποσύνολο του σχεδιασμού κίνησης αποτελεί ο σχεδιασμός διαδρομής οπού ο χρήστης θέτοντας την αρχική και την τελική θέση του ρομπότ και χρησιμοποιώντας έναν αλγόριθμο σχεδιασμού διαδρομής για την εύρεση του πλησιέστερου μονοπατιού και την αποφυγή εμποδίων. Οι αλγόριθμοι σχεδιασμού διαδρομής έχουν εξελιχθεί και η επιλογή τους γίνεται βάση πολλών παραγόντων , ένας από αυτούς τους λόγους είναι η δομή του περιβάλλοντος δηλαδή αν ένα περιβάλλον είναι στατικό η δυναμικό.

Όσον αφορά τον σχεδιασμό διαδρομής για ρομποτικά συστήματα, οι αλγόριθμοι RRT (Rapidly-exploring Random Trees) είναι μια δημοφιλής επιλογή λόγω της ικανότητάς τους να χειρίζονται χώρους υψηλών διαστάσεων με πολύπλοκα εμπόδια. Προκειμένου να βελτιστοποιηθεί η απόδοση και να ξεπεραστούν οι περιορισμοί του αρχικού αλγορίθμου RRT, έχουν αναπτυχθεί διάφορες παραλλαγές και βελτιώσεις με την πάροδο των ετών όπως οι αλγόριθμοι rrt-star και rrt-connect.

1.2 Σκοπός διπλωματικής εργασίας

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός κίνησης ενός ρομποτικού βραχίονα σε ένα περιβάλλον με στατικά εμπόδια με τη χρήση αλγορίθμων σχεδιασμού διαδρομής(path planning algorithms) με σκοπό να αποφευχθεί η σύγκρουση και να βρεθεί το πλησιέστερο μονοπάτι προς τον στόχο. Συγκεκριμένα, στόχος είναι η ανάλυση των αλγορίθμων σχεδιασμού διαδρομής RRT, RRT-STAR, RRT-

CONNECT και η αξιολόγηση τους βάση του μήκους της διαδρομής, του χρόνου εκτέλεσης της τροχιάς και του χρόνου σχεδιασμού της τροχιάς, από μια αρχική έως μια τελική θέση βάση 2 σεναρίων: εκτέλεση τροχιάς σε χώρο δίχως εμπόδιο, εκτέλεση τροχιάς σε χώρο με εμπόδιο.

1.3 Μεθοδολογία

Η συγγραφή της παρούσας διπλωματικής χωρίζεται στο θεωρητικό τμήμα και το τμήμα της εφαρμογής. Το θεωρητικό μέρος χωρίζεται και εκείνο σε δυο μέρη. Στο πρώτο μέρος παρουσιάζεται το υπόβαθρο αναλύοντας τα ρομποτικά συστήματα και ιδιαίτερα τα βιομηχανικά ρομπότ και cobots. Στο δεύτερο μέρος γίνεται περιγραφή των τροπών αποφυγής εμποδίων από ένα ρομποτικό σύστημα μελετώντας κυρίως τον τομέα του σχεδιασμού κίνησης. Τέλος, αναλύονται οι τρόποι σχεδιασμού κίνησης και ιδιαίτερα οι αλγόριθμοι σχεδιασμού διαδρομής.

Στο τμήμα της εφαρμογής , γίνεται η ρύθμιση του ρομποτικού βραχίονα μέσω του λογισμικού MoveIt. Για την οπτικοποίηση του ρομποτικού βραχίονα χρησιμοποιήθηκε το λογισμικό gazebo. Μέσω προγραμματισμού κατασκευάστηκαν οι κώδικες σε γλώσσα python για την τοποθέτηση αντικειμένων στο περιβάλλον του gazebo και RViz, για τον σχεδιασμό κίνησης του ρομποτικού βραχίονα καθώς και για την αξιολόγηση των αλγορίθμων σχεδιασμού διαδρομής. Στην συνέχεια κατασκευάζονται 2 σενάρια προσομοίωσης με σκοπό την ανάλυση την σύγκριση και την συμπεριφορά των αλγορίθμων. Στο πρώτο σενάριο γίνεται η εκτέλεση τροχιάς χωρίς εμπόδιο ενώ στο δεύτερο παρεμβάλλεται ένα εμπόδιο. Τέλος, γίνεται ανάλυση και η σύγκριση των αλγορίθμων σχεδιασμού διαδρομής που επιλέχθηκαν βάση των δυο σεναρίων προσομοίωσης. Οι αλγόριθμοι που επιλέχθηκαν είναι ο βασικός αλγόριθμος RRT και οι παραλλαγές του RRT-STAR και RRT-CONNECT.

1.4 Δομή διπλωματικής εργασίας

Κεφάλαιο 2: Γίνεται η ιστορική αναδρομή στις έννοιες ρομποτ, ρομποτική και των βιομηχανικών ρομποτ. Στην συνέχεια κατηγοριοποιούνται τα βιομηχανικά ρομπότ βάση των χαρακτηριστικών τους και περιγράφονται οι εφαρμογές τους στην βιομηχανία .Αναφέρονται οι διάφορες τους με τα ρομποτ τύπου cobot (collaborative robot) βάση της συμπεριφοράς τους. Τέλος αναλύεται το βασικό υπόβαθρο γύρω από τον κλάδο της ρομποτικής όπως είναι ο κλάδος της κινηματικής και του χώρου διαμόρφωσης.

Κεφάλαιο 3: Περιγράφονται οι τρόποι αποφυγής εμποδίου και ιδιαίτερα του σχεδιασμού κίνησης(Motion planning) κατηγοριοποιώντας τον σε σχεδιασμό τροχιάς και σχεδιασμό διαδρομής. Τέλος, αναλύονται οι αλγόριθμοι σχεδιασμού διαδρομής δίνοντας με έμφαση στον αλγόριθμο rapidly random trees.

Κεφάλαιο 4: Γίνεται η εισαγωγή του “λογισμικού” Ros (robotic operating system) καθώς και παρουσιάζεται η ιστορική αναδρομή του τα χαρακτηριστικά της δομής του και των εργαλείων του.

Κεφάλαιο 5: Παρουσιάζεται η κατασκευή της εφαρμογής εκτελεσης τροχιάς του ρομποτικού βραχίονα βάση των αλγορίθμων RRT, η επεξήγηση των σεναρίων και των προγραμμάτων καθώς και η ανάλυση και σύγκριση των αλγορίθμων.

Κεφάλαιο 6: Παρουσιάζονται τα συμπεράσματα και τα αποτελέσματα της εφαρμογής μέσω της προσομοίωσης και του θεωρητικού μέρους της εργασίας.

2 θεωρητικό Υπόβαθρο

2.1 Ιστορική Αναδρομή

Ρομποτ - Ρομποτική

Η λέξη ρομπότ εισήχθη στο αγγλικό λεξιλόγιο από τον Τσέχο συγγραφέα Karel Čapek το 1920 στο έργο με τίτλο R.U.R(*Rossum's Universal Robots)*. Η λέξη robota στα τσέχικα προέρχεται από την σλαβική γλωσσική του ρίζα, «rab» που σημαίνει «σκλάβος». Η πλοκή του έργου διαδραματίζεται σε ένα εργοστάσιο που κατασκευάζονται τεχνητοί άνθρωποι με την ονομασία ρομπότ. Η προσέγγιση του συγγραφέα όμως ήταν τελείως διαφορετική από την σημερινή πραγματικότητα αφού τα ρομπότ ήταν κατασκευασμένα με πρότυπο την ανθρώπινη μορφή και φυσιολογία και απαρτίζονταν από τεχνητή σάρκα και αίμα. Επιπλέον τα ρομπότ είχαν την δυνατότητα να σκέπτονται και να λαμβάνουν πρωτοβουλίες, γεγονός που αντιτίθεται στον σχεδιασμό των σημερινών ρομποτ. Το 1923 το έργο μεταφράστηκε σε 30 γλώσσες κάνοντας το πασίγνωστο σε Ευρώπη και βορειά Αμερική. [1]

Το 1941 ο συγγραφέας διηγημάτων επιστημονικής φαντασίας Isaac Asimov εισήγαγε τον όρο ρομποτική στο βιβλίο του με τίτλο “Liar”, ενώ το 1942 εισήγαγε τους τρεις ηθικούς νομούς στην επιστήμη της ρομποτικής στην συνέχεια οι τρεις νόμοι λόγω της επίδρασης που επιτέλεσαν στον κλάδο της ρομποτικής συμπεριλήφθηκαν στο βιβλίο Handbook of Robotics, 56^{η} έκδοση 2058 AD. Στην συνέχεια ο Asimov στο βιβλίο του ¨Robots and Empire” πρόσθεσέ έναν ακόμα νομό τον μηδενικό οπού προηγείται από τους τρεις κλασικούς νομούς.

Αναλυτικά οι τρεις νόμοι είναι:

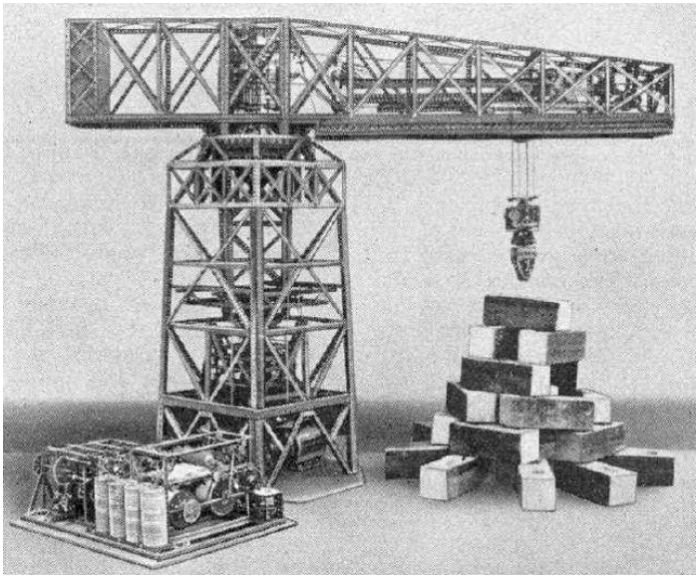
- Πρώτος Νόμος: ” Ένα ρομπότ δεν μπορεί να τραυματίσει έναν άνθρωπο ή, μέσω της αδράνειας, να επιτρέψει σε έναν άνθρωπο να βλάψει.”
- Δεύτερος Νόμος: “ Ένα ρομπότ πρέπει να υπακούει στις εντολές που του δίνονται από τα ανθρώπινα όντα, εκτός εάν τέτοιες εντολές θα έρχονταν σε αντίθεση με τον Πρώτο Νόμο.”

- Τρίτος Νόμος: "Ένα ρομπότ πρέπει να προστατεύει την ύπαρξή του, εφόσον αυτή η προστασία δεν έρχεται σε αντίθεση με τον Πρώτο ή τον Δεύτερο Νόμο."
- Μηδέν Νόμος: "Ένα ρομπότ δεν μπορεί να βλάψει την ανθρωπότητα είτε με την αδράνεια να επιτρέψει στην ανθρωπότητα να βλάψει".

Οι τρεις νόμοι και ο μηδενικός επηρέασαν την σκέψη των διηγημάτων της επιστημονικής φαντασίας καθώς και την σκέψη στον κλάδο της τεχνητής νοημοσύνης. Ενώ οι νόμοι του Asimov έδεναν καταπληκτικά στα μυθιστορήματα επιστημονικής φαντασίας στον τομέα της σύγχρονης ρομποτικής προκύπτουν πολλά αναπάντητα ηθικά ερωτήματα όπως για παράδειγμα: με ποιον τρόπο ένα ρομπότ μπορεί να κατανοήσει την σημασία της βλάβης για τους ανθρώπους. Επιπλέον υπάρχουν περιπτώσεις και συνθήκες που ένα ρομπότ είναι αναγκαίο να βλάψει έναν άνθρωπο για την αποφυγή κάποιας μεγαλύτερης βλάβης. [2]

Βιομηχανικά ρομπότ

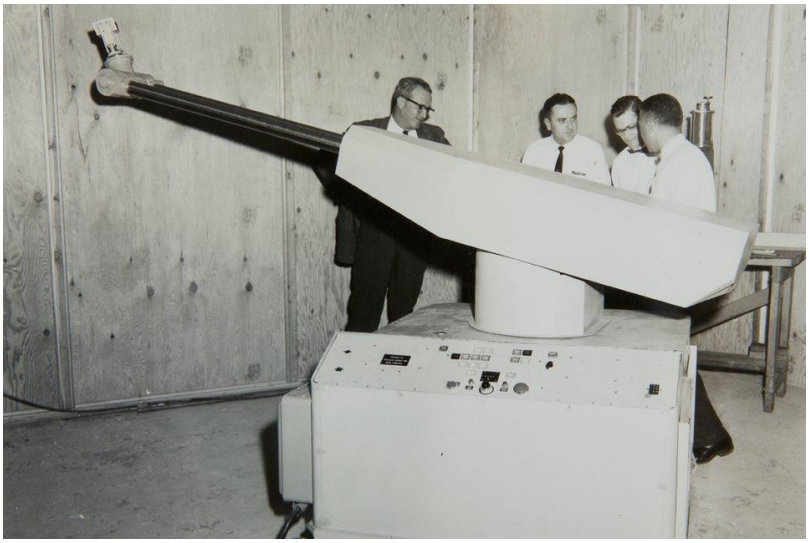
Το πρώτο βιομηχανικό ρομπότ κατασκευάστηκε και ολοκληρώθηκε από τον 21^{ος} ετών φοιτητή της μηχανικής "Bill" Griffith P.Taylor το 1937. Πρόκειται για έναν γερανό κατασκευασμένο με υλικά της εταιρίας Meccano με 5 άξονες κίνησης που τροφοδοτούταν από έναν ηλεκτροκινητήρα. Για τον αυτοματισμό του χρησιμοποιήθηκε διατρητή χαρτοταινία ως μέσο αποθήκευσης δεδομένων με την ενεργοποίηση των ηλεκτρομαγνητικών βαλβίδων. Η βασική λειτουργεία του ρομπότ ήταν η στοίβαξη ξύλινων μπλοκ. [3]



Εικόνα 1 Δημοσίευση του ρομπότ GARGANTUA το 1938 στο περιοδικο MECCANO [3]

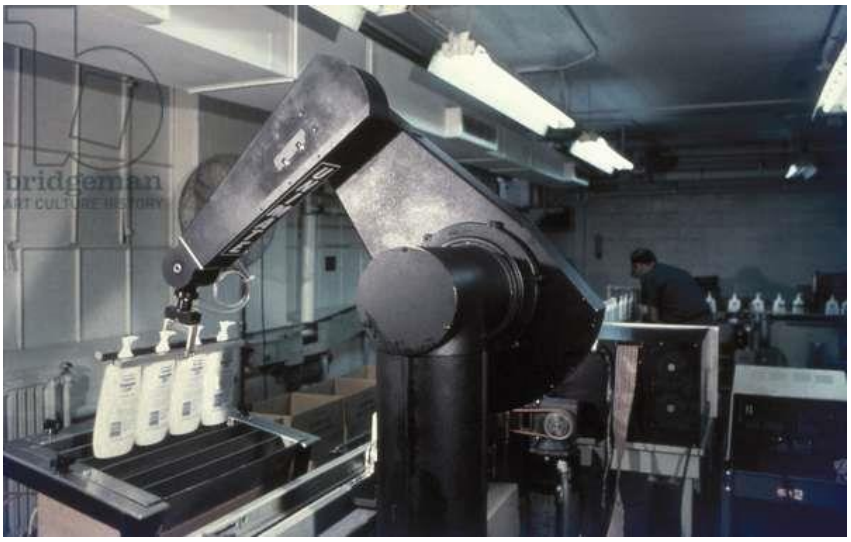
Το 1954 πραγματοποιήθηκε η πρώτη αίτηση για δίπλωμα ευρεσιτεχνίας στο χώρο της ρομποτικής από τον George Devol. Πρόκειται για ένα ρομπότ βιομηχανικού τύπου οπού ονομάστηκε Unimate. Ο George Devol με τον Engelberger ιδρύσαν ένα νέο τμήμα κάτω από την ομπρελά της εταιρίας Condec με όνομα Unimation incorporated.Τα πρώτα πρωτότυπα ελεγχόντουσαν μέσω σωλήνων κενού που λειτουργούσαν ως ψηφιακοί διακόπτες ενώ στην συνέχεια την θέση τους πήραν τα τρανζίστορ. Το 1960 πουλήθηκε το

πρώτο Unimate ρομπότ στην εταιρία General Motor που χρησιμοποιήθηκε για να χειρίζεται εξαρτήματα κάτω από επεξεργασία χύτευσης υπό πίεση και για σημειακή συγκόλληση και το 1961 για να μετακινεί και να στοιβάζει κομμάτια μετάλλου υπό υψηλή θερμοκρασία. [4] Το Unimate ρομποτ αποτέλεσε ένα ογκώδες ρομποτ με βάρος 1575 κιλά, ύψος 1,42m και μήκος 4 μετρά. Για την κίνηση του χρησιμοποιούσε υδραυλικούς ενεργοποιητές και ως αισθητήρα έναν περιστροφικό κωδικοποιητή για τον υπολογισμό της περιστροφής των αρθρώσεων. Ήταν ένα ρομποτ 6 βαθμών ελευθέριας (6Dof) με 3 για την περιστροφή των αρθρώσεων και τρεις για την περιστροφή του τελικού σημείου δράσεως. [5]



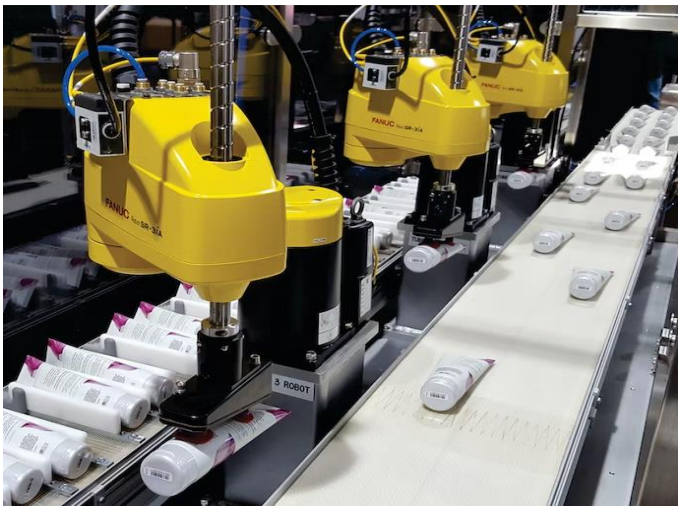
Εικόνα 2 Το ρομπότ Unimate Ο George Devol ο Engelberger και συνάδελφοι. [6]

Το 1969 ο Scheinman ένας φοιτητής του Στάνφορντ σχεδίασε και κατασκεύασε έναν βραχίονα 6 βαθμών ελευθέριας με 6 κινητήρες συνεχούς ρεύματος (dc) που ελεγχόντανε από έναν pdp-6 μικροεπεξεργαστή. Η αντιστροφή κινηματική του ρομποτικού βραχίονα (Η αντιστροφή κινηματική λαμβάνει ως είσοδο την τιμή της επιθυμητή θέση του τελικού σημείου δράσης και βρίσκει τις γωνίες των αρθρώσεων) πραγματοποιήθηκε με χρήση κλειστής μορφής και συνεπώς η εκτέλεση της τροχιάς ήταν ταχύτερες σε σχέση με τις παλαιότερες μεθόδους. Πάνω του διέθετε αισθητήρες που υπολογίζανε την απόσταση και την ταχύτητα των τροχιών του ρομπότ. Το 1973 δημιούργησε ένα μικρό και ελαφρύ ηλεκτρικό ρομπότ το vicarm, που εκτελούσε εργασίες συναρμολόγησης ανταλλακτικών κατά πλείστων. Στην συνέχεια το ρομπότ vicarm αγοράστηκε από την εταιρία Unimation και μετατράπηκε στο γνωστό Puma ρομπότ ένα ρομπότ το οποίο θεωρούταν υπόδειγμα ρομποτικού βραχίονα. [7]



Εικόνα 3 Το ρομπότ puma συσκευάζει μπουκάλια σε ένα εργοστάσιο το 1980 [8]

Το 1978 ο Hiroshi Makino ένας καθηγητής του πανεπιστημίου yamanashi της Ιαπωνίας σχεδίασε έναν ρομποτικό βραχίονα που διέφερε από τους προηγουμένους δεδομένου ότι μπορούσε να εκτελέσει την ίδια εργασία με λιγότερη κίνηση αλλά περισσότερη ταχύτητά και ακρίβεια. Τα ρομπότ χρησιμοποιούνταν για εργασίες συναρμολόγησης και pick and place. Το ρομπότ scara είναι ένα ρομποτ 4 αξόνων που είναι ευέλικτο ως προς τους χ,γ άξονες, αλλά πιο άκαμπτο ως προς τον Z άξονα. Στο ρομπότ scara χρησιμοποιούνται οπτικοί αισθητήρες (πχ κάμερα) για δοκιμές και επιθεώρηση του αντικειμένου. [9]



Εικόνα 4 Scara ρομπότ συσκευάζουν σωληνάρια σε μια γραμμή παραγωγής [10]

2.2 Βιομηχανικά Ρομπότ

Τα βιομηχανικά ρομπότ είναι αυτόματες και προγραμματιζόμενες μηχανές που έχουν ως σκοπό να εκτελέσουν μια συγκεκριμένη εργασία με τρόπο γρήγορο, ακριβή και αποτελεσματικό. Ένα βιομηχανικό ρομπότ αποτελείται από αρθρώσεις που συνδέονται μεταξύ τους μέσω των συνδέσμων τους. Οι αρθρώσεις αυτές ανάλογα με τον είδος τους (περιστροφική, πρισματική κτλ.) καθώς και τον συνδυασμό τους κατηγοριοποιούν τον τύπο ενός βιομηχανικού ρομπότ. Για παράδειγμα ένα ρομπότ τύπου Scara έχει το χαρακτηριστικό ότι οι διαθέτει 2 στροφικές αρθρώσεις και μια πρισματική (*βλέπε εικόνα 4*). Το τελικό σημείο δράσης του ρομπότ αποτελεί ένα από τα σημαντικότερα μέρη του, διότι μέσω αυτού μπορεί να αλληλοεπιδρά με το περιβάλλον στο οποίο εργάζεται και να εκτελεί διάφορων ειδών εργασίες και ανάλογα με το φύση της εργασίας προσάπτονται και διαφορετικού είδους εργαλεία. Τέλος για την διασφάλιση της ακρίβεια και της επαναληψιμότητα μιας εργασίας χρησιμοποιούνται μέθοδοι ελέγχου κίνησης..

Μέθοδοι ελέγχου κίνησης

Τα ρομπότ ανάλογα με τον έλεγχο κίνησης τους ταξινομούνται σε ρομπότ σημείου προς σημείου (point to point) και ρομπότ συνεχούς τροχιάς (Continuous path).

- Point to point: Προγραμματισμός ρομπότ για κίνηση από ένα σημείο προς ένα άλλο χωρίς τον προσδιορισμό της τροχιάς που ακολουθεί. Η τροχιά του ρομπότ και η ταχύτητα του κατά την μετακίνηση δεν παίζει κανέναν ρόλο και αποτελεί λειτουργία του ελεγκτή.

- Continuous path: Μέσω της συγκεκριμένης μεθόδου ελέγχου κίνησης το ρομπότ επαναλαμβάνει μια κίνηση μέσα από ένα σύνολο σημείων τα οποία έχουν προγραμματισθεί με την μέθοδο της διδασκαλίας. Τα σημεία αυτά καταγράφονται από την μονάδα ελέγχου όταν το ρομπότ δημιουργήσει μια επιθυμητή τροχιά. Στην συνέχεια μέσω αλγορίθμων η επιθυμητή αυτή κίνηση γίνεται κατ' επανάληψη [11]

2.2.1 Τύποι Βιομηχανικών Ρομπότ

Οι τύποι των βιομηχανικών ρομπότ είναι οι εξής:

- Αρθρωτό
- Scara
- Καρτεσιανό και gantry
- Παράλληλο (η δέλτα)
- Κυλινδρικό [12]

Καρτεσιανό και gantry

Τα καρτεσιανό ρομπότ αποτελείται από τρεις πρισματικές αρθρώσεις και οι σύνδεσμοι του κινούνται γραμμικά με κύλιση στους 3 κάθετους άξονες του (x,y,z). Ονομάζεται καρτεσιανό διότι οι κινήσεις του

συμπίπτουν με το καρτεσιανό σύστημα συντεταγμένων. Το ρομπότ τύπου gantry είναι παρόμοιο με τα καρτεσιανό με την διαφορά ότι στηρίζεται σε δυο άξονες βάσης Χ αντί για έναν με αποτέλεσμα να ανελκύουν μεγαλύτερο φορτίο από τα καρτεσιανό τύπου ρομποτ. Οι εφαρμογές ποικίλλουν αρκετά, αν και συνήθως χρησιμοποιούνται κυρίως σε εφαρμογές Pick and place, χύτευσης και συναρμολόγησης. Μπορούν να μεταφέρουν βαριά αντικείμενα, έχουν ακρίβεια στην θέση αλλά η κίνηση τους είναι περιορισμένη. [12]

Εικόνα 5 gantry Ρομπότ μεταφέρει παλέτες και κούτες με φαρμακευτικά προϊόντα [13]

Κυλινδρικό

Τα κυλινδρικό ρομπότ έχει τουλάχιστον μια περιστροφική άρθρωση που βρίσκεται στην βάση και τουλάχιστον μια πρισματική. Εργάζονται κυρίως σε στενούς χώρους εργασίας χωρίς απώλεια ταχύτητας και επαναληψημότητα. Εκτελεί κυρίως απλές εργασίες όπως Pick and place ,χύτευση και συναρμολόγηση. Μπορεί να μεταφέρει μεγάλα φορτία αλλά παρουσιάζει μειονεκτήματα στην ακρίβεια της περιστροφικής κίνησης καθώς και στην αποφυγή εμποδίων λόγω ότι δεν μπορεί να ελιχθεί ευέλικτα. [14]

Εικόνα 6 R19E bench top 4-axis κυλινδρικού τύπου ρομπότ [15]

Delta ή Παράλληλο

Το ρομπότ τύπου δέλτα η αλλιώς παράλληλο είναι γνωστό για την εμφάνιση τους καθώς μοιάζει με αράχνη.Αποτελείται από τρεις παράλληλους συνδέσμους με μια ενιαία βάση, οι βραχίονες διαθέτουν ταυτόχρονα είτε πρισματικές είτε περιστροφικές αρθρώσεις. Τοποθετείτε στην κορυφή του χώρου εργασίας με αποτέλεσμα να εκμεταλλεύεται το μεγαλύτερο μέρος του χώρου. Τα πλεονεκτήματα του είναι υψηλή ταχύτητα, επιτάχυνση αλλά και ακρίβεια. Τα μειονεκτήματα του είναι ότι δεν έχει την ικανότητα να μεταφέρει βαριά φορτία. Χρησιμοποιείτε κυρίως στην βιομηχανία τροφίμων και παρασκευής φαρμάκων. [12]

Εικόνα 7 delta ρομπότ τοποθετεί το προϊόν στα κουτιά και απορρίπτει τα λανθασμένα [16]

SCARA

Το ρομποτ scara σχεδιάστηκε με σκοπό να εκτελεί εργασίες συναρμολόγησης γεγονός που προκύπτει και από την ονομασία του Selective compliance Assembly robot arm. Είναι ένα ρομπότ 4 αξόνων με κίνηση στα επίπεδα X-Y και Z και περιστροφική κίνηση 360 μοιρών γύρω από τον άξονα Z. Η πρώτη και η δεύτερη άρθρωση είναι παράλληλες, περιστροφικού τύπου και δίνουν την δυνατότητα στο ρομπότ να κινείται στον τρισδιάστατο χώρο, η τρίτη είναι πρισματική και κινεί το τελικό σημείο δράσης κατακόρυφα και η τέταρτη είναι περιστροφική και εκτελεί εργασίες όπως περιστροφή αντικειμένου, βίδωμα. Είναι από τα πιο ταχύτητά ρομπότ βιομηχανικού τύπου με εξαιρετική επαναληψιμότητα και ακρίβεια και χρησιμοποιείτε κυρίως στον τομέα της Βιοϊατρικής και της αυτοκινητοβιομηχανίας. [12] [14]

Εικόνα 8 Scara robot Thl1200 [17]

Αρθρωτά

Πρόκειται για την πιο κοινή διαμόρφωση βιομηχανικού ρομπότ και είναι όμοιος με το ανθρώπινο βραχίονα. Συνήθως αποτελείται από 6 αρθρώσεις περιστροφικού τύπου αλλά υπάρχουν αρθρωτά ρομπότ με 7 αρθρώσεις έτσι ώστε το τελικό σημείο δράσης του να έχει πρόσβαση σε δύσβατα σημεία. Λόγω των πολλών αρθρώσεων και του βάρους που μεταφέρεται από το τελικό σημείο δράσης στις κατώτερες αρθρώσεις, τα αρθρωτά ρομπότ υστερούν στην ακρίβεια την επαναληψιμότητα καθώς και στην ταχύτητα. Τα θετικά των αρθρωτών ρομπότ είναι ότι το τελικό σημείο δράσης μπορεί να φτάσει σε μεγάλη απόσταση και να σηκώσει μεγάλα φορτία. Τα μεγέθη των αρθρωτών ρομπότ κυμαίνονται από 0,5 έως πάνω από 3,5 μέτρα και έχει την ικανότητα να μεταφέρει από 3 έως πάνω από 1000 κιλά. Εκτός από τις βασικές εφαρμογές όπως pick and place εκτελεί και εφαρμογές ηλεκτροσυγκόλλησης τόξου, συναρμολόγησης αυτοκίνητων, κατασκευή μεταλλικών γεφυρών και άλλων. [12] [14]

Εικόνα 9 articulated robot [18]

2.2.2 Εφαρμογές Βιομηχανικών Ρομπότ

- Χειρισμός (η διακίνηση) Υλικών
- Συγκόλληση
- Βάψιμο
- Συναρμολόγηση
- Μηχανουργική κατεργασία

Χειρισμός η διακίνηση Υλικών

Στην πρώιμη βιομηχανία το 95% του χρόνου αφιερωνόταν αποκλειστικά στην μεταφορά του επεξεργασμένου υλικού έναντι του 5% που χρειαζόταν για να παρασκευαστεί. Τα τότε αυτόματα συστήματα για την μεταφορά υλικού ήταν άκαμπτα και δεν ήταν κατάλληλα για τεράστιες παρτίδες παραγωγής. Σήμερα οι κυρίες εφαρμογές των ρομπότ αποτελούν την τροφοδοσία εργαλειομηχανών με πρώτη ύλη. Ο έλεγχος από σημείο σε σημείο είναι απαραίτητος για εφαρμογές χειρισμού - διακίνησης υλικού, καθώς επιτρέπει στον βραχίονα ρομπότ να επιλέγει ή να τοποθετεί εξαρτήματα σε ίσα βήματα κατά μήκος των αξόνων X, Y και Z ορίζοντας απλώς το μήκος, την κατεύθυνση και τον αριθμό των βημάτων.

Συγκόλληση

Η συγκόλληση είναι η διαδικασία κατά την οποία δυο κομμάτια μετάλλου ενώνονται συνήθως μέσω θέρμανσης και συγκόλλησης. Υπάρχουν δυο τύποι διεργασίας συγκόλλησης από τα ρομπότ. Η σημειακή συγκόλληση και ηλεκτροσυγκόλληση τόξου. Κατά την σημειακή συγκόλληση δυο κομμάτια μετάλλου ενώνονται αφού δημιουργηθεί η απαραίτητη θερμότητα μέσω της διέλευσης ηλεκτρικού ρεύματος. Η συγκόλληση αυτή γίνεται σημειακά. Κατά την ηλεκτροσυγκόλληση τόξου παράγεται θερμότητα που δημιουργείται από το ηλεκτρικό τόξο μέσω του ηλεκτροδίου και των μετάλλων. Η συγκόλληση αυτή γίνεται κατά μήκος της επιφάνειας συγκόλλησης.

- *Ρομπότ σημειακής συγκόλλησης*
  Η πρώτη εφαρμογή των ρομπότ σημειακής συγκόλλησης ήταν η συγκόλληση αμαξωμάτων αυτοκίνητων και εγκαταστάθηκαν το 1969 στο εργοστάσιο της general motors.Τα ρομπότ σημειακής συγκόλλησης τροφοδοτούνται με υδραυλική ενέργεια λόγο του βάρους που προσδίδει το πιστόλι συγκόλλησης στο ρομποτ. Ο έλεγχος της κίνησης πραγματοποιείτε με την μέθοδο από σημείο σε σημείο (ptp). Η λειτουργία της σημειακής συγκόλλησης είναι πολύ γρήγορη, με αρκετές σημειακές συγκολλήσεις να γίνονται ανά δευτερόλεπτο όταν η απόσταση μεταξύ των σημειακών συγκολλήσεων είναι 25-50 χιλιοστά. Η επαναληψημότητα του ρομπότ ως προς την θέση συγκόλλησης είναι επαρκής στο +- 1 χιλιοστό.

- *Ρομπότ ηλεκτροσυγκόλλησης τόξου*
  Τα ρομποτικά συστήματα συγκόλλησης χρησιμοποιούν αναλώσιμο ηλεκτρόδιο καλωδίου με αυτόματο τροφοδότη σύρματος(mig) και μη αναλώσιμα ηλεκτρόδια βολφραμίου με προστατευτικό αέριο(tig). Το καλώδιο τροφοδοτείται από έναν κινητήρα με ρυθμιζόμενη ταχύτητα σε προκαθορισμένο ρυθμό και το προστατευτικό αέριο ρέει σε ένα σωλήνα κατά μήκος του ηλεκτροδίου. Μπορούν επίσης να χρησιμοποιηθούν πιστόλια ψύξης νερού. Τα ρομπότ με σερβοκινητήρα συνεχούς ρεύματος χρησιμοποιούνται συνήθως στη συγκόλληση τόξου, με ταχύτητες συγκόλλησης που κυμαίνονται από 254 έως 3048 mm/min. Τα ρεύματα συγκόλλησης κυμαίνονται από 100 έως 1200 A και το ρομπότ είναι υπεύθυνο για την οδήγηση του πιστολιού συγκόλλησης κατά μήκος της προγραμματισμένης τροχιάς. Το σύστημα ελέγχου εκτελεί είτε μέθοδο συνεχούς διαδρομής (CP) είτε από σημείο σε σημείο (PTP) και συνδέεται με τον εξοπλισμό συγκόλλησης για να συγχρονίσει την έναρξη και τον τερματισμό των κινήσεων του ρομπότ.

<u>Βαφή</u>

Η βαφή αποτελεί μια ανθυγιεινή εργασία για τον άνθρωπο κυρίως διότι: Το περιβάλλον του θαλάμου βαφής είναι ανθυγιεινό. Τα διαλυτικά υλικά που χρησιμοποιούνται στη βαφή με σπρέι είναι τοξικά και πρέπει να προστατεύονται με μάσκες και αερισμό καθαρού αέρα. Η περιοχή βαφής πρέπει να είναι απαλλαγμένη από σκόνη και σε ελεγχόμενη θερμοκρασία. Ο θόρυβος που προέρχεται από την εκροή αέρα μέσω των ακροφυσίων βαφής μπορεί να προκαλέσει μη αναστρέψιμη βλάβη στα αυτιά του ανθρώπου. Για τους παραπάνω λογούς χρήση ρομπότ στη βαφή με σπρέι έχει γίνει ολοένα και πιο δημοφιλής. Τα ρομποτ βαφής είναι ελαφριά διότι δεν χρειάζεται να μεταφέρουν φορτίο εκτός από το πιστόλι βαφής που το βάρος είναι περίπου ενός κιλού. Εκτός αυτού δεν απαιτείται υψηλή ακρίβεια θέσης και η επαναληψιμότητα διότι το σφάλμα της εφαρμογής βαφής είναι σε ορισμένο βαθμό ανεκτό. Τέλος ο έλεγχος του ρομπότ είναι τύπου συνεχούς διαδρομής με επαναληψιμότητα της τάξης των 2 χιλιοστών.

<u>Συναρμολόγηση</u>

Τα βιομηχανικά ρομπότ χρησιμοποιούνται κυρίως στην συναρμολόγηση μικρών προϊόντων όπως ηλεκτρικοί διακόπτες και μικροί κινητήρες. Τα ρομποτικά συστήματα συναρμολόγησης είναι

προγραμματιζόμενα και παρέχουν μια οικονομικά αποδοτική λύση για τη συναρμολόγηση μικρών σε μέγεθος παρτίδων και για παρτίδες που περιέχουν διαφορετικά προϊόντα. Τα ρομπότ μπορούν να σχεδιαστούν σε οποιοδήποτε σύστημα συντεταγμένων, καρτεσιανό, κυλινδρικό, σφαιρικό ή αρθρωτό. Για την αντιμετώπιση περίπλοκων εργασιών συναρμολόγησης προθέτονται οπτικοί η ακουστικοί αισθητήρες.

Μηχανουργική κατεργασία

Η διάτρηση είναι η μόνη μηχανική εργασία που γίνεται με επιτυχία με ρομπότ, κυρίως στη βιομηχανία αεροσκαφών. Επίσης τα ρομπότ έχουν την ικανότητα να εκτελούν την διαδικασία λείανσης των μεταλλικών επιφανίων καθώς και να αφαιρούν τα γρέζια που έχουν απομείνει στις επεξεργασμένες επιφάνειες των επεξεργασμένων μετάλλων.

- *Διάτρηση*
  Ο έλεγχος κίνησης του ρομπότ κατά την διάτρηση είναι τύπου από σημείο σε σημείο (ptp). Το τελικό σημείο δράσης κρατάει ένα φορητό πνευματικό τρυπάνι και εκτελεί σε κάθε τρυπά έναν σταθερό κύκλο διάτρησης και στην συνέχεια περνάει στην επόμενη οπή.
- *Λείανση*
  Τα γρέζια δημιουργούνται κατά τη μηχανική κατεργασία μεταλλικών μερών και είναι δαπανηρή η αφαίρεσή τους. Τα ρομπότ μπορούν να λύσουν τα περισσότερα προβλήματα αφαίρεσης γρεζιών ακολουθώντας πιστά τη μη αυτόματη μέθοδο. Υπάρχουν δύο βασικοί τρόποι για να κάνετε ρομποτική λείανση. Αν το εξάρτημα είναι ελαφρύ τότε ρομπότ το μεταφέρει στο εργαλείο λείανσης. Αν είναι βαρύ το ρομπότ κρατάει το εργαλείο λείανσης. Και στις δυο περιπτώσεις Η σχετική κίνηση μεταξύ του εργαλείου και του εξαρτήματος είναι τύπου CP (continuous path) με υψηλή επαναληψιμότητα (περίπου 0,2 mm) και εξαιρετικά ελεγχόμενη ταχύτητα. Η αφαίρεση γρεζιών είναι μια από τις πιο δύσκολες εργασίες για τα ρομπότ. [19]

Εικόνα 10 industrial robot painting [20]

2.3 Cobots (Συνεργατικά ρομποτ)

Η λέξη cobots προέρχεται από τις λέξεις collaborative robots που σημαίνει συνεργατικά ρομπότ. Έχουν κατασκευαστεί με τέτοιον τρόπο ώστε να εργάζονται σε συνεργασία με τον άνθρωπο σε ένα ασφαλές και παραγωγικό περιβάλλον.

Cobots vs traditional robots

Τα παραδοσιακά ρομπότ εκτελούν εργασίες σε περιβάλλοντα που δεν είναι καταλληλά για τους ανθρώπους. Σχεδιάζονται με σκοπό τον αυτοματισμό την αποδοτικότητα και την παραγωγή χωρίς την φυσική αλληλεπίδραση με τον άνθρωπο εκτός από τις εργασίες συντήρησης και ενεργοποίησης/ απενεργοποίησης του ρομποτ. Το περιβάλλον εργασίας είναι βρώμικο και επικίνδυνο και συνήθως αυτά τα ρομπότ βρίσκονται περιορισμένα και μακριά από τους ανθρώπους. Όταν στο περιβάλλον του βρεθεί κάποιο αντικείμενο τότε ένα παραδοσιακό ρομπότ λόγω της έλλειψης των αισθητήρων κρούσης, σπρώχνει το αντικείμενο έξω από τον χώρο εργασίας του.

Σε αντίθεση τα cobots εργάζονται συνεργατικά με τον άνθρωπο και διαθέτει απαραίτητα εργαλεία για την αποφυγή συγκρούσεων. Αν μια εξωτερική δύναμη τίθεται σε μια άρθρωση τότε αυτή η άρθρωση θα υποταχθεί σε αυτήν και θα κινηθεί από την αντίθετη κατεύθυνση της δύναμης η θα σταματήσει την κίνηση του για να αποφύγει την σύγκρουση. Όταν βρεθεί ένα αντικείμενο στον χώρο, το αναγνωρίζουν και διακόπτουν άμεσα την εργασία τους έτσι ώστε να αποφύγουν την σύγκρουση με το εμπόδιο. Όταν υπάρχουν δυο cobots στο ίδιο περιβάλλον επικοινωνούν μεταξύ τους μέσω του πρωτοκόλλου tcp/ip (σύνδεση με ethernet), με σκοπό την αποφυγή σύγκρουσης μεταξύ τους, αν βρεθούν σε μια κοινή διαμόρφωση εκτελώντας μια εργασία τότε ένα από τα δυο ρομπότ θα σταματήσει την εργασία του για να συνεχίσει το άλλο. Τα cobots μαθαίνουν μέσω της διδασκαλίας και δεν χρειάζονται σκληροπυρηνικό προγραμματισμό όπως τα βιομηχανικά ρομποτ.

Επαναληψημότητα

Στον κόσμο των συνεργατικών ρομποτικών συστημάτων, η ακρίβεια ενός ρομπότ δεν είναι το πιο σημαντικό κριτήριο. Αντιθέτως, η επαναληπτικότήτα είναι πιο σημαντική, καθώς τα cobots προγραμματίζονται συνήθως με διδασκαλία/καθοδήγηση με το χέρι και η ικανότητα του ρομπότ να επαναλαμβάνει ακριβώς τις ίδιες κινήσεις είναι πιο πολύτιμη από το να έχει ακρίβεια χιλιοστού στις διαμορφώσεις του χώρου εργασίας.

Εφαρμογές και εξατομίκευση

Τα cobots εκτελούν ένα πλήθος εργασιών μέσα στις οποίες είναι: βαφή, τριβή, συγκόλληση, βίδωμα , μεταφορά εξαρτημάτων. Ανάλογα με την εργασία υπάρχει η δυνατότητα τα cobots να τοποθετούνται σχετικά ευκολά σε διάφορες επιφάνειες , όπως στο ταβάνι η στον τοίχο. Ακόμα μπορούν να μεταφερθούν ευκολά από μια τοποθεσία σε μια άλλη λόγω του μικρού βάρους που έχουν χωρίς να επηρεαστεί το περιβάλλον της παραγωγής. [21]

Εικόνα 11 ur3 robot [22]

Το ur3 ρομποτ είναι ένα αρθρωτό ρομποτ που αποτελείτε από 6 περιστροφικές αρθρώσεις και ζυγίζει 11 κιλά με μέγιστο φορτίο ανύψωσης τα 3 κιλά και το τελικό σημείο δράσης μπορεί να φτάσει έως και 500mm.

Εικόνα 12 Το ρομποτ ur3 με αρπαγή εκτελεί εφαρμογή pick and place [23]

2.4 Κινηματική

Η κινηματική είναι ο κλάδος της μηχανικής που μελετά την σχέση μεταξύ των συντεταγμένων των ρομποτικών αρθρώσεων και της χωρικής διάταξης που βρίσκεται το ρομπότ χωρίς να λαμβάνει υπόψη τις εξωτερικές δυνάμεις όπως είναι η βαρύτητα και η τριβή. Η δυναμική είναι εκείνη που μελετά τις εξωτερικές δυνάμεις ενώ η κινοδυναμική είναι η μελέτη και των δυο.

Η κίνηση του ρομποτικού βραχίονα περιγράφεται από το ευθύ και το αντίστροφο κινηματικό πρόβλημα.

- Το ευθύ κινηματικό πρόβλημα βρίσκει ποια είναι η θέση και ο προσανατολισμός του άκρου εργασίας όταν γνωρίζουμε τις γωνίες (μοίρες) των αρθρώσεων του ρομποτικού βραχίονα.
- Το αντίστροφο κινηματικό πρόβλημα βρίσκει ποιες γωνίες των αρθρώσεων επιτυγχάνουν μια επιθυμητή θέση όταν γνωρίζουμε την θέση και τον προσανατολισμό του άκρου εργασίας του ρομπότ [24]

<u>Ευθύ Κινηματικό Πρόβλημα</u>

Ο κύριος τρόπος υπολογισμού του ευθέως κινηματικού προβλήματος αποτελεί η μέθοδος Denavit-hartenberg. Η μέθοδος DH χρησιμοποιεί τέσσερις παραμέτρους για να καθορίσει τη σχέση μεταξύ δύο γειτονικών συνδέσμων σε έναν ρομποτικό βραχίονα

- το μήκος του συνδέσμου.
- τη συστροφή του συνδέσμου.
- τη μετατόπιση συνδέσμου.
- τη γωνία άρθρωσης.

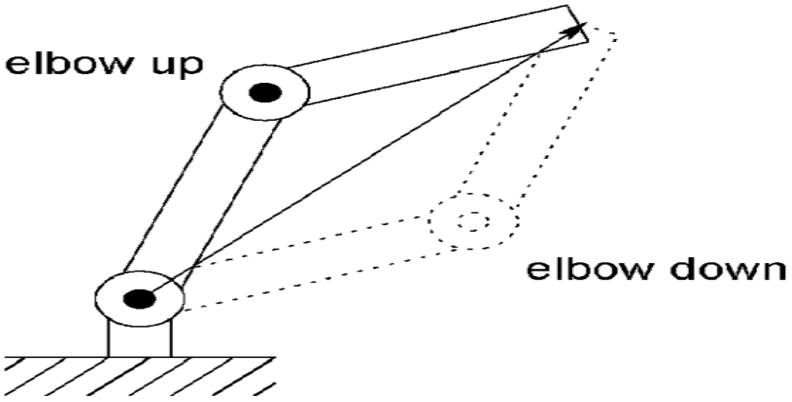
UR3				
Kinematics	theta [rad]	a [m]	d [m]	alpha [rad]
Joint 1	0	0	0.1519	π/2
Joint 2	0	-0.24365	0	0
Joint 3	0	-0.21325	0	0
Joint 4	0	0	0.11235	π/2
Joint 5	0	0	0.08535	-π/2
Joint 6	0	0	0.0819	0

Εικόνα 13 ur3 dh parameters [25]

Αυτές οι παράμετροι χρησιμοποιούνται για τον υπολογισμό των επιμέρους πινάκων μετασχηματισμού για κάθε σύνδεσμο του ρομποτικού βραχίονα και τελικώς τον πολλαπλασιασμό τους για τον υπολογισμό του ολικού πίνακα μετασχηματισμού. [26] [27]

Αντίστροφο Κινηματικό Πρόβλημα

Η αντιστροφή κινηματική είναι ένα πρόβλημα με δεδομένα την θέση του τελικού σημείου δράσης του βραχίονα και ζητούμενα τις θέσεις των αρθρώσεων. Αυτό το καθιστά αυτόματα ποιο δύσκολο πρόβλημα από την ευθεία κινηματική διότι όσο αυξάνεται ο αριθμός των αρθρώσεων του βραχίονα τόσο αυξάνεται και το πλήθος των πιθανών καταστάσεων που βρίσκονται οι αρθρώσεις αυτές.



Εικόνα 14 Οι 2 πιθανές καταστάσεις (elbow up/elbow down) [28]

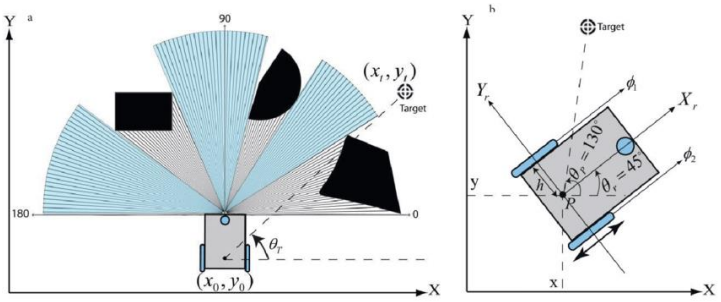
Στην παραπάνω εικόνα βλέπουμε έναν βραχίονα 3 αρθρώσεων που βρίσκεται σε δυο πιθανές καταστάσεις elbow up – elbow down. Και στις δυο καταστάσεις το σημείο και η απόσταση του τελικού σημείου δράσης του καθένα από την βάση είναι ιδίες, αυτό που διαφέρει όπως παρατηρούμε είναι η θέση του μεσαίας άρθρωσης. [29]

Η λύση αντίστροφης κινηματικής είναι ιδιαίτερα χρήσιμη για εργασίες όπως ο σχεδιασμός κίνησης και η δημιουργία τροχιάς, καθώς επιτρέπει στο ρομπότ να εκτελεί συγκεκριμένες εργασίες σε μια επιθυμητή θέση και προσανατολισμό. Η αναλυτική μέθοδος δίνει την αντιστροφή κινηματική λαμβάνοντας ως είσοδο την θέση του τελικού σημείου δράσεως και δίνει ως έξοδο τις θέσεις των αρθρώσεων. Δεν είναι ιδανική και μπορεί να μην βρεθεί η επίλυση της όταν το ρομποτ έχει πολλούς βαθμούς ελευθέριας. Μια άλλη μέθοδος αποτελεί την προσεγγιστική μέθοδο οπού προσαρμόζουμε τις γωνίες αρθρώσεων έως ότου το τελικό σημείο δράσης να φτάσει στην επιθυμητή θέση και προσανατολισμό (pose). [30]

2.5 Σημαντικές Έννοιες

Χώρος Διαμόρφωσης (Configuration space)

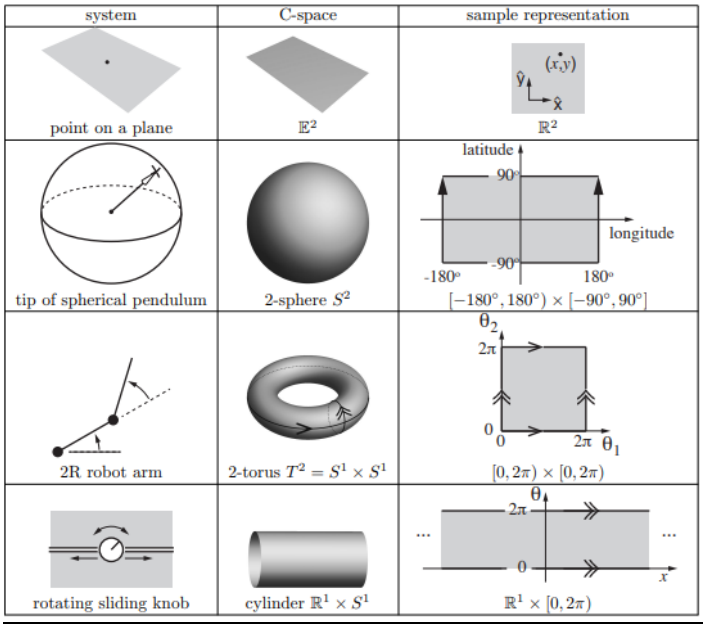
Η διαμόρφωση (configuration) ενός ρομπότ είναι ο προσδιορισμός των θέσεων όλων των σημείων του σε σχέση με ένα σταθερό σύστημα αναφοράς. Για να βρεθεί η διαμόρφωση πρέπει το σώμα να χαρακτηρίζεται ως άκαμπτο δηλαδή να μην υπόκειται σε κάποιου είδους παραμόρφωση, και να είναι γνωστής μορφής. Ο προσδιορισμός της θέσης ενός σημείου στον χώρο εξαρτάται αποκλειστικά από την διάσταση του, σε έναν δυσδιάστατο χώρο η διαμόρφωση του σημείου δίνεται από δυο τιμές τις συντεταγμένες (x,y) , ενώ σε έναν τρισδιάστατο χώρο χρειάζεται τρεις τιμές (x,y,z). Για την διαμόρφωση ενός άκαμπτου σώματος και όχι σταθερού σημείου υπάρχουν και άλλοι παράμετροι που λαμβάνουν μέρος, παραδείγματος χάρη η διαμόρφωση ενός αυτοκινουμένου ρομπότ που κινείται σε ένα ευθύ επίπεδο στον χώρο δίνεται ως q=(x,y,δ) οπού x,y οι συντεταγμένες του ρομπότ που κινείτε σε δυσδιάστατο χώρο και δ ο προσανατολισμός του ρομπότ.



Εικόνα 15 [31]

Για έναν ρομποτικό βραχίονα που αποτελείται από συνδέσμους σε σειρά η διαμόρφωση του μπορεί να εκφράζει τις γωνίες των αρθρώσεων αν είναι περιστροφικού τύπου και δίνεται ως q(θ1,θ2,θ3..θν) διαφορετικά αν είναι πρισματικού τύπου εκφράζει την απόσταση. Αν η διαμόρφωση είναι ένα σημείο σε κάποιο χώρο τότε αυτός ο χώρος ονομάζεται χώρος διαμόρφωσης. Ως παράμετροι διαμόρφωσης χαρακτηρίζονται οι συντεταγμένες πραγματικής αξίας όπως x,y,z,θ,δ. Για παράδειγμα για ένα κέρμα στον χώρο δεν μπορούμε να επιλέξουμε την μεταβλητή [κορώνα ,γράμματα] καθώς βρίσκονται στον διακριτό σύνολο και δεν αποτελούν μεταβλητές που σχετίζονται με την θέση του σώματος.

Η τοπολογία του χώρου διαμόρφωσης είναι μια μαθηματική έννοια που χρησιμοποιείται για να περιγράψει τις πιθανές διαμορφώσεις ενός συστήματος. Στην περίπτωση των δισδιάστατων συστημάτων, η τοπολογία του χώρου διαμόρφωσης μπορεί να πάρει διάφορα σχήματα, συμπεριλαμβανομένου του τόρου (σχήμα ντόνατς) , της σφαίρας και του κυλίνδρου.



Εικόνα 16 2d c-spaces [32]

Βαθμοί Ελευθέριας

Ο ελάχιστος αριθμός n μεταβλητών που απαιτούνται για την αναπαράσταση της διαμόρφωσης ονομάζεται βαθμοί ελευθέριας (degrees of freedom). Η διαμόρφωση παραδείγματος χάρη ενός κέρματος που βρίσκεται τοποθετημένο σε ένα επίπεδο περιγράφεται ως qk=(x,y,θ) οπού θ η γωνία περιστροφής του κέρματος . Το κέρμα μπορεί να μετακινηθεί στον άξονα x στον άξονα y αλλά και να περιστρέφει κατά γωνιά θ, οπότε λέμε ότι το κέρμα έχει συνολικά 3 βαθμούς ελευθέριας. Ένα άκαμπτο σώμα που κινείτε σε ένα δυσδιάστατο χώρο και είναι ελεύθερο (χωρίς περιορισμούς) έχει 3 βαθμούς ελευθερίας και ονομάζεται επίπεδο άκαμπτο σώμα (planar rigid body) ενώ ένα άκαμπτο σώμα που κινείται ελεύθερο έχει 6 βαθμούς ελευθερίας και ονομάζεται χωρικό άκαμπτο σώμα.

Παραδείγματος χάρη μια πόρτα που κινείται περιστροφικά γύρω από τον μεντεσέ της έχει διαμόρφωση qπ=Θ οπότε έχει βαθμούς ελευθερίας = 1. Αν αφαιρέσουμε τον μεντεσέ τότε οι βαθμοί ελευθερίας είναι 6, που σημαίνει ότι ο μεντεσές προσθέτει 5 ανεξάρτητους περιορισμούς στην πόρτα.

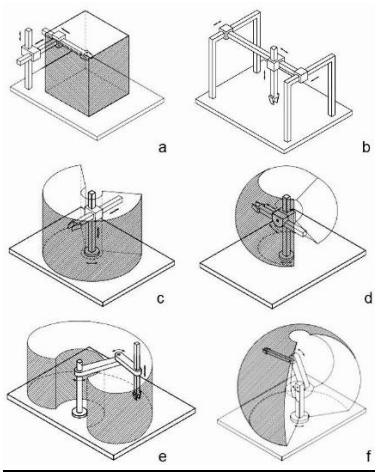
Task space

Είναι ο χώρος όπου το ρομπότ εκτελεί μια εργασία. Για παράδειγμα αν η εργασία ενός ρομπότ είναι να σχεδιάσει στο χαρτί με ένα μολυβί τότε ο χώρος αυτός εκφράζεται ως R^2. Αν πρέπει το ρομπότ μέσω του τελικού σημείου δράσης να χειραγωγήσει ένα άκαμπτο αντικείμενο στο χώρο διαμόρφωσης ο χώρος εκφράζεται μόνο από την εργασία που έχει να κάνει το ρομπότ.

[33] [34]

Work envelopes (Περιβάλλον εργασίας ρομπότ)

Το περιβάλλον εργασίας ενός βιομηχανικού ρομπότ εκφράζει το εύρος της κίνησης που μπορεί να επιτύχει ένα ρομπότ. Όπως βλέπουμε και στην (εικόνα 10) διαφορετικοί τύποι ρομπότ έχουν διαφορετικά εύρη κίνησης. Το σχήμα του εκφράζει όλα τα σημεία που μπορεί να φτάσει το τελικό σημείο δράσης ενός ρομπότ και εξαρτάται από τους τύπους των αρθρώσεων του (πρισματική, περιστροφική). Παραδείγματος χάρη ένα καρτεσιανού τύπου ρομπότ που αποτελείται από 3 αρθρώσεις πρισματικού τύπου, οι διαμορφώσεις που μπορεί να φτάσει το τελικό σημείο δράσης σχηματίζουν ένα τετράγωνο σχήμα. [35]



Εικόνα 17 Διαφορετικοί τύποι ρομπότ και τα περιβάλλοντα εργασίας τους (a) Καρτεσιανό (b) gantry (c) Κυλινδρικός (d) Σφαιρικό (e) SCARA (f) αρθρωτό [36]

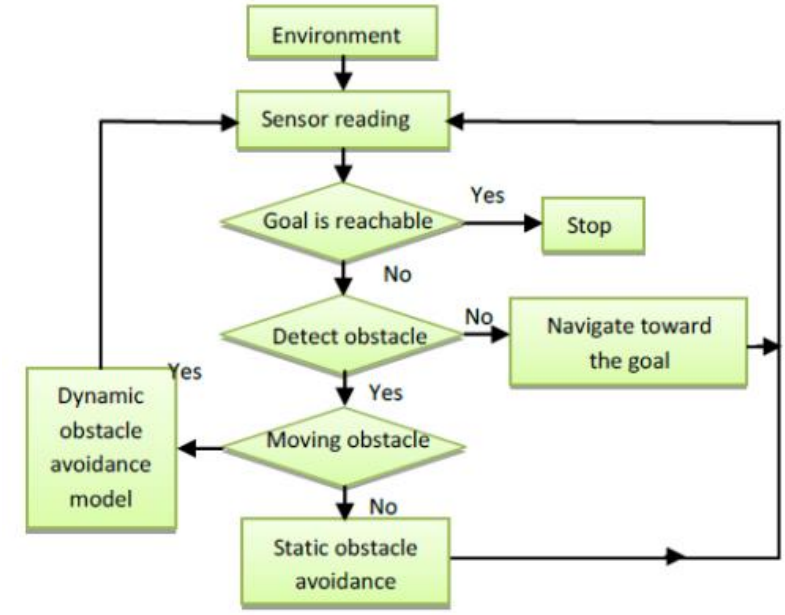
3 Αποφυγή Εμποδίων

Ένας ρομποτικός βραχίονας αλληλοεπιδρά με ένα αντικείμενο είτε αναγνωρίζοντας το ως κομμάτι της εργασίας του (μετακίνηση ενός αντικειμένου) είτε ως εμπόδιο (αποφυγή ενός εμποδίου). Ανεξάρτητα από το περιβάλλον που εργάζεται ένας βραχίονας είτε είναι οργανωμένο είτε ανοργάνωτο υπάρχει η πιθανότητα να αντιμετωπίσει κάποιο εμπόδιο. Ο κύριος τρόπος αποφυγής εμποδίου αποτελεί ο σχεδιασμός κίνησης (motion planning). Ο σχεδιασμός κίνησης του ρομπότ εξαρτάται από την δομή του περιβάλλοντος καθώς και από την φύση των εμποδίων. [37]

Ο σχεδιασμός κίνησης βάση του περιβάλλοντος κατηγοριοποιείτε σε global σχεδιασμό μονοπατιού και σε local σχεδιασμό μονοπατιού. Αν γνωρίζουμε την δομή του περιβάλλοντος πριν εκτελέσουμε τον αλγόριθμο τότε μιλάμε για παγκόσμιο(global) σχεδιασμό μονοπατιού αν δεν το γνωρίζουμε τότε μιλάμε για τοπικό(local) σχεδιασμό μονοπατιού. Ανάλογα με την συμπεριφορά των εμποδίων ο σχεδιασμός κίνησης κατηγοριοποιείτε σε στατικό η offline οπού το περιβάλλον περιέχει μη κινούμενα εμπόδια και η κίνηση του ρομπότ στον χώρο εργασίας του είναι προγραμματισμένη από πριν. Στην άλλη κατηγορία έχουμε το δυναμικό σχεδιασμό κίνησης οπού τα εμπόδια δεν είναι στατικά αλλά κινούνται στο

περιβάλλον η μπορεί να προκύψουν ανά πάσα στιγμή εκεί οι αποφάσεις του ρομπότ πρέπει να είναι δυναμικές η αλλιώς on-line και να αλλάζουν σύμφωνα με τα δεδομένα του περιβάλλοντος. [38]

Συμπερασματικά η global στρατηγική περιορίζετε σε εφαρμογές που αποτελούνται από στατικά και καλά καθορισμένα περιβάλλοντα. Ενώ η local στρατηγική δεν προορίζονται να αντικαταστήσουν τις παγκόσμιες, υψηλότερου επιπέδου μεθόδους σχεδιασμού διαδρομής και είναι υπολογιστικά λιγότερο απαιτητικές από τις παγκόσμιες μεθόδους. Ωστόσο, μπορεί να προκαλέσουν μη βέλτιστη συμπεριφορά ή να κολλήσουν όταν δεν μπορεί να βρεθεί μια διαδρομή χωρίς σύγκρουση. [37]

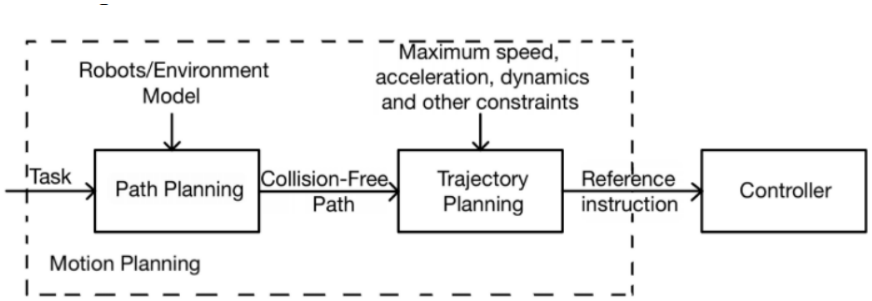


Εικόνα 18 Διάγραμμα Ροής της διαδικασίας αποφυγής εμποδίου [38]

Η εικόνα 13 είναι ένα Διάγραμμα ροής που εξηγεί την διαδικασία αποφυγής εμποδίου. Αρχικά διαβάζει τα δεδομένα του περιβάλλοντος από έναν αισθητήρα. Αν το ρομπότ δεν βρίσκεται κοντά στον στόχο, ψάχνει να βρει αν κοντά του υπάρχει κάποιο εμπόδιο. Αν βρίσκεται κάποιο εμπόδιο τότε ελέγχει αν το εμπόδιο κινείται η είναι ακίνητο και ανάλογα με αυτά τα δεδομένα χρησιμοποιεί διαφορετική στρατηγική αποφυγής του εμποδίου.

3.1 Σχεδιασμός Κίνησης (Motion Planning)

Ο σχεδιασμός κίνησης (motion planning) αποτελείται από τον σχεδιασμό διαδρομής (path planning) και τον σχεδιασμό τροχιάς (trajectory planning). Σκοπός του σχεδιασμού κίνησης είναι η ομαλή μετάβαση του τελικού σημείου δράσης του ρομπότ από ένα αρχικό σημείο προς ένα τελικό σημείο.



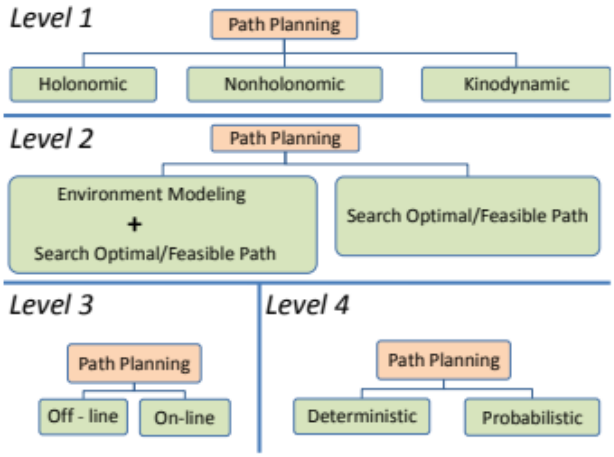
Εικόνα 19 Ο σχεδιασμός κίνησης ενός ρομπότ αποτελείται από τον σχεδιασμό μονοπατιού και τον σχεδιασμό τροχιάς. [24]

Η διαδρομή δημιουργείται από μια ακολουθία σημείων που συνδέουν την αρχική με την τελική θέση. Η στρατηγική με την οποία δημιουργείται το μονοπάτι ονομάζεται σχεδιασμός διαδρομής (path planning) με σημεία διαδρομής να αποτελούν θέσεις ή γωνίες αρμών στο χώρο. Σκοπός του σχεδιασμού διαδρομής είναι να δημιουργήσει την μέγιστη απόσταση μεταξύ της διαδρομής και του εμποδίου και ταυτόχρονα να ελαχιστοποιήσει την απόσταση του μονοπατιού από την αρχικό σημείο έως το τελικό.

Ο σχεδιασμός τροχιάς δίνει πληροφορίες σχετικά με τον χρόνο της διαδρομής και έχει ως σκοπό να ελαχιστοποιήσει τους χρόνους που το ρομπότ διασχίζει τα σημεία στο μονοπάτι η να μειώσει την ενέργεια της κίνησης του ρομπότ στον χώρο των αρθρώσεων. [39]

3.1.1 Σχεδιασμός διαδρομής (path planning)

Στο κείμενο (path planning : A 2013 survey) οι συγγράφεις έχουν διαχωρίσει τα προβλήματα σχεδιασμού κίνησης σε 4 επίπεδα ανάλογα με το είδος του προβλήματος.



Εικόνα 20 Κατηγοριοποίηση των επιπέδων σχεδιασμού μονοπατιού [40]

Στο πρώτο επίπεδο αναφέρονται οι περιορισμοί της κίνησης του τελικού σημείου δράσης του ρομπότ στον χώρο. Ένας ρομποτικός περιορισμός είναι ένας περιορισμός που τίθεται σε ένα ρομποτικό σύστημα που περιορίζει τις εφικτές δυνατότητες κίνησης του. Οι περιορισμοί μπορεί να περιλαμβάνουν τόσο τους μηχανικούς περιορισμούς του συστήματος, όπως μια άρθρωση που βρίσκεται στο όριο της, όσο και τους περιορισμούς που εφαρμόζονται από τον χρήστη, όπως η επιθυμητή θέση του εργαλείου. Οι περιορισμοί διακρίνονται σε ολονομικούς, μη ολονομικούς και κινοδυναμικούς περιορισμούς.

Στο δεύτερο επίπεδο γίνεται ο διαχωρισμός ανάμεσα σε δυο στρατηγικές που χρησιμοποιούνται για να εκτελέσουν έναν σχεδιασμό κίνησης. Υπάρχει η στρατηγική οπού χρησιμοποιούνται αλγόριθμοι όπως ο a* και ο Dijkstra και απαιτούν την μοντελοποίηση του περιβάλλοντος πριν ξεκινήσουν την αναζήτηση του μονοπατιού ενώ μια άλλη στρατηγική είναι οι αλγόριθμοι όπως ο RRT η potential fields οπού δεν είναι αναγκαία η μοντελοποίηση του περιβάλλοντός.

Στο τρίτο επίπεδο γίνεται ο διαχωρισμός ανάμεσα σε αλγόριθμους που χρησιμοποιούνται καλυτέρα online και σε αλγόριθμους που χρησιμοποιούνται μόνο off line. Γενικά οι on-line αλγόριθμοι χρησιμοποιούνται χωρίς το βήμα μοντελοποίησης περιβάλλοντός αλλά υπάρχουν και περιπτώσεις οπού αλγόριθμοι που χρειάζονται την μοντελοποίηση του περιβάλλοντος χρησιμοποιούνται σε online περιπτώσεις. Ως offline σχεδιασμός θεωρείται όταν ένας αλγόριθμος υπολογίζει το σχέδιο κίνησης με βάση τις πληροφορίες που έχουν ενταχθεί εκείνη την στιγμή χωρίς να ενημερώνει όταν υπάρξουν νέες πληροφορίες. Παραδείγματος χάρη εάν ενταχθεί στον χώρο ένα νέο εμπόδιο δεν θα αλλάξει η σχεδίαση της κίνησης. Σε αντίθεση ο online σχεδιασμός δημιουργεί συνεχώς ένα νέο σχέδιο κίνησης και ενημερώνει το προηγούμενο όταν υπάρξουν νέες πληροφορίες και όλα αυτά γίνονται σε πραγματικό χρόνο.

Στο τέταρτο και τελευταίο επίπεδο ο σχεδιασμός μονοπατιού διακρίνεται σε αιτιοκρατικό και πιθανολογικό σχεδιασμό. Ως αιτιοκρατική μέθοδο βρίσκονται οι αλγόριθμοι όπου επιτρέπουν την επίτευξη του ιδίου αποτελέσματος σε κάθε εκτέλεση με τις ιδίες αρχικές συνθήκες. Αυτοί οι αλγόριθμοι είναι ο Dijkstra και ο A*. Αυτή η μέθοδος έχει το μειονέκτημα ότι σε καταστάσεις πραγματικού χρόνου δεν μπορούν να ανταπεξέλθουν διότι δεν προσαρμόζονται στα νέα δεδομένα. Η λύση έρχεται με τους πιθανολογικούς αλγόριθμους όπως probabilistic road mapping και rrt που σε real time περιβάλλοντα αναπροσαρμόζονται και δημιουργούν νέα μονοπάτια συνεχώς.

Συμπερασματικά για η σχεδίαση ενός μονοπατιού εξαρτάται από ορισμένους παράγοντες οι οποίοι είναι: το είδος του ρομπότ που έχουμε επιλέξει, αν η σχεδίαση κίνησης γίνεται σε πραγματικό χρόνο, αν χρειάζεται να γίνει μοντελοποίηση του περιβάλλοντος και αν ο σχεδιασμός απαιτεί τον ανασχεδιασμό του μονοπατιού ανάλογα με τα νέα δεδομένα που θα προκύψουν. Οι αλγόριθμοι που διαφοροποιήθηκαν και τονίστηκαν σε μεγαλύτερο βαθμό από τους υπόλοιπους με βάση τους παραπάνω παράγοντες ήταν : Dijkstra, A*, probabilistic roadmaps , RRT [40]

3.1.2 Σχεδιασμός Τροχιάς (Trajectory Planning)

Ο σχεδιασμός τροχιάς ενός ρομποτικού βραχίονα περιγράφει τον τρόπο εκτέλεσης μιας τροχιάς με την πάροδο του χρόνου. Μια κυριά κατηγοριοποίηση σχεδίασης τροχιάς είναι μεταξύ της τροχιάς στον χώρο αρθρώσεων (joint space) και της τροχιάς στον χώρο εργασίας (task space) δηλαδή χώρο τελικού σημείου δράσεως.

Task space

Εδώ τα ενδιάμεσα σημεία και η παρεμβολή συμβαίνουν στην καρτεσιανή θέση και προσανατολισμό μιας συγκριμένης τοποθεσίας του τελικού σημείου δράσεως.

Το τελικό σημείο δράσεως κινείται πιο φυσικά στο χώρο οπότε είναι πιο εύκολο να χειριστεί αντικείμενα αντίθετα οι αρθρώσεις του όμως δεν κινούνται φυσικά . Το πιο σημαντικό πρόβλημα είναι ότι κατά την διάρκεια της τροχιάς είναι απαραίτητο να βρεθεί πολλαπλές φορές το αντίστροφο κινηματικό πρόβλημα πράγμα που υπολογιστικά ακριβή και πιο αργό στην εκτέλεση της τροχιάς.

Joint space

Τα σημεία του μονοπατιού και η παρεμβολή της τροχιάς βρίσκονται στα σημεία των αρθρώσεων του ρομπότ. Δεν είναι τόσο υπολογιστικά ακριβό διότι χρησιμοποιεί την επίλυση του αντίστροφου κινηματικού προβλήματος μόνο στα σημεία της διαδρομής για αυτόν τον λόγο είναι γρήγορο στην εκτέλεση. Το μόνο μειονέκτημα είναι ότι επειδή βρίσκονται τα ενδιάμεσα σημεία στον χώρο των αρθρώσεων δεν εγγυάται ότι η τροχιά θα σεβαστεί τους περιορισμούς των αρθρώσεων η τις συγκρούσεις.

Ανεξάρτητα με την παραπάνω επιλογή άλλοι τρόποι δημιουργίας τροχιάς αποτελούν οι παρακάτω:

- τραπεζοειδείς τροχιές (Trapezoidal Velocity)
- Πολυώνυμη (Polynomial)
- Spline

Σε αντίθεση με τις θέσεις, ο προσανατολισμός παρεμβολής μπορεί να είναι λίγο πιο δύσκολος, καθώς οι γωνίες τυλίγονται συνεχώς. Μια τέτοια τεχνική ονομάζεται Spherical Linear Interpolation (Slerp), η οποία βρίσκει τη συντομότερη διαδρομή μεταξύ δύο προσανατολισμών με σταθερή γωνιακή ταχύτητα γύρω από έναν σταθερό άξονα. [41]

3.2 Κατηγοριοποίηση Αλγορίθμων Σχεδιασμού Κίνησης

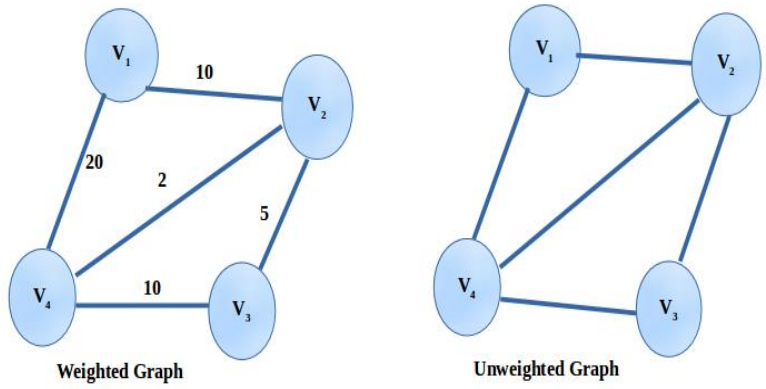
Οι αλγόριθμοι σχεδιασμού κίνησης διακρίνονται σε

- Graph Based Algorithms
- Sampling Based Algorithms

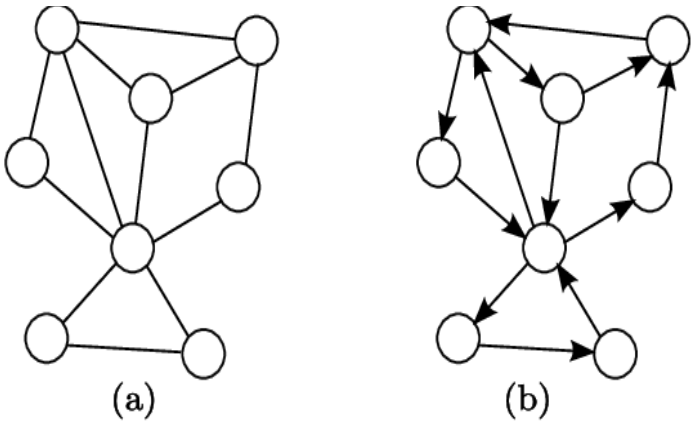
3.2.1 Graph Based Algorithms

Οι αλγόριθμοι που βασίζονται σε γραφήματα περιλαμβάνουν την αναπαράσταση του προβλήματος ως γράφημα και στη συνέχεια τη χρήση της θεωρίας γραφημάτων για την επίλυσή του.

Ένας Γράφος (G) αποτελείται από ένα σύνολο από κόμβους(V) και ακμές(E) και συμβολίζεται ως G = (V,E). Ένας κόμβος αναπαριστά μια τοποθεσία στον χώρο ενώ η ακμή αναπαριστά την σύνδεση 2 κόμβων. Τα γραφήματα κατηγοριοποιούνται βάση της κατεύθυνσης και του κόστους. Όταν για παράδειγμα ένα ρομπότ μπορεί να κινείται από έναν κόμβο σε έναν άλλο αλλά όχι αντίστροφα τότε ονομάζουμε τον Γράφο κατευθυνόμενο ενώ όταν μπορεί να κινηθεί και αντίθετα ονομάζεται μη κατευθυνόμενος. Όταν μια ακμή χαρακτηρίζεται από μια μη αρνητική τιμή τότε αυτή η τιμή ονομάζεται κόστος και αναπαριστά την τιμή της διέλευσης από τον ένα κόμβο στον άλλο. Τέλος μια διαδρομή που ξεκινά από μια δεδομένη κορυφή και τελειώνει στην ίδια κορυφή ονομάζεται κύκλος. [42]

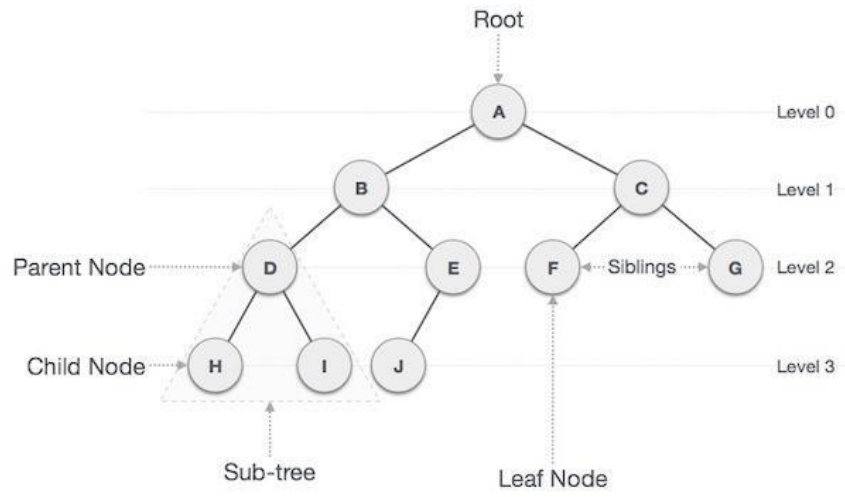


Εικόνα 21 Κατηγοριοποίηση Γράφου βάση Κόστους. Αριστερά: Γράφος κόστους, Δεξιά: Γράφος χωρίς κόστος [43]



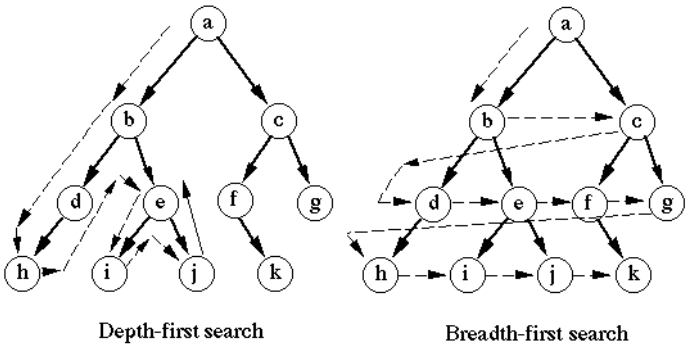
Εικόνα 22 Κατηγοριοποίηση Γράφου βάση κατεύθυνσης. (α): Μη κατευθυνόμενος Γράφος (β): Κατευθυνόμενος Γράφος [44]

Ένα είδος γραφήματος είναι το Δένδρο. Το δένδρο είναι ένας κατευθυνόμενο Γράφημα χωρίς κύκλο. Ξεκινάει από έναν αρχικό κόμβο που ονομάζεται ριζά του δένδρου. Χρησιμοποιεί την αναλογία Γονέα-Παιδιού για να κατηγοριοποιήσει τους κόμβους. Ένας κόμβος τύπου Γονέα διαθέτει κόμβους κάτω από εκείνον που ονομάζονται παιδιά. Τέλος ο κόμβος που δεν διαθέτει παιδιά ονομάζεται φύλλο. [42]



Εικόνα 23 Γράφος τύπου Δένδρου και η κατηγοριοποίηση των κόμβων του. [45]

Οι αλγόριθμοι Γράφου χωρίζονται σε τρεις μεθόδους αναζήτησης: Depth first, Breadth first και Best first. Οι Depth first ψάχνουν βαθιά και γρηγορά μέσα στον Γράφο ενώ οι Breadth first ελέγχουν όλους τους γειτονικούς κόμβους πριν πάνε να ψάξουν βαθιά στον Γράφο. Για να επιλέξουμε έναν από τους δυο θα πρέπει να γνωρίζουμε αν ο κόμβος που αναζητάμε βρίσκεται κοντά στον αρχικό κόμβο η βρίσκεται βαθιά στον Γράφο. Οι best first προσθέτουν τους παράγοντες τιμή και κόστος στους κόμβους και τις ακμές του Γράφου έτσι ώστε να λαμβάνουν αποφάσεις για το προς τα που πρέπει να επεκτείνεται το δένδρο η πιο κλαδί του δένδρου πρέπει να επεκταθεί. Οι αλγόριθμοι Dijkstra και Astar είναι αλγόριθμοι τύπου Best first. [46] .



Εικόνα 24 Μέθοδοί Αναζήτησης Γράφου. Depth-first, Breadth-first [47]

Αλγόριθμος Dijkstra

Ο Edsger Wybe Dijkstra εισήγαγε τον Dijkstra αλγόριθμο το 1956, και το 1959 δημοσιεύθηκε. Αποτελεί έναν δημοφιλή αλγόριθμο που βρίσκει το κοντινότερο μονοπάτι σε έναν κατευθυνόμενο Γράφο. Τα πλεονεκτήματα του είναι ότι τείνει να είναι γρήγορος στην εύρεση λύσεων αλλά έχει μειονεκτήματα όπως ότι είναι λιγότερο ωφέλιμος όταν η απόσταση μεταξύ του αρχικού και του τελικού σημείου είναι μεγάλη αυτό το μειονέκτημα λύνεται μέσω της παραλλαγής Extended Dijkstra algorithm. Τέλος επειδή ο αλγόριθμος Dijkstra έχει σχεδιαστεί με σκοπό να αντιμετωπίζει μόνο μη αρνητικά κόστη, όταν στον

Γράφο εμφανίζονται αρνητικά κόστη τότε μπορεί να δώσει λάθος αποτελέσματα και έτσι να μην βρει το πλησιέστερο μονοπάτι. Τα βήματα του αλγορίθμου Dijkstra είναι:

1- Θέτω την τιμή της απόστασης του αρχικού κόμβου 0 και όλων των άλλων στο άπειρο.
2- Όσο υπάρχουν μη επισκέψιμοι κόμβοι, επιλέγω εκείνον με τη μικρότερη απόσταση και υπολογίζω τις αποστάσεις από τους γείτονές του.
3- Ενημερώνω τις αποστάσεις των γειτόνων εάν μπορούν να προσεγγιστούν πιο αποτελεσματικά μέσω του τρέχοντος κόμβου.
4- Επισημάνετε τον τρέχοντα κόμβο ως επισκέψιμο και επαναλάβετε το βήμα 2 μέχρι να γίνει επίσκεψη στον τελικό κόμβο.

Παραδείγματα εφαρμογών του αλγορίθμου Dijkstra αποτελεί η εύρεση το πλησιέστερου μονοπατιού μιας διαδρομής εκκένωσης κτήριού, προσδιορισμό των πλησιέστερων αποστάσεων των πόλεων. [48] .

Αλγόριθμος A-star

Ο αλγόριθμος A-Star είναι ένας από τους πιο γνωστούς αλγόριθμους σχεδιασμού διαδρομής, ο οποίος μπορεί να εφαρμόζεται σε μια μετρική ή τοπολογία χώρου διαμόρφωσης. Αυτός ο αλγόριθμος χρησιμοποιεί ευρετική αναζήτηση και αναζήτηση με βάση τη συντομότερη διαδρομή . Όσον αφορά την αναζήτηση, αυτός ο αλγόριθμος είναι καλός σε διάφορα περιβάλλοντα . Ο αλγόριθμος A-Star είναι ένας κλασικός αλγόριθμος αναζήτησης μονοπατιών και μπορεί να χρησιμοποιηθεί για την αναζήτηση λαβύρινθων . Ο αλγόριθμος A-Star καθοδηγεί τη βέλτιστη διαδρομή προς τον στόχο, εάν η ευρετική συνάρτηση h(n) είναι αποδεκτή, που σημαίνει ότι δεν θα υπερεκτιμήσει ποτέ το αρχικό κόστος ή το πραγματικό κόστος .

Συνάρτηση αξιολόγησης f(n) = g(n) + h(n), όπου

g(n) = κόστος μέχρι στιγμής για την επίτευξη n.

h(n) = εκτιμώμενο κόστος από n έως στόχο (στόχος).

f(n) = εκτιμώμενο συνολικό κόστος της διαδρομής μέσω n προς τον στόχο. [49]

3.2.2 Sampling Based Algorithms

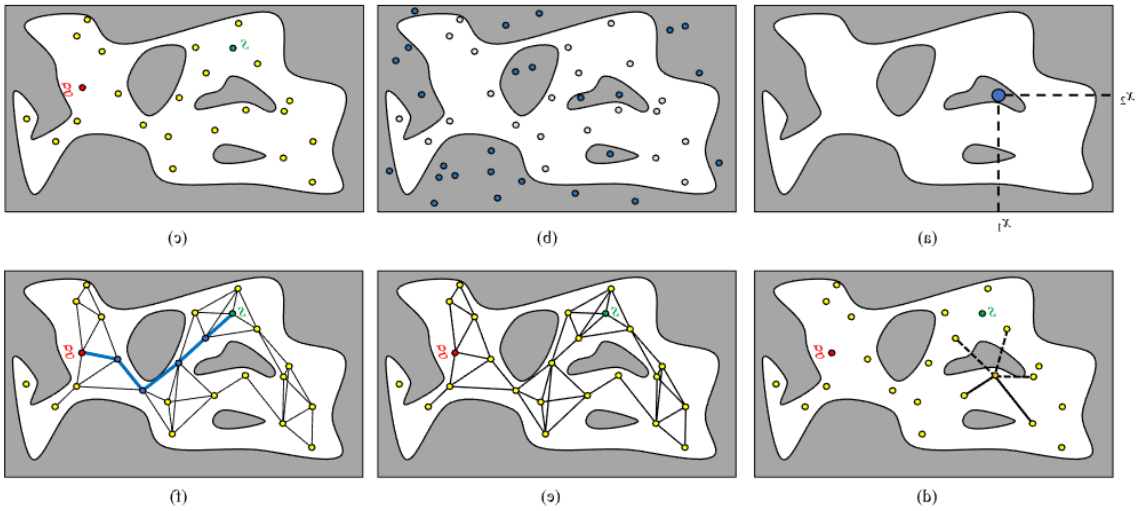
Οι αλγόριθμοι βάση δειγματοληψίας (sampling based) εκτελούν τυχαία δειγματοληψία έναν σταθερό χώρο εργασίας για να δημιουργήσουν ημι-βέλτιστες διαδρομές. Οι αλγόριθμοι βάση δειγματοληψίας κατηγοριοποιούνται σε ενεργητικούς και παθητικούς. Παράδειγμα ενεργού αλγορίθμου δειγματοληψίας είναι ο RRT αλγόριθμος που μπορεί να επιτύχει την καλύτερη δυνατή διαδρομή προς τον στόχο με τη δική του διαδικασία επεξεργασίας. Ένα παράδειγμα παθητικού αλγορίθμου βάση δειγματοληψίας είναι ο (PRM) που δημιουργεί ένα οδικό δίκτυο από την αρχική μέχρι την τελική διαμόρφωση και στην συνέχεια επιλέγει έναν άλλο αλγορίθμου για την εύρεση του πλησιέστερου μονοπατιού, επομένως ένας συνδυασμός αλγορίθμων αναζήτησης για την επιλογή της καλύτερης εφικτής διαδρομής στον χάρτη δικτύου όπου υπάρχουν πολλά εφικτά μονοπάτια. [50]

Probabilistic roadmaps

Το 1996 Λυδία Ε. Καβράκη πρότεινε στην διδακτορική διατριβή της μια μέθοδο σχεδιασμού κίνησης ονόματι probabilistic roadmaps [51]. Ο Prm αλγόριθμος ανήκει στους πιθανολογικούς αλγορίθμους αφού λαμβάνει τυχαία δείγματα στον χώρο διαμόρφωσης του ρομπότ και στην συνέχεια χρησιμοποιεί έναν Local planner για να συνδέσει την αρχική με την τελική διαμόρφωση που ο χρήστης έχει προσδιορίσει. Ο αλγόριθμος prm αποτελείται από 2 φάσεις την φάση εκμάθησης και την φάση query.

Στην φάση εκμάθησης κατασκευάζεται ο οδικός χάρτη roadmap. Ο οδικός χάρτης αναπαρίσταται από έναν Γράφο με μορφή G = (V,E) οπού V είναι οι κομβόι δηλαδή το σύνολο διαμορφώσεων στον ελεύθερο χώρο και (E) οι ακμές που ενώνουν τις διαμορφώσεις που βρίσκονται στον ελεύθερο χώρο και αποτελούν ένα μονοπάτι που ενώνει δυο διαμορφώσεις (q1,q2). Αφού προσθέσει όλες τις εφικτές διαμορφώσεις προσθέτει και την αρχική (qinit) και τελική διαμόρφωση (qgoal).

Στην φάση query έχοντας ως δεδομένα την αρχική και τελική διαμόρφωση ο Planner προσπαθεί να συνδέσει αυτές τις διαμορφώσεις μέσω μιας ευθείας γραμμής αν αυτή υπάρχει. Αν συνδεθούν αυτές οι δυο διαμορφώσεις τότε γίνεται αναζήτηση μιας ακολουθίας από ακμές που συνδέουν αυτούς τους δυο κόμβους. Τέλος μετατρέπει αυτήν την ακολουθία των ακμών σε ένα εφικτό μονοπάτι για το ρομπότ υπολογίζοντας εκ νέου των αντίστοιχων τοπικών μονοπατιών και τη συνένωση τους. Οι τοπικές διαδρομές μπορούν να αποθηκευτούν στον οδικό χάρτη, αλλά αυτό θα αυξήσει τις απαιτήσεις αποθήκευσης του οδικού χάρτη. [52]



Εικόνα 25 Κύρια βήματα PRM αλγορίθμου. [53]

Rapidly random trees

Ο κύριος στόχος του rrt είναι να αναπτύξει ένα τυχαίο δένδρο σε ένα αρχικό σημείο και στην συνέχεια να αναπτύξει η δημιουργήσει ένα τυχαίο σημείο σε ένα άδειο χώρο. Η διαδικασία εξελίσσεται ως εξής: Βρες το σημείο στο τυχαίο δένδρο οπού βρίσκεται πλησιέστερα στο τυχαίο σημείο και επεκτείνετε κατά ένα βήμα από το κοντινότερο σημείο έως το τυχαίο σημείο. Όταν το εκτεταμένο βήμα δεν οδηγεί σε επιτυχή πλοήγηση μέσα από τα εμπόδια, νέοι κομβόι επιτυχώς προστίθενται στο τυχαίο δένδρο έως ότου ο στόχος- σημείο είναι εξίσου μέρος του τυχαίου δένδρου ,συνεπώς το μονοπάτι έχει βρεθεί. Κατά συνέπεια, ολοκληρώνεται η διαδικασία σχεδιασμού διαδρομής. Αν και ο αλγόριθμος RRT μπορεί βασικά να σχεδιάσει μια αποτελεσματική διαδρομή, η αποτελεσματικότητά του αναζήτησής του θα μειωθεί σημαντικά λόγω της κατά προσέγγιση ομοιόμορφης επέκτασης του τυχαίου δέντρου στον περιβάλλοντα χώρο. Στη συνέχεια αναλύεται λεπτομερώς ο αλγόριθμος RRT καθώς και οι παραλλαγές του. [54]

3.2.2 Αλγόριθμος RRT

Ο αλγόριθμος RRT προτάθηκε το 1998 από τον Steven M LaValle, επιστήμονα στον κλάδο της πληροφορικής και ένας από τους πρώτους δημιουργούς και επικεφαλής επιστήμονας της γνωστής κονσόλας επαυξημένης πραγματικότητας oculus rift. [55] Ο Kuffner βοήθησε στην εργασία του LaValle [rapidly-exploring random trees a new tool for path planning] κάνοντας πειράματα και παρέχοντας σημαντικές προτάσεις και αναφέρεται ως συν δημιουργός του αλγορίθμου rrt. Ο Kuffner ως μέρος της διδακτορικής διατριβής μαζί με την συνεργασία του LaValle ανέπτυξαν μια σημαντική παραλλαγή του αλγορίθμου rrt που τον βελτιώνει σημαντικά αφού είναι ιδανικός για χώρους αναζήτησης υψηλών διαστάσεων και έχει την ονομασία RRT-connect [56].

Μέθοδος rrt

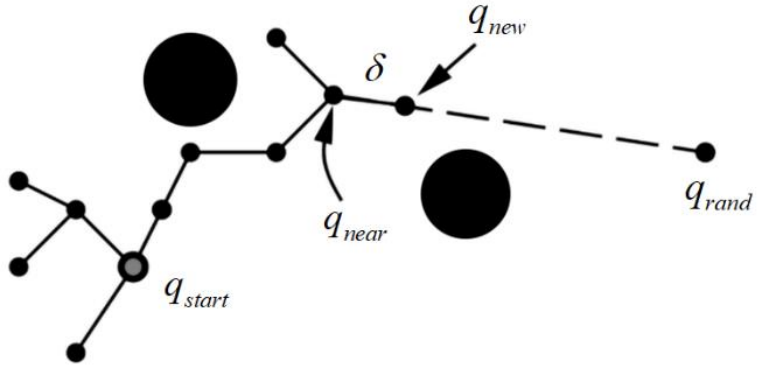
Ο αλγόριθμος RRT εκτελεί μια ραγδαία αναζήτηση στον χώρο διαμορφώσεων με σκοπό να δημιουργήσει ένα μονοπάτι που να ενώνει το σημείο εκκίνησης της διαδρομής με το σημείο στόχου. Οι κομβόι εκφράζονται ως σημεία στον χώρο διαμορφώσεων. Ο χώρος διαμορφώσεων αποτελείται από μια περιοχή εμποδίων qobst οπού ανήκει στο σύνολο q και πρέπει να αποφεύγεται, καθώς και την περιοχή qfree ανήκει στον q οπού το ρομπότ πρέπει να βρίσκεται. [57]

```
RRT(q_start, q_goal)
1    T.add(q_start)
2    q_new ← q_start
3    while(DISTANCE(q_new, q_goal) > d_threshold)
4       q_target = RANDOM_NODE()
5       q_nearest = T.NEAREST_NEIGHBOR(q_target)
6       q_new = EXTEND(q_nearest, q_target, expansion_time)
7       if(q_new != NULL)
8             q_new.setParent(q_nearest)
9             T.add(q_new)
10   ResultingPath ← T.TraceBack(q_new)
11   return ResultingPath
```

Εικόνα 26 βασικός αλγόριθμος RRT [58]

Ο αλγόριθμος ξεκινάει την αναζήτηση του δημιουργώντας ένα σημείο εκκίνησης qstart που αναφέρεται ως ρίζα του δένδρου. Στην συνέχεια επιλέγει τυχαία ένα σημείο στον χώρο διαμόρφωσης qrand. Αφού ο αλγόριθμος έχει δημιουργήσει μεγάλο αριθμό σημείων στον χώρο διαμορφώσεων q, γίνεται η αναζήτηση και η επιλογή του πλησιέστερου σημείου στο σημείο qrand. Αυτό το σημείο ονομάζεται qnear. Βάση των δυο σημείων qrand και qnear δημιουργείται ένα νέο σημείο qnew όπου βρίσκεται ανάμεσα τους σε συγκεκριμένη απόστασή. Το σημείο qnew έτσι ώστε να καταστεί ένα νέο σημείο αναφοράς ελέγχεται μέσω μιας συνθήκης Αν. Αν το σημείο qnew βρίσκεται εντός του χώρου qobst τότε το qnew απορρίπτεται και η διαδικασία εύρεσης σημείου επαναλαμβάνεται, αλλιώς αν βρίσκεται στον χώρο qfree το σημείο προστίθεται ως σημείο διαδρομής.

Για να βρεθεί το τελικό μονοπάτι πρέπει το σημείο qnew θα πρέπει να ελεγχθεί βάση της απόστασης από το τελικό σημείο qtarget. Αν το σημείο βρίσκεται σε μικρότερη ίση από την καθορισμένη απόστασή από το qtarget τότε το μονοπάτι βρέθηκε αλλιώς η διαδικασία εύρεσης νέου σημείου qnew συνεχίζεται έως ότου η συνθήκη απόστασης βρεθεί αληθής η οι φορές αναζητήσεων υπερβούν τον αριθμό επαναλήψεων. Τέλος ο αλγόριθμος επιστρέφει το μονοπάτι που βρέθηκε από το τελικό σημείο έως το αρχικό. [59] [57]

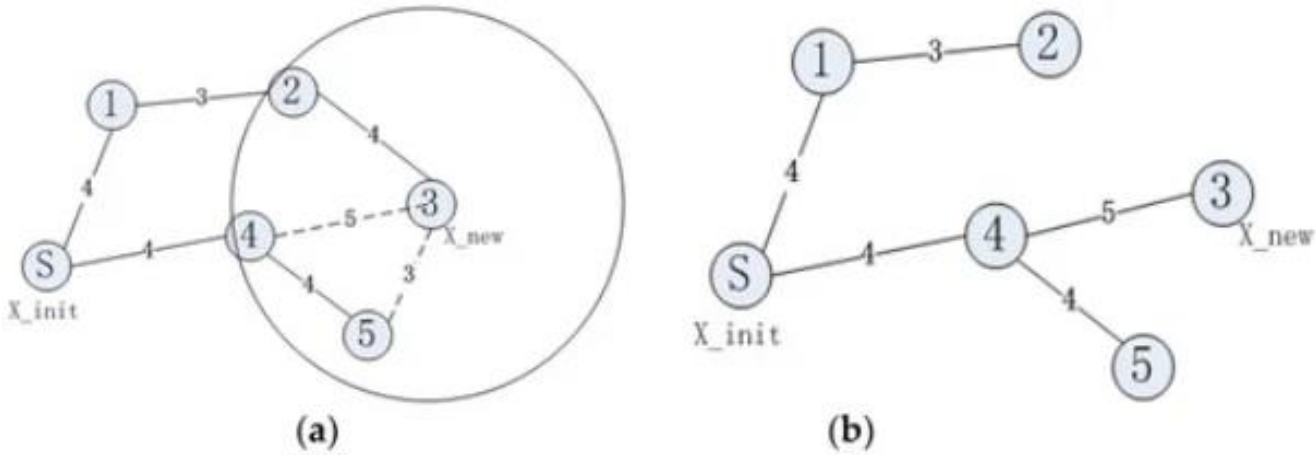


Εικόνα 27 Διάγραμμα διαδικασίας ανάπτυξης RRT αλγορίθμου [60]

Παραλλαγές RRT αλγορίθμου

- RRT-STAR
- RRT connect

RRT-Star

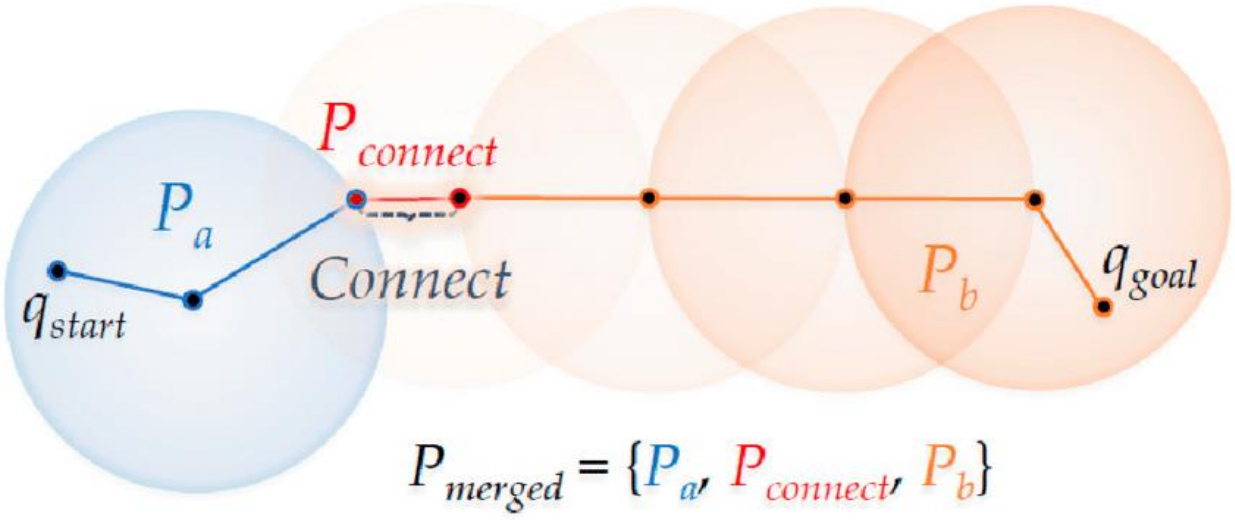


Εικόνα 28 (α) Διαδρομή σχεδιασμένη από τον παραδοσιακό αλγόριθμο RRT, (β) Βελτιστοποιημένη διαδρομή [54]

Το σχήμα (α) στην παραπάνω εικόνα δείχνει το σχεδιασμένο μονοπάτι από τον αλγόριθμο RRT και το σχήμα (β) δείχνει το σχεδιασμένο μονοπάτι του αλγορίθμου RRT_STAR. Όπως παρατηρούμε στην παραπάνω εικόνα ο κόμβος με αριθμό (3) είναι ο x_new και η διαδρομή που σχεδιάστηκε από τον αλγόριθμο RRT από το x_init έως το x_new είναι η 1,2,3. Ο κύκλος γύρω από το κόμβο 3 αποτελεί την ακτίνα του με κέντρο τον κόμβο 3. Όπως παρατηρούμε οι υπόλοιποι κομβόι εκτός από τον 1 και τον αρχικό βρίσκονται εντός της ακτίνας του κύκλου. Ο αλγόριθμος rrtstar ενώνει τον κόμβο 4 με τον 3 δημιουργώντας ένα βελτιστοποιημένο νέο μονοπάτι. [54]

RRT connect

Ο rrt connect δημιουργεί δυο δέντρα, ένα δέντρο ξεκινάει από το σημείο εκκίνησης ενώ το άλλο ξεκινάει από το σημείο στόχου. Σκοπός την δημιουργίας των δυο δέντρων είναι να συνδεθούν όσο το δυνατότερο συντομότερα μεταξύ τους μέσω ενός κόμβου ελέγχοντας ταυτόχρονα την περιοχή για πιθανά εμπόδια. Το δέντρο στόχου χρησιμοποιεί την βοήθεια της ευρετικής για να εξερευνήσει και να βρει πιο γρηγορά τον αρχικό κόμβο.



Εικόνα 29 rrt connect method [61]

4 Το Περιβάλλον ROS

Εισαγωγή

Το Ros αν και η ονομασία του προδιαθέτει σε ένα λειτουργικό σύστημα, όμως στην πραγματικότητα αποτελεί ένα ενδιάμεσο λογισμικό που διαθέτει εργαλεία βιβλιοθήκες και κανόνες για τον χειρισμό ρομπότ αναμεσά σε ένα πλήθος από διάφορου είδους πλατφόρμες. Ο ρομποτικός προγραμματισμός διαφέρει από τον κλασικό προγραμματισμό προγραμμάτων διότι διαθέτει την δυνατότητα να διαχειρίζεται σε αρμονία πολλά και διαφορετικού σκοπού ρομποτικά εξαρτήματα.

Ιστορική αναδρομή

Το 2007 το Stanford ai project ξεκίνησε το Ros με ένα όραμα. Το ρομποτικό περιβάλλον ανάπτυξης να παρέχει δωρεάν και να είναι ανοιχτού λογισμικού, να διαθέτει κυριά εξαρτήματα σε βιβλιοθήκες που να μπορούν ευκολά οι χρήστες να τα χρησιμοποιούν και να λειτουργεί με ήδη υπάρχοντα λογισμικά όπως είναι το gazebo για την προσομοίωση η το OpenCV. Το 2008 ο Scott Hassan (βασικός χρηματοδότης του Ros) μαζί με τον George willow πήραν κάτω από την ομπρελά τους το Ros με σκοπό να το μετατρέψουν σε μια πρότυπη πλατφόρμα ανάπτυξης ρομπότ. Το 2009 κυκλοφόρησε η έκδοση Ros 0.4 με το Pr2 κινητό ρομποτικό βραχίονα ως κύρια πλατφόρμα ανάπτυξης κερδίζοντας βραβεία σε πολλά πανεπιστήμια. Το 2012 η open source robotics foundation (osrf) ξεκίνησε να επιβλέπει το μέλλον του Ros με την να υποστήριξη την ανάπτυξη, διανομή και υιοθέτηση ανοιχτού λογισμικού και υλικού για χρήση σε πολλούς τομείς

Χαρακτηριστικά του Ros

Όπως τον σπονδυλωτό (modularity) σχεδιασμό του δηλαδή κάθε εξάρτημα που επιτελεί διαφορετικές εργασίες να αναπτύσσεται η λειτουργία του ξεχωριστά από τα αλλά σε ένα αρχείο κώδικα που ονομάζεται Node με τον κάθε κόμβο(Node) να μοιράζεται πληροφορίες με έναν άλλο. Διαφορετικές δυνατότητες ενός ρομπότ μπορεί να αναπτυχθεί σε πακέτα. Τα πακέτα μπορούν να εγκατασταθούν σε άλλους υπολογιστές.

Την δυνατότητα μετάδοσης μηνύματος δηλαδή την δυνατότητα των Nodes να επικοινωνούν μεταξύ τους. Για παράδειγμα ένα Node αισθητήρα στέλνει μηνύματα σε ένα άλλο παρέχοντας του πληροφορίες για το που βρίσκεται το εμπόδιο στο περιβάλλον με σκοπό την αποφυγή εμποδίου.

Built in algorithms: περιέχει πακέτα αλγορίθμων όπως PID, SLAM όπως και γνωστούς αλγορίθμους σχεδιασμού μονοπατιού όπως τον Dijkstra και rrt.

Παρέχει τρίτου τύπου βιβλιοθήκες όπως open cv και plc . Τέλος έχει μια τεράστια κοινότητα που βοηθάει στην ανάπτυξη του Ros αλλά και των χρηστών έτσι ώστε να αντιμετωπίσουν τα προβλήματα που τους έρχονται.

Ο προγραμματισμός με το Ros

Το Ros χρησιμοποιεί την δομή του subscriber/publisher για να επικοινωνεί μεταξύ των εξαρτημάτων και των Modules. Για να υποστηριχτεί αυτή η δομή χρειάζονται τα βασικά στοιχεία.

Nodes, Messages, topics, Master.

Nodes: Οι κόμβοι είναι ξεχωριστές εκτελέσιμες εφαρμογές που εκτελούν συγκεκριμένες εργασίες σε ένα σύστημα ROS. Κάθε κόμβος μπορεί να δημοσιεύσει ή να εγγραφεί σε ένα θέμα, παρέχοντας ή λαμβάνοντας πληροφορίες προς ή από άλλους κόμβους.

Messages: Τα messages είναι οι πληροφορίες που στέλνουν ή λαμβάνουν τα nodes για ένα topic. Ένα message ορίζεται ως μια δομή δεδομένων με συγκεκριμένα πεδία δεδομένων. Τα nodes μπορούν να στέλνουν ή να λαμβάνουν μηνύματα για ένα topic με δημοσίευση ή εγγραφή (publishing/subscribing).

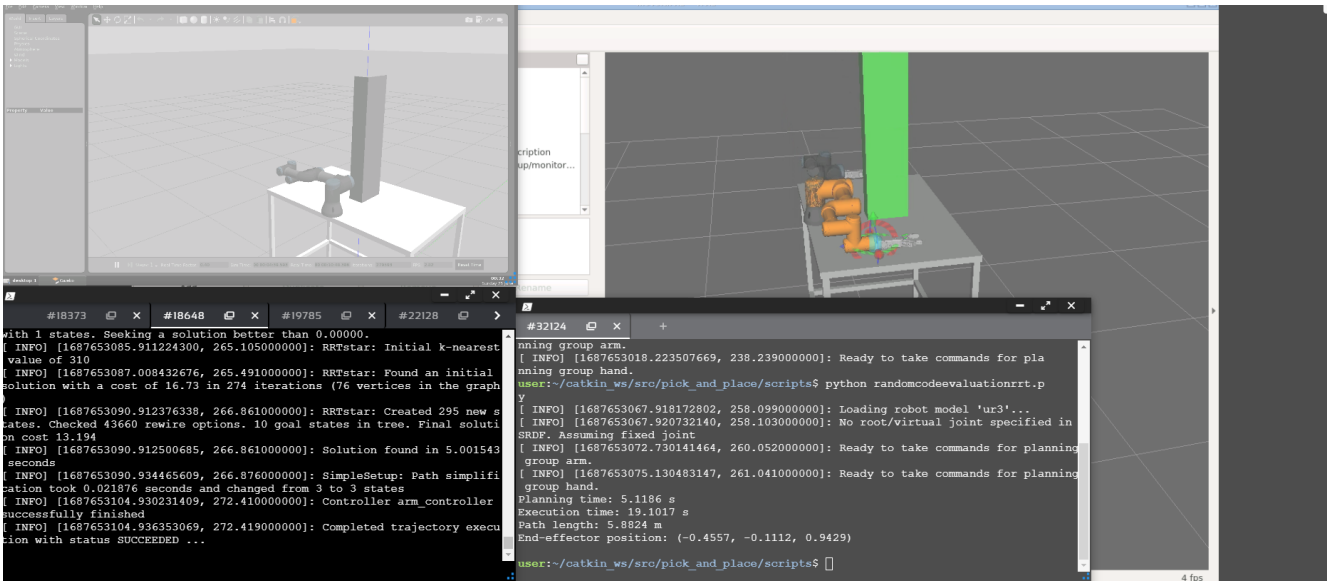
Topics: Τα topics είναι μια μορφή επικοινωνίας μεταξύ κόμβων σε ένα σύστημα ROS. Ένα topic ROS είναι ένας δίαυλος με όνομα μέσω του οποίου ανταλλάσσονται μηνύματα. Οι κόμβοι μπορούν να δημοσιεύουν(publish) μηνύματα σε ένα topic ή να εγγραφούν(subscribe) σε ένα topic για να λαμβάνουν μηνύματα.

Master: Το Master είναι ένα πρόγραμμα που διαχειρίζεται την ονομασία και την καταχώρηση node και topic σε ένα σύστημα ROS. Παρέχει στο σύστημα έναν κατάλογο με ενεργούς κόμβους και τα θέματα στα οποία δημοσιεύουν και εγγράφονται(subscribing/publishing). [62]

Εργαλεία Ros

Το MoveIt είναι ένα εργαλείο στο ROS που βοηθά τα ρομπότ να σχεδιάζουν και να εκτελούν τις κινήσεις τους. Είναι ένα πλαίσιο λογισμικού ανοιχτού κώδικα για σχεδιασμό κίνησης, χειρισμό, τρισδιάστατη οπτικοποίηση, κινηματική, έλεγχο και πλοήγηση. Το MoveIt προσφέρει μια σειρά από εργαλεία και πακέτα για τον έλεγχο ρομποτικών συστημάτων και την εκτέλεση εργασιών όπως ο σχεδιασμός κίνησης και η ανίχνευση σύγκρουσης. Παρέχει ενοποίηση με άλλα εργαλεία ROS όπως το RViz για οπτικοποίηση και εντοπισμό σφαλμάτων. Συμπερασματικά, το MoveIt είναι ένα ισχυρό εργαλείο που βοηθά τα ρομπότ να κινούνται και να αλληλεπιδρούν με το περιβάλλον τους με ασφαλή και αποτελεσματικό τρόπο.

Το Gazebo είναι ένα εργαλείο που χρησιμοποιείται στο ROS (Robot Operating System) για προσομοίωση. Επιτρέπει στους προγραμματιστές να προσομοιώνουν ρομπότ και το περιβάλλον τους σε έναν τρισδιάστατο χώρο. Μπορεί να χρησιμοποιηθεί για δοκιμή αλγορίθμων, σχεδιασμό ρομπότ και οπτικοποίηση ρομποτικών συστημάτων. Το Gazebo βοηθά τους προγραμματιστές να βελτιώσουν και να βελτιστοποιήσουν την απόδοση, χωρίς την ανάγκη δαπανηρών δοκιμών και δοκιμών σε φυσικό υλικό.



Εικόνα 30 Περιβάλλον Ros

5 Εφαρμογή

5.1 Εισαγωγή – Μεθοδολογία

Η μεθοδολογία που χρησιμοποιώ για να υλοποιήσω το project είναι η εξής: Αρχικά χρησιμοποιώ 3 βασικά μοντέλα για το project μου. Τον ρομποτικό βραχίονα που είναι ο ur3 από την universal robots, την αρπαγή του βραχίονα που είναι η rg2 αρπαγή και ένα μοντέλο γραφείου.

Μέσω του λογισμικού MoveIt θέτω τις ρυθμίσεις του βραχίονα. Χρησιμοποιήθηκε το gazebo ως προσομοιωτή και το MoveIt ως controller. Μέσω του MoveIt setup assistant τέθηκε ο βασικός αλγόριθμος σχεδιασμού διαδρομής RRT, καθώς και οι βασικές ρυθμίσεις ώστε ο ρομποτικός βραχίονας να είναι λειτουργικός.

Τα launch files που χρησιμοποιήθηκαν είναι 3. Ένα Launch file για την εκτέλεση του προσομοιωτή RViz, έναν για τον προσομοιωτή gazebo και ένα για το node move group το οποίο είναι υπεύθυνο για τον σχεδιασμό κίνησης και χειρισμού του ρομποτ.

Για την εκτέλεση της κίνησης του ρομποτ χρησιμοποιώ 4 προγράμματα σε γλώσσα Python.

- Πρόγραμμα που εκτελεί την τροχιά του ρομποτικού βραχίονα από μια αρχική θέση σε μια τελική και υπολογίζει ορισμένες παραμέτρους των αλγορίθμων σχεδιασμού διαδρομής ,
- Πρόγραμμα για την τοποθέτηση του εμποδίου στο gazebo την τοποθέτηση του εμποδίου στο MoveIt και RViz
- Πρόγραμμα για την επιστροφή του ρομποτικού βραχίονα στην αρχική του θέση.
- Πρόγραμμα για την εφαρμογή pick and place

Το Project αποτελείτε από 2 σενάρια:

- Στο πρώτο σενάριο γίνεται η εκτέλεση της κίνησης του ρομποτικού βραχίονα από ένα αρχικό έως ένα τελικό σημείο, χωρίς αποφυγή εμποδίου.
- Στο δεύτερο σενάριο προστίθεται ένα εμπόδιο στο gazabo αλλά και στο MoveIt και γίνεται η εκτέλεση της ιδιά τροχιάς.

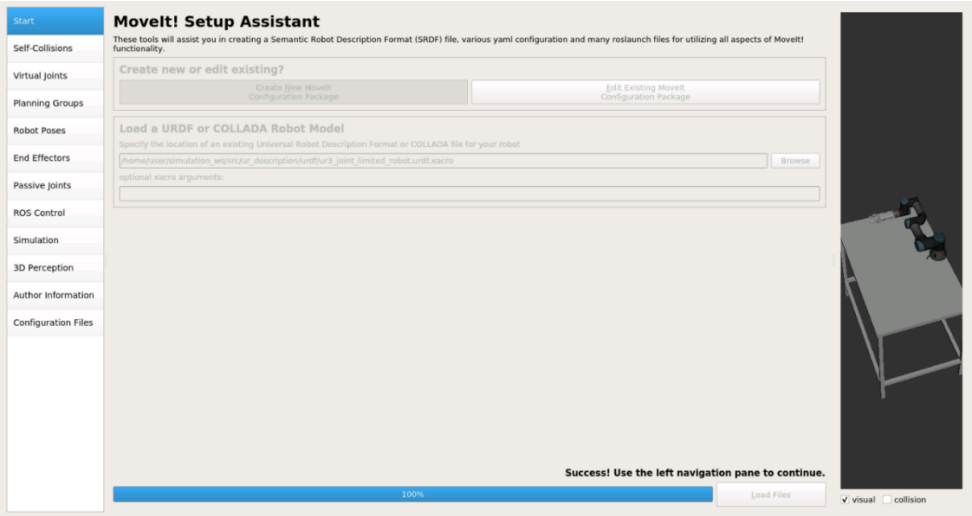
χρησιμοποιήθηκαν αυτά τα δυο σενάρια για την πραγματοποίηση μιας αξιολόγησης των αλγορίθμων RRT. Συγκεκριμένα χρησιμοποιώ τον αλγορίθμους RRT, RRTstar και RRTconnect. Σε κάθε σενάριο υπολογίζω βάση του προγράμματος **randomcodeevaluationrrt.py (** υπάρχουν μαζί με τα υπόλοιπα στο κεφάλαιο παραρτήματα) τους εξής παράγοντες των αλγορίθμων RRT. Μήκος διαδρομής, σχεδιασμός τροχιάς, εκτέλεση τροχιάς και θέση τελικού σημείου δράσης. Στην συνέχεια πραγματοποιείται η εκτέλεση της τροχιάς από το αρχικό σημείο του βραχίονα που ορίζεται στο πρόγραμμα **randomcodeevaluationrrt.py** έως το τελικό σημείο κάθε αλγορίθμου 3 φορές και στα δυο σενάριο(Σύνολο 9 φορές για κάθε σενάριο). Για την επιστροφή του ρομποτικού βραχίονα στο αρχικό σημείο χρησιμοποιήθηκε το πρόγραμμα **movement1.py**. Γίνεται ο υπολογισμός του μέσου όρου των

παραγόντων, των μέγιστων και ελάχιστων τιμών και τέλος βγαίνει ένα τελικό συμπέρασμα. Επιπρόσθετα εκτελείται μια εφαρμογή τύπου PickAndPlace.

5.2 Setup

Configure MoveIt package

Πραγματοποιείται η δημιουργία ενός πακέτου MoveIt configuration το οποίο επιτρέπει τον σχεδιασμό κίνησης του ρομποτικού βραχίονα στον χώρο. Μέσω του MoveIt package configuration προσδιορίζεται η κινηματική του ρομποτ, ο αλγόριθμος σχεδιασμού διαδρομής, η αποφυγή σύγκρουσης μεταξύ των τμημάτων του ρομποτ αλλά και με τα προσθετά μοντέλα που έχουν προστεθεί εντός του urdf αρχείου (στην προκείμενη περίπτωση είναι η αρπαγή του ρομποτ και το γραφείο). Παρακάτω παρατίθενται οι κατηγορίες του MoveIt configuration package σύμφωνα με τις ρυθμίσεις που πράχθηκαν.

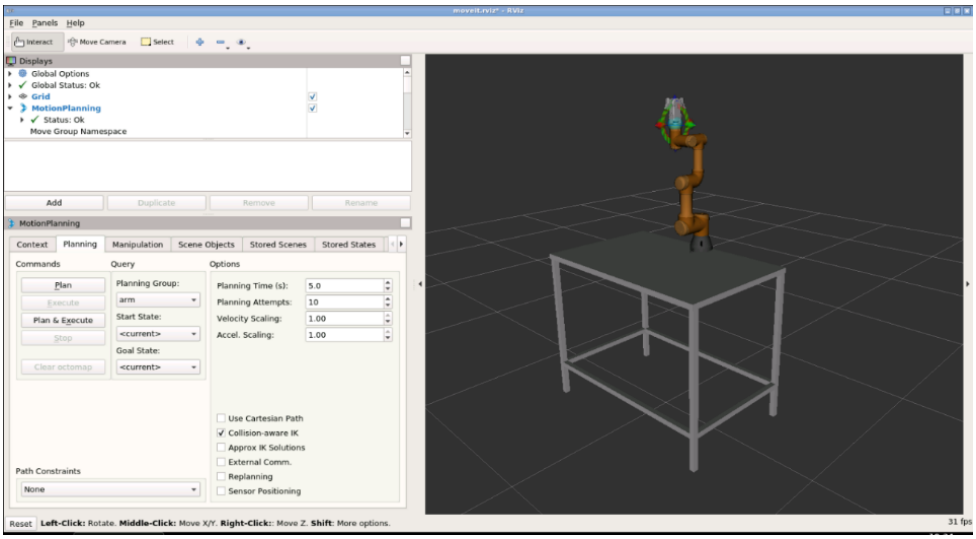


Εικόνα 31 Διαμόρφωση και δημιουργία του πακέτου MoveIt

- Self-Collisions: Σε αυτήν την κατηγορία πραγματοποιείται η δημιουργία ενός πίνακα σύγκρουσης του ρομποτ σε σχέση με τα μέρη του. Δηλαδή καθορίζονται τα ζεύγη των συνδέσμων για έλεγχο από συγκρούσεις κατά τον σχεδιασμό κίνησης με σκοπό την αποφυγή σύγκρουσης του ρομποτ με τα μέλη του. Επίσης καθορίζονται και οι σύνδεσμοι των μοντέλων που έχουν προστεθεί στο urdf αρχείο (πχ μοντέλο γραφείου).
- Planning Groups: Σε αυτήν την κατηγορία γίνεται η δημιουργία των ομάδων του ρομποτικού βραχίονα. Θέτω τις αρθρώσεις του βραχίονα (Το ur3 ρομποτ αποτελείτε από 6 αρθρώσεις περιστροφικού τύπου) ως την πρώτη ομάδα με όνομα “arm” και χρησιμοποιώ ως προκαθορισμένο αλγόριθμο σχεδιασμού κίνησης σε αυτήν την ομάδα τον αλγόριθμο rrt επίσης θέτω και την κινηματική επίλυση χρησιμοποιώντας το Plugin kdl kinematics solver το οποίο σκοπός του είναι να λύνει το αντίστροφο κινηματικό πρόβλημα. Η δεύτερη ομάδα περιγράφει την ρομποτική αρπαγή με όνομα “hand” , σε αυτήν την ομάδα θέτω μόνο μια άρθρωση και είναι η άρθρωση που πρισματικά κλείνει και ανοίγει την αρπαγή.

- Robot Poses: Σε αυτήν την κατηγορία θέτω τις προκαθορισμένες θέσεις του ρομποτικού βραχίονα. Δηλαδή τις θέσεις που θέλω να χρησιμοποιήσω στον σχεδιασμό κίνησης. Αυτό γίνεται επιλέγοντας τις ομάδες planning group και επιλέγοντας τις γωνίες των αρθρώσεων έτσι ώστε το ρομποτ να βρίσκεται σε μια συγκεκριμένη θέση και την αποθηκεύω, το ιδιο κανω και για την αρπαγή . Αυτές τις θέσεις στον χώρο θα τις χρησιμοποιήσω αργότερα για τον σχεδιασμό τροχιάς.
- End Effectors: Σε αυτήν την κατηγορία ορίζω το τελικό σημείο δράσης του ρομποτ ως την ομάδα “hand” δηλαδή την άρθρωση που έθεσα στο Planning group και ως σύνδεση με αυτήν την τελευταία άρθρωση του ρομποτ με όνομα wrist_3 και ονοματίζω το τελικό σημείο δράσης.

Τέλος δίνονται οι προσωπικές πληροφορίες δημιουργίας του πακέτου: όνομα, και email και πραγματοποιείται η δημιουργία του πακέτου. Για τον έλεγχο της ορθότητας εκτελείται η εντολή roslaunch <όνομα πακέτου> demo.launch έτσι ώστε να ανοίξει ο προσομοιωτής του RViz και μέσω του Motion planning panel εκτελείται είτε χειροκίνητα μια τροχιά στον χώρο είτε επιλέγοντας μέσω των προκαθορισμένων θέσεων που ρύθμισα μέσω της κατηγορίας robot poses κατά την δημιουργία του πακέτου MoveIt.



Εικόνα 32 RViz simulation του πακέτου MoveIt.

Αφού δημιουργηθεί το πακέτο γίνεται η σύνδεση του MoveIt με τον προσομοιωτή gazebo ώστε όταν εκτελούνται οι εντολές του MoveIt να εκτελούνται ταυτόχρονα και στον προσομοιωτή gazebo. Αυτό σημαίνει ότι μέσω του movegroup και ορισμένων plugin το gazebo επικοινωνεί με το MoveIt και τον προσομοιωτή RViz. Οπότε οποία κίνηση εκτελείται μέσω του RViz και των MoveIt εντολών πραγματοποιείτε εξίσου και στον προσομοιωτή gazebo.

Για την αλλαγή των αλγορίθμων σχεδιασμού διαδρομής χρησιμοποιείται η βιβλιοθήκη open motion planning library(ompl) που παρέχεται μέσω του λογισμικού MoveIt. Η αλλαγή αλγορίθμου πραγματοποιείται μέσω του αρχείου που βρίσκεται στο MoveIt πακέτο που δημιουργήθηκε στο κεφάλαιο setup με όνομα ompl_planning.yaml. Το αρχείο Ompl_planning.yaml παρέχει όλους τους βασικούς αλγορίθμους σχεδιασμού διαδρομής όπως Dijkstra, Astar, prm και πολλούς ακόμα. Στο αρχείο

ompl_planning.yaml πραγματοποιείται η αλλαγή του καθιερωμένου αλγόριθμου RRT σε RRTstar και RRTconnect. Οι συγκεκριμένοι αλγόριθμοι δεν χρειάζονται παραμετροποιήσεις.

```
arm:
  default_planner_config: RRT #RRTstar # RRTConnect
  planner_configs:
    - SBL
    - EST
    - LBKPIECE
    - BKPIECE
    - KPIECE
    - RRT
    - RRTConnect
    - RRTstar
    - TRRT
    - PRM
    - PRMstar
    - FMT
    - BFMT
    - PDST
    - STRIDE
    - BiTRRT
    - LBTRRT
    - BiEST
    - ProjEST
    - LazyPRM
    - LazyPRMstar
    - SPARS
    - SPARStwo
  projection_evaluator: joints(shoulder_pan_joint,shoulder_lift_joint)
  longest_valid_segment_fraction: 0.005
```

Εικόνα 33 αρχείο ompl_planning.yaml

5.3 Εκτέλεση Σεναρίων

Πρόγραμμα σχεδιασμού και εκτέλεσης κίνησης σεναρίου 1,2

Στο σενάριο 1 και 2 χρησιμοποιήθηκε ένα πρόγραμμα για την εκτέλεση της κίνησης του βραχίονα από ένα αρχικό σημείο σε ένα τελικό σημείο - στόχο. Για την επιστροφή του ρομποτικού βραχίονα στο αρχικό σημείο χρησιμοποιήθηκε ένα δεύτερο πρόγραμμα ως σημείο αναφοράς για την αποφυγή λανθασμένων αποτελεσμάτων.

Τμήμα (α) κώδικα **randomcodeevaluationrrt.py**

```
20.robot = moveit_commander.RobotCommander()
21.scene = moveit_commander.PlanningSceneInterface()
22.
23.group = moveit_commander.MoveGroupCommander("arm")
24.group2 = moveit_commander.MoveGroupCommander("hand")
25.display_trajectory_publisher = rospy.Publisher(
```

```
    '/move_group/display_planned_path',
    moveit_msgs.msg.DisplayTrajectory, queue_size=20)
```

Στο τμήμα (α) ορίζεται το ρομποτ μου μέσω της συνάρτησης RobotCommander για την περιγραφή του ρομποτ, της κινηματικής του και των group του ρομποτ που έχουν τεθεί στο MoveIt πακέτο. Ως group τίθεται το σύνολο των περιστροφικών αρθρώσεων του βραχίονα ενώ με την ονομασία group2 τίθεται η πρισματική άρθρωση της αρπάγης. Τέλος το topic `'/move_group/display_planned_path'` δίνει την δυνατότητα να αποτυπώνονται τα μηνύματα των θέσεων, ταχυτήτων και των επιταχύνσεων του συνόλου των αρθρώσεων του ρομποτ κάθε στιγμή κατά την διάρκεια της τροχιάς από το αρχικό έως το τελικό σημείο.

```
          effort: []
          time_from_start:
            secs: 2
            nsecs: 871916003
        -
          positions: [-0.3938224753139896, -0.5178681588923055, 0.449124
15958207086, -0.3038133270541521, 0.2310929200650227, 0.3592469270846945
]
          velocities: [0.667174671076977, 0.7526301247788051, -0.1540828
6099694207, 0.22998060059943237, -0.7123523006586236, -0.799288186020016
3]
          accelerations: [-0.7025391415383809, -0.7925242738973959, 0.16
225022559059635, -0.24217102465047027, 0.7501114707633321, 0.84165550700
78201]
          effort: []
          time_from_start:
            secs: 2
            nsecs: 926904422
```

Εικόνα 34 Topic /move_group/display_planned_path

Τμήμα (β) κώδικα **randomcodeevaluationrrt.py**

```
# thesi prin koytaki
group.get_current_joint_values()
group_variable_values[0] = -0.0915172139475
group_variable_values[1] = -0.176953153593
group_variable_values[2] = 0.379329819822
group_variable_values[3] = -0.199541716426
group_variable_values[4] = -0.0916237938249
group_variable_values[5] = -0.00286632744277
group.set_joint_value_target(group_variable_values)
plan1 = group.plan()
rospy.sleep(0.1)
```

```
group.go(wait=True)

# THESI telos
group_variable_values = group.get_current_joint_values()
group_variable_values[0] = 0.0118702462598    # joint 0
group_variable_values[1] = -3.14153438889
group_variable_values[2] = -0.0175243176233   # joint 0
group_variable_values[3] = -3.1415700392
group_variable_values[4] = 0.0237287330466   # joint 0
group_variable_values[5] = -0.0952336759661
group.set_joint_value_target(group_variable_values)
```

Στο τμήμα (β) του κώδικα ορίζεται ένα αρχικό και ένα τελικό σημείο του ρομποτικού βραχίονα. Στο τμήμα (α) του κώδικα ορίζεται ως group το planning group με την ονομασία " arm" δηλαδή οι μεταβλητές των αρθρώσεων του ρομπότ. Ο ρομποτικός βραχίονας ur3 αποτελείται από 6 αρθρώσεις [0-5]. Οπότε τοποθετούνται οι τιμές κάθε άρθρωσης σε τέτοια διάταξη ώστε να μου δώσουν μια αρχική και τελική διαμόρφωση του ρομπότ. Η εύρεση των τιμών των αρθρώσεων πραγματοποιήθηκε προσεγγιστικά βάζοντας διάφορες τιμές και βρίσκοντας τις κατάλληλες γωνίες ώστε καθοριστούν οι θέσεις. Η εντολή group go εκτελεί την τροχιά την αρχικής θέσης

Τμήμα (γ) κώδικα **randomcodeevaluationrrt.py**

```
# Plan the trajectory from the start to goal configuration
plan_start_time = time.time()
plan1 = group.plan()
plan_time = time.time() - plan_start_time

# Execute the planned trajectory
exe_start_time = time.time()
group.go(wait=True)
exe_time = time.time() - exe_start_time

# Calculate the path length
path_len = 0
previous_point = None
for point in plan1.joint_trajectory.points:
    if previous_point is not None:
        path_len += np.linalg.norm(np.array(point.positions) -
np.array(previous_point.positions))
    previous_point = point

```

```
70. # Get the end-effector position
71. end_effector_pose = group.get_current_pose().pose
```

Στο Τμήμα του κώδικα (γ) ξεκινάει ένα χρονόμετρο με την εντολή `plan_start_time = time.time()` Για την μέτρηση του σχεδιασμού κίνησης , και όχι της εκτέλεσης. Στην συνέχεια μέσω της εντολής `group.go(wait=True)` εκτελείται η τροχιά από το αρχικό έως το τελικό σημείο και στην συνέχεια αφαιρείται από τον τωρινό χρόνο που τέλειωσε η σχεδίαση, τον χρόνο που ξεκίνησε η σχεδίαση κίνησης και βρίσκεται ο χρόνος σχεδίασης της τροχιάς από το αρχικό έως το τελικό σημείο (**`plan_time = time.time() - plan_start_time)`**. Η ιδιά διαδικασία πραγματοποιείται και με τον χρονο εκτελεσης ξαναενεργοποιοντας ένα χρονόμετρο.

Για να υπολογιστεί το μήκος της διαδρομής χρησιμοποιώ τα σημεία(points). Τίθεται `previous_point = None` που σημαίνει ότι το προηγούμενο σημείο ισούται με μηδέν οπότε το αρχικοποίω. Ο βρόγχος ελέγχει αν το ρομποτ έχει περάσει από προηγούμενο σημείο και αν δεν υπάρχει τότε δεν κάνει τίποτα, αν υπάρχει προηγούμενο σημείο τότε υπολογίζει την απόσταση του τωρινού από το προηγούμενο σημείο μέσω αυτής της γραμμής κώδικα `np.linalg.norm(np.array(point.positions) - np.array(previous_point.positions)` οπού υπολογίζει την ευκλείδεια απόσταση από το τωρινό έως το προηγούμενο σημείο.

Το end_effector_pose λαμβάνει μέσω του get_current_pose() την θέση και τον προσανατολισμό του τελικού σημείου δράσης.

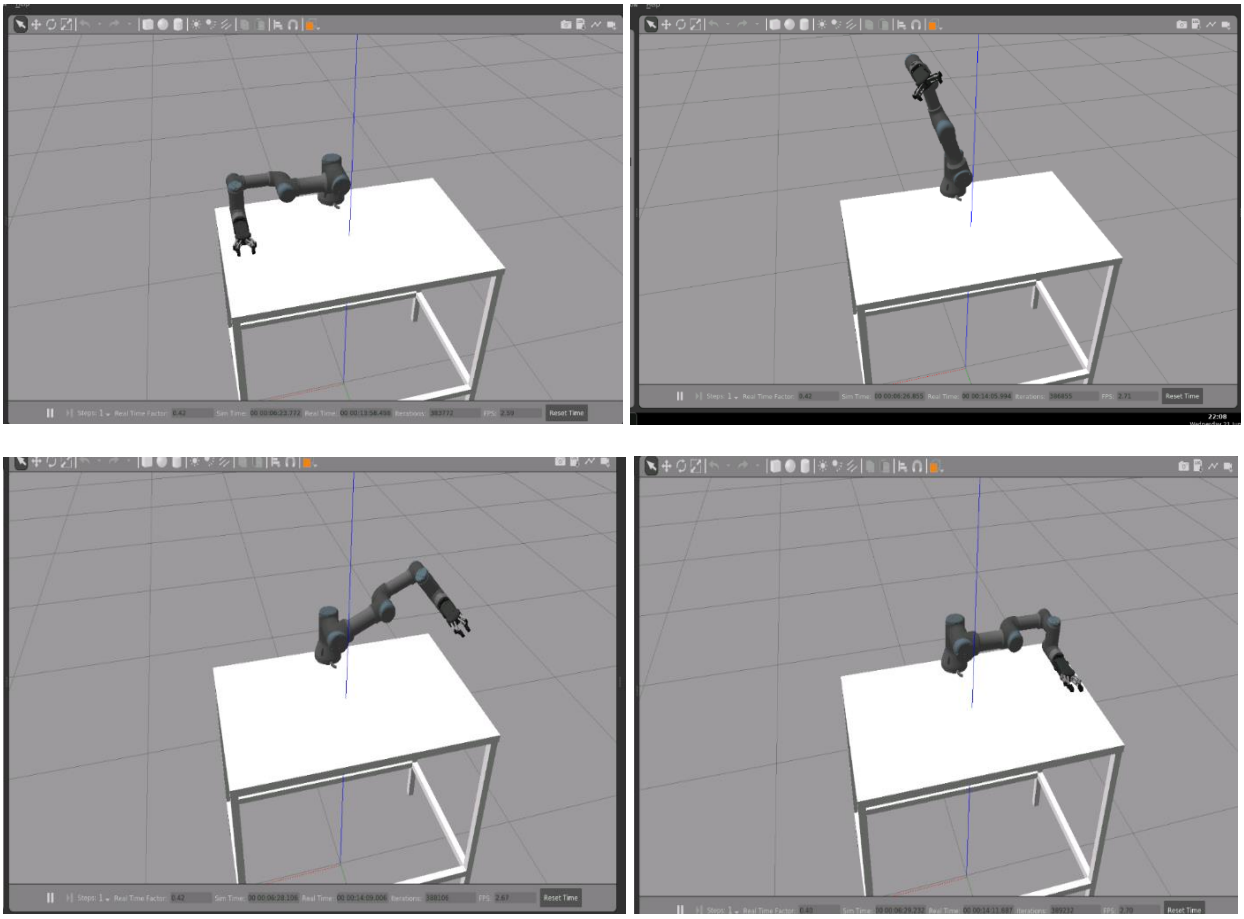
Τμήμα (δ) κώδικα **randomcodeevaluationrrt.py**

```
73. # Output
74. print("Planning time: {:.4f} s".format(plan_time))
75. print("Execution time: {:.4f} s".format(exe_time))
76. print("Path length: {:.4f} m".format(path_len))
77. print("End-effector position: ({:.4f}, {:.4f},
    {:.4f})".format(end_effector_pose.position.x,
    end_effector_pose.position.y, end_effector_pose.position.z))
78. print(group.get_planner_id())
```

Στο Τμήμα (δ) του κώδικα γίνεται η εκτύπωσή των θέσεων του τελικού σημείου δράσεως καθώς και του χρόνου σχεδίασης, εκτελεσης και του μήκους του μονοπατιού.

5.3.1 Σενάριο 1: Εκτέλεση Τροχιάς ρομποτικού βραχίονα χωρίς εμπόδιο

Στο σενάριο 1 εκτελείται η κίνηση του ρομποτικού βραχίονα χωρίς την ύπαρξη εμποδίων. Παρακάτω βλέπουμε την εκτέλεση της τροχιάς του προσομοιωτή gazebo. Η συγκεκριμένη εκτέλεση πραγματοποιείται χρησιμοποιώντας τον αλγόριθμο RRT.



Εικόνα 35 Εφαρμογή Σεναρίου 1

5.3.1.1 Αποτελεσματα Σεναριου 1

Μετρήσεις RRT

Planning time: (0.2768 , 0.0221 , 0.1976) | Μέσος όρος = 0,1655 | max = 0.2768 ,min=0.0221

Execution time: (7.0971 , 7.5607 , 7.2329) | Μέσος όρος = 7.2969 | max = 7.5607 ,min= 7.0971

Path length : (4.1991, 4.1993 , 4.1993) | Μέσος όρος = 4.19923 | max = 4.1993 ,min=4.1991

Μετρήσεις RRTstar

Planning time: (5.0212 , 5.0884 , 5.0883) | Μέσος όρος = 5.066 | max =5.0884 ,min=5.0212

Execution time: (12.0849 , 12.3113 , 12.1884) | Μέσος όρος = 12.1948 |max =12.3113 ,min=12.0849

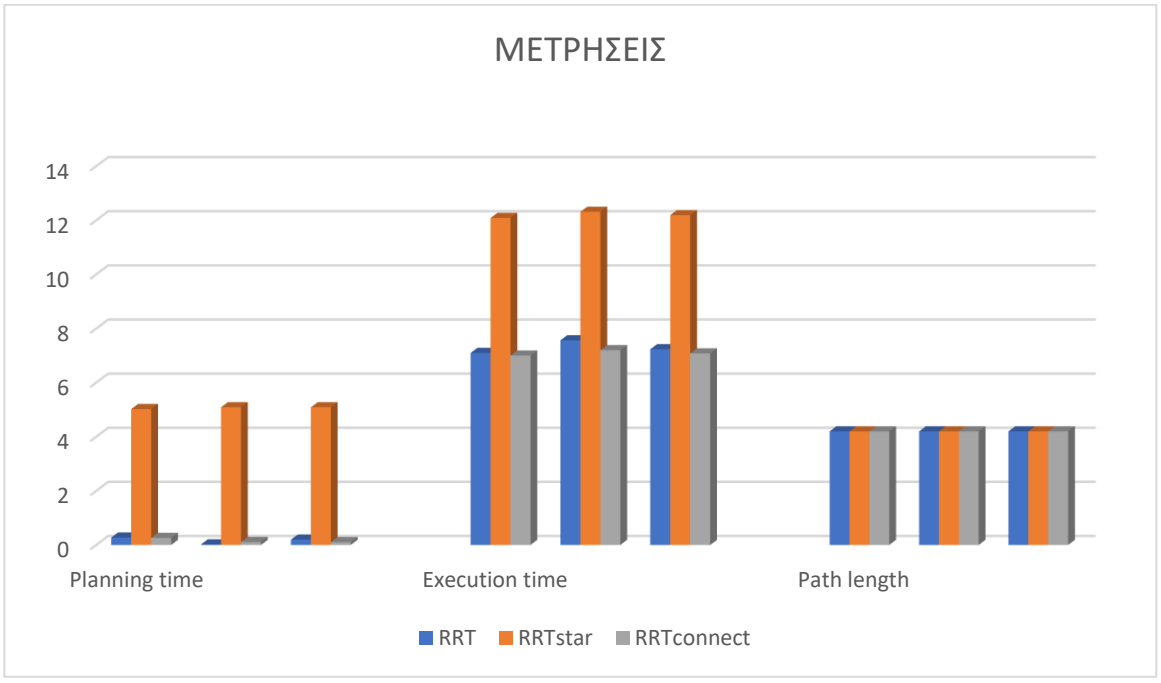
Path length : (4.1993, 4.1992 , 4.1992) | Μέσος όρος = 4.19923 | max =4.1993 ,min=4.1992

Μετρήσεις RRTconnect

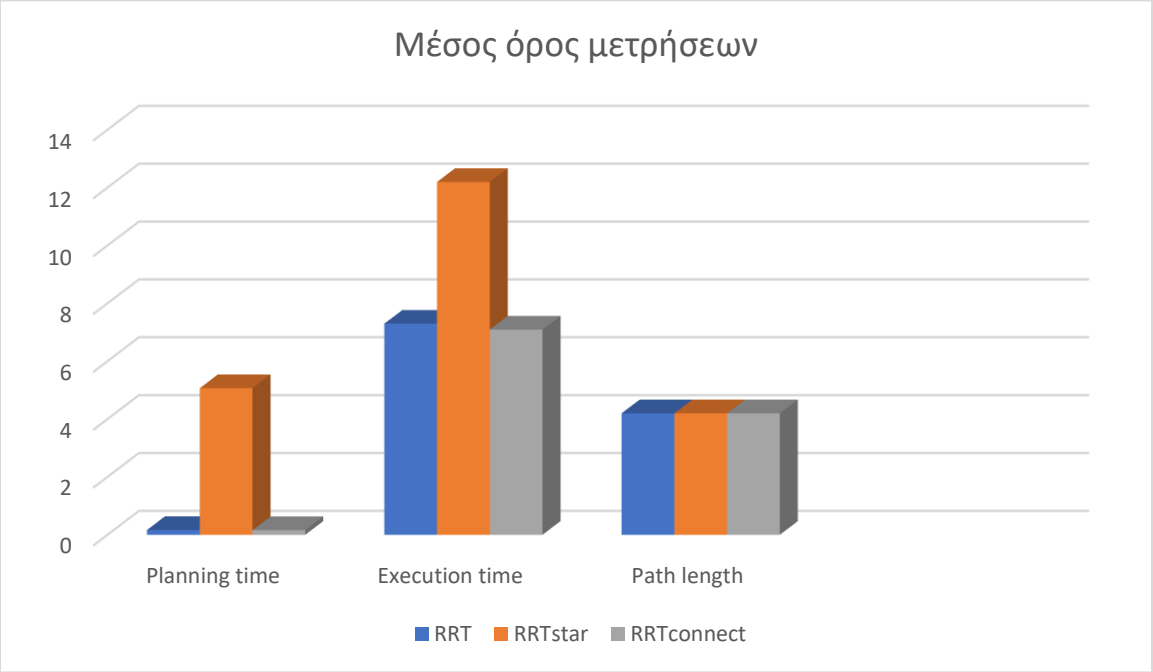
Planning time: : (0.2608 , 0.1127 , 0.1084) | Μέσος όρος = 0.1606 | max =0.2608 ,min=0.1084

Execution time: (7.0017, 7.1975 , 7.0779) | Μέσος όρος = 7.0923 | max =7.1975 ,min=7.0017

Path length : (4.1992 , 4.1992 , 4.1994) | Μέσος όρος = 4.19926 | max =4.1994 ,min=4.1992

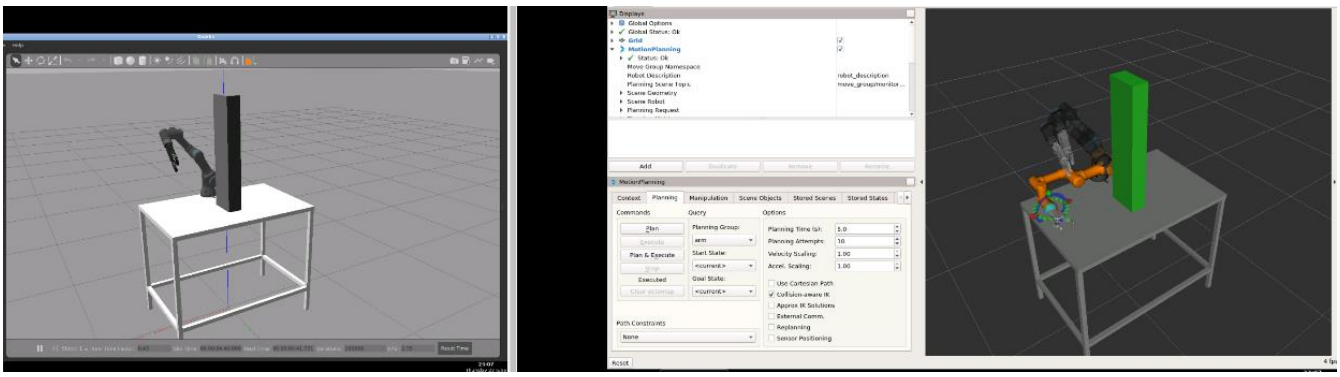


SUM ΑΛΓΟΡΙΘΜΩΝ ΣΧΕΔΙΑΣΜΟΥ ΚΙΝΗΣΗΣ	**PLANNING TIME**	**EXECUTION TIME**	**PATH LENGTH**	**END EFFECTOR POSITION**
RRT	0.1655 s	7.2969 s	4.19923 m	(-0.4557, -0.1112, 0.9429)
RRT STAR	5.066 s	12.1948 s	4.19923 m	(-0.4557, -0.1111, 0.9429)
RRT CONNECT	0.1606s	7.0923s	4.19926 m	(-0.4557, -0.1111, 0.9429)



5.3.2 Σενάριο 2: Εκτέλεση τροχιάς με εμπόδιο

Για την υλοποίηση του σεναρίου 2 δημιουργήθηκε ένας κώδικας ο οποίος προσθέτει στο gazebo και στο MoveIt καθώς και στο RViz ένα εμπόδιο. Το εμπόδιο στο gazebo είναι καθαρά θέμα αισθητικής αφού το gazebo δεν το αναγνωρίζει ως εμπόδιο αλλά το εμπόδιο αναγνωρίζεται στο gazebo μέσω της ένταξης του στο MoveIt. Όπως βλέπουμε στην παρακάτω εικόνα η ένταξη του εμποδίου πραγματοποιείται και στις δυο προσομοιώσεις αλλά αναγνωρίζεται μόνο μέσω του move_group κόμβου.



Εικόνα 36 Εκτέλεση τροχιάς και οπτικοποίηση σε προσομοιωτή RViz και gazebo προσομοιωτή.

Τμήμα (α) κώδικα **objects.py**

```
collision_object = CollisionObject()
collision_object.header.frame_id = 'world'  # Set the frame ID of the object

# Set the ID of the collision object (should be unique)
collision_object.id = 'box'

# Set the dimensions of the box
box_size = [0.1, 0.2, 0.8]

# Set the pose of the box
box_pose = PoseStamped()
box_pose.header.frame_id = 'world'
box_pose.pose.position = Point(x=0.0, y=0.04, z=1.28)

# Set the orientation of the box
box_orientation = Quaternion(x=0.0, y=0.0, z=0.0, w=1.0)
box_pose.pose.orientation = box_orientation

# Assign the pose to the collision object
```

```
35. collision_object.primitives.append(SolidPrimitive(type=SolidPrimitive.BOX
    , dimensions=box_size))
36. collision_object.primitive_poses.append(box_pose.pose)
37.
38. # Publish the collision object
39. co_pub = rospy.Publisher('/collision_object', CollisionObject,
    queue_size=1)
```

Ο παραπάνω κώδικας δημιουργεί ένα αντικείμενο με διαστάσεις x=0.1, y=0.2, z=0.8, και θέση x=0.0, y=0.04, z=1.28. Τέλος ορίζει το αντικείμενο ως ένα αντικείμενο σύγκρουσης.

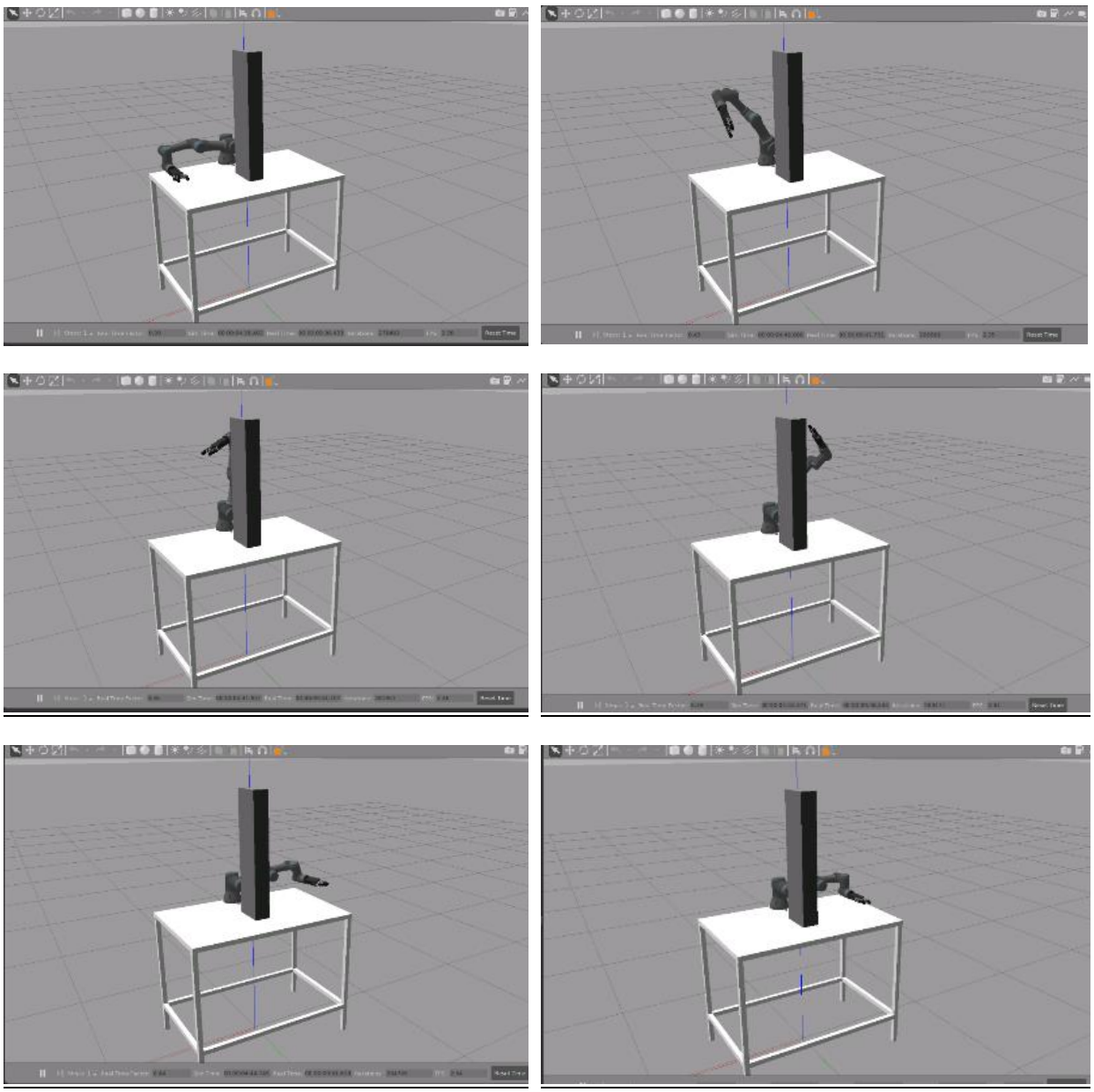
Τμήμα (β) κώδικα **objects.py**

```
48. # Create a service proxy for the spawn_sdf_model service
49. rospy.wait_for_service('/gazebo/spawn_sdf_model')
50. spawn_sdf_model = rospy.ServiceProxy('/gazebo/spawn_sdf_model',
    SpawnModel)
51.
52. # Open the model SDF file and read the XML content
53. bin_file = open('/home/user/model_editor_models/project/model.sdf')
54. bin_xml = bin_file.read()
55.
56. # Set the pose of the object
57. bin_pose = Pose()
58. bin_pose.position.x = 0
59. bin_pose.position.y = 0.04
60. bin_pose.position.z = 1.28
61.
62. # Set the name of the model and the frame ID
63. bin_name = 'bin'
64. reference_frame = 'world'
65.
66. # Call the spawn_sdf_model service to spawn the object
67. spawn_sdf_model(bin_name, bin_xml, '', bin_pose, reference_frame)
```

Το ιδιο συμβαίνει και για το gazebo simulation, δημιουργώντας για αρχή χειροκίνητα ένα αντικείμενο στο προσομοιωτή του gazebo και βάζοντας του τις διαστάσεις σχήματος x = 0.8 , y=0.2, z=0.1. Στην συνέχεια το αποθηκεύουμε ως model sdf '/home/user/model_editor_models/project/model.sdf'.Μέσα

στον κώδικα βάζω την θέση του αντικειμένου στον προσομοιωτή του gazebo καθώς και εκτελώ την εντολή spawn_sdf_model() για να εμφανιστεί στο περιβάλλον.

Παρακάτω παρατηρούμε την εκτέλεση της τροχιάς στο gazebo προσομοιωτή.



Εικόνα 37 Εφαρμογή Σεναρίου 2

5.3.2.1 Αποτελέσματα Σενάριού 2

Μετρήσεις RRT

Planning time: (0.5884 , 1.2086 , 3.1005) | Μέσος όρος = 1.6325 | max = 3.1005 , min = 0.5884

Execution time: (15.9290, 16.9963 , 17.3136) | Μέσος όρος = 16.7463 | max =17.3136 , min= 15.9290

Path length : : (6.0489, 6.0402 , 7.2763) | Μέσος όρος = 6.4551 | max = 7.2763 , min = 6.0402

Μετρήσεις RRTstar

Planning time: (5.1036, 5.1186 , 5.1073) | Μέσος όρος = 5.1098 | max = 5.1186, min = 5.1036

Execution time: (20.6858 , 19.1017 , 18.0029) | Μέσος όρος = 19.2634 | max = 20.6858 ,min= 18.0029

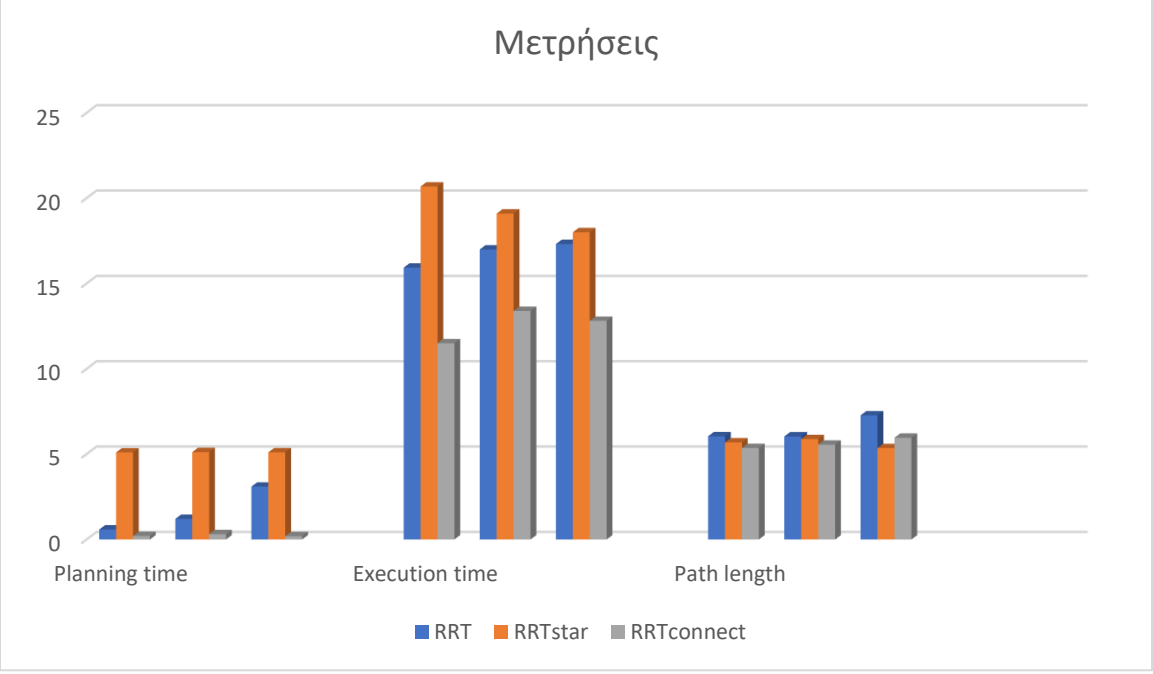
Path length : (5.6896 , 5.8824 , 5.3553) | Μέσος όρος = 5.6424 | max = 5.8824 ,min= 5.3553

Μετρήσεις RRTconnect

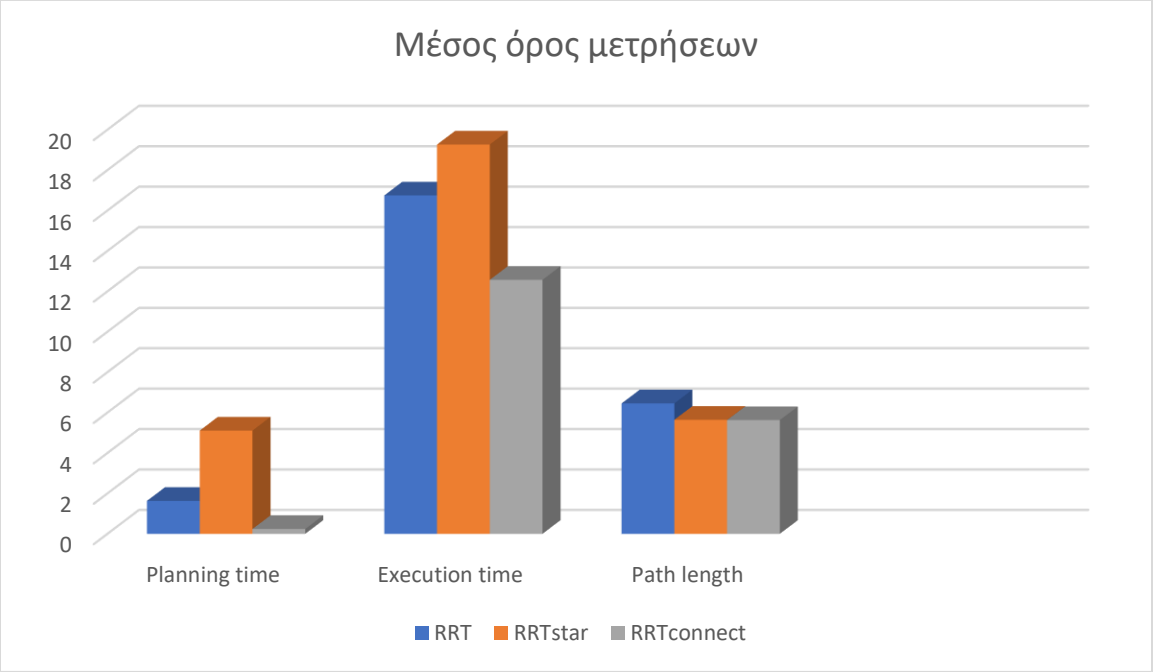
Planning time: (0.2079 , 0.3127 , 0.1940) | Μέσος όρος = 0.2382 | max = 0.3127 ,min=0.1940

Execution time: (11.5005 , 13.3949 , 12.8129) | Μέσος όρος = 12.5694 | max = 13.3949 ,min=11.5005

Path length : (5.3634, 5.5574 , 5.9623) | Μέσος όρος = 5.6277 | max = 5.9623,min= 5.3634

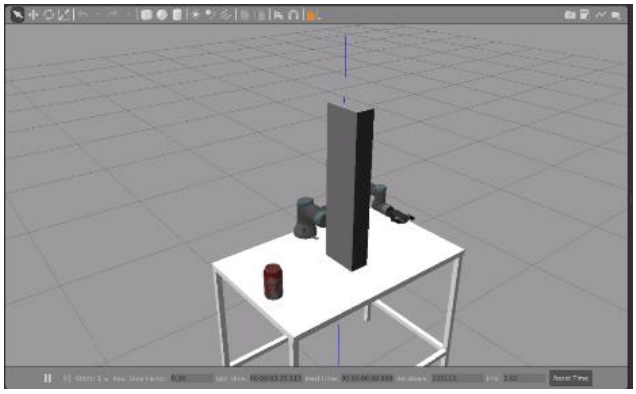
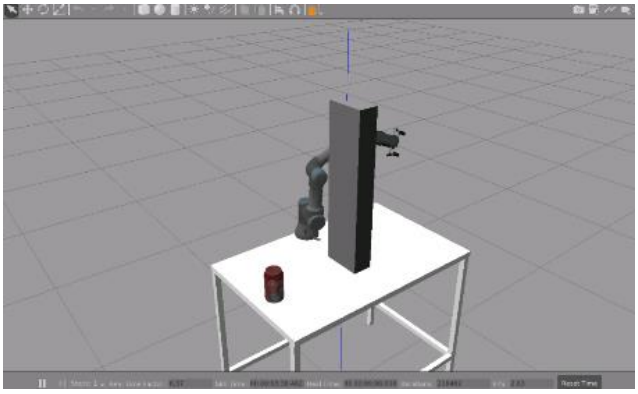
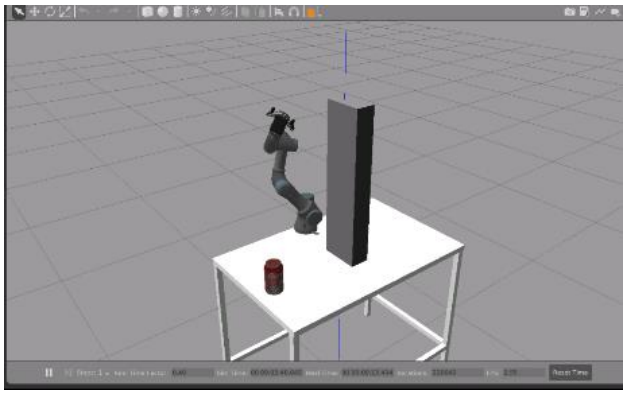
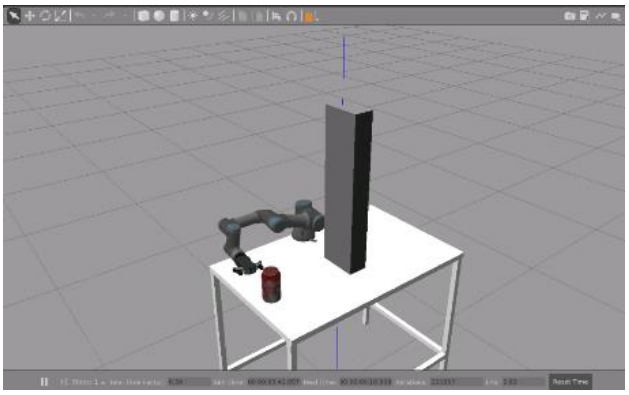
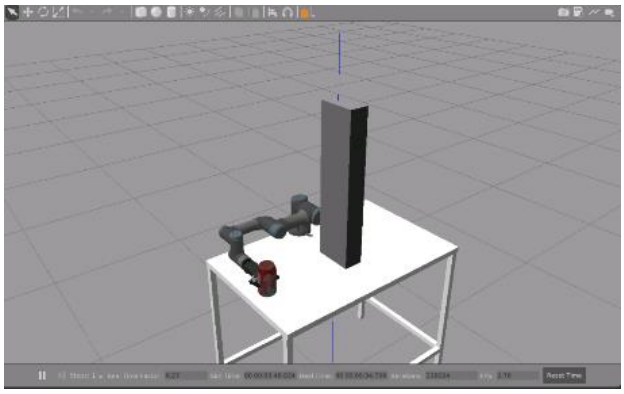
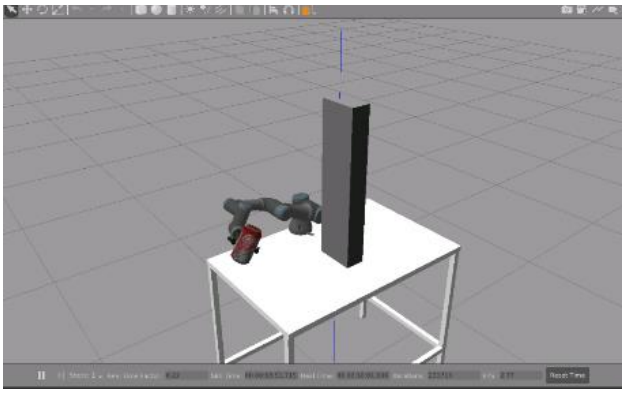


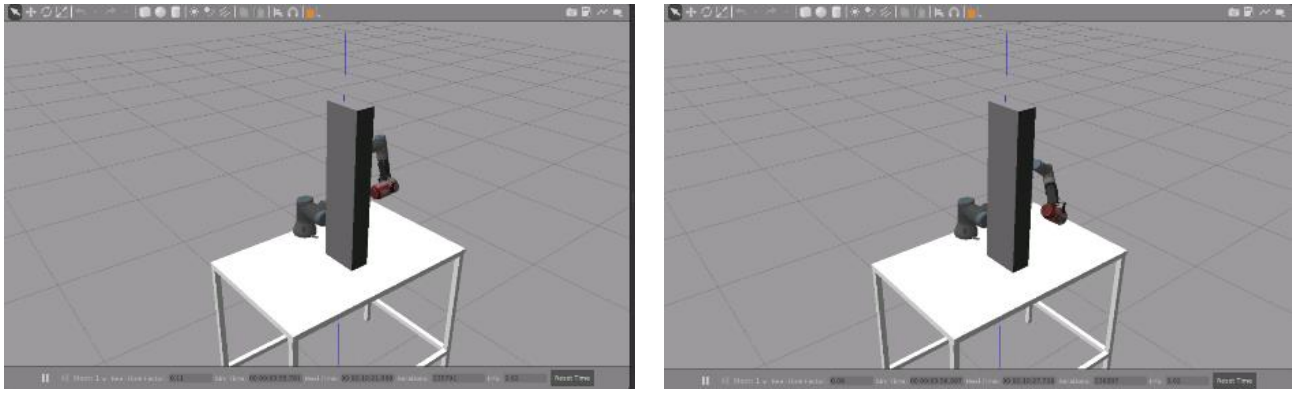
SUM ΑΛΓΟΡΙΘΜΩΝ ΣΧΕΔΙΑΣΜΟΥ ΚΙΝΗΣΗΣ	**PLANNING TIME**	**EXECUTION TIME**	**PATH LENGTH**	**END EFFECTOR POSITION**
RRT	1.6325 s	16.7463 s	6.4551 m	(-0.4557, -0.1111, 0.9429)
RRT STAR	5.1098 s	19.2634 s	5.6424 m	(-0.4557, -0.1112, 0.9429)
RRT CONNECT	0.2382 s	12.5694 s	5.6277 m	(-0.4557, -0.1112, 0.9429)



Εφαρμογη pick and place

Για την εφαρμογή pick and place χρησιμοποιησα δημιουργήθηκε ο κωδικας **pick_and_place.py** οπου εισάγει ένα αντικείμενο στο περιβάλλον gazebo το οποιο δεν αναγνωρίζεται από το rviz. Το αρχικο και τελικο σημειο είναι ιδιο. Ο ρομποτικος βραχιονας ξεκιναει ανοιγοντας τη αρπαγη εκτελη κινηση ως προς την θεση zero και στην συνεχεια προσεγγιζει το κουτακι. Όταν το κουτακι βρεθει εντος της αρπαγης τοτε η αρπαγη κλεινει και ο βραχιονας εκτελει σχεδιασμο τροχιας επιστρεφοντας στο αρχικο σημειο ετσι ώστε να αφησει το κουτακι.



Εικόνα 38 Εφαρμογή Pick and place

5.4 Έλεγχος

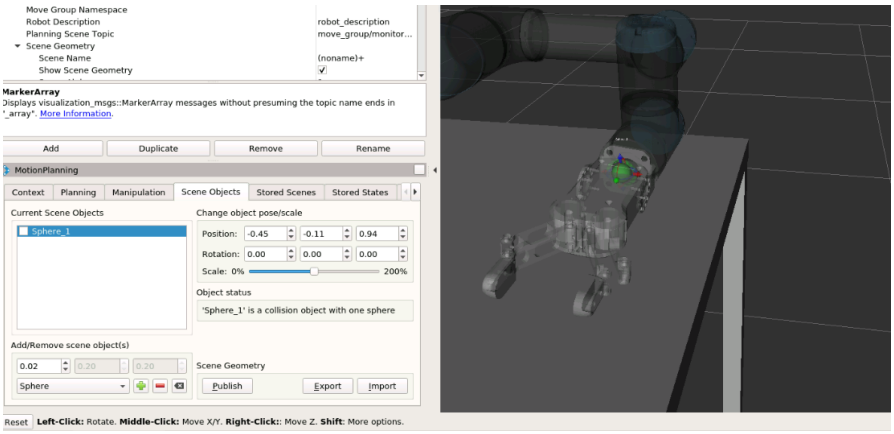
Έλεγχος θέσης τελικού σημείου δράσεως

Για τον έλεγχο του αν πράγματι ο κώδικας βρίσκει την θέση του τελικού σημείου δράσης του ρομποτ. Πάρθηκαν τα αποτελέσματα της εκτέλεσης του κώδικα από την αρχική έως μια τελική θέση.

```
ripts$ python randomcodeevaluationrrt.
py
[ INFO] [1687623653.778063704, 40.5920
00000]: Loading robot model 'ur3'...
[ INFO] [1687623653.786905283, 40.6020
00000]: No root/virtual joint specifie
d in SRDF. Assuming fixed joint
[ INFO] [1687623657.003392177, 42.2330
00000]: Ready to take commands for pla
nning group arm.
[ INFO] [1687623658.578001460, 43.0730
00000]: Ready to take commands for pla
nning group hand.
Planning time: 0.1958 s
Execution time: 7.0752 s
Path length: 4.1994 m
End-effector position: (-0.4557, -0.11
11, 0.9429)
```

Εικόνα 39 Αποτελέσματα εκτελεσης προγράμματος

Και δημιουργείται στο RViz ένα αντικείμενο σε σχήμα σφαίρας και ακτίνα 0.02 στις παραπάνω συντεταγμένες. Το RViz αφήνει να χρησιμοποιηθούν έως και 2 ψηφιά μετά την υποδιαστολή οπότε γίνεται η τοποθέτηση των συντεταγμένων του αντικειμένου σφαίρας χ=-0.45 y=-0.11 z=0.94. Και παρατηρώ ότι βρίσκεται στην τελική θέση της τροχιάς και συγκεκριμένα στο τελικό σημείο δράσης του ρομποτικού βραχίονα.



Εικόνα 40 Έλεγχος θέσης τελικού σημείου δράσης μέσω του προσομοιωτή RViz

Εκτέλεση αλγορίθμου

Με την εναλλαγή των αλγορίθμων, ελέγχουμε στο Movegoup node αν εκτελούνται σωστά. Στην παρακάτω εικόνα έχω χρησιμοποιήσει τον αλγόριθμο rrt star για να δημιουργηθεί μια τροχιά. Παρατηρήθηκε ότι τα δέντρα του αλγορίθμου δημιουργούν νέου κόμβους.

```
123.983000000]: RRTstar: proble
m definition is not set, deferr
ing setup completion...
[ INFO] [1687652748.725374861,
123.984000000]: RRTstar: Starte
d planning with 1 states. Seeki
ng a solution better than 0.000
00.
[ INFO] [1687652748.729459167,
123.987000000]: RRTstar: Initia
l k-nearest value of 310
[ INFO] [1687652748.933935922,
124.087000000]: RRTstar: Found
```

Εικόνα 41 Αποτελέσματα move group node

Σε περίπτωση μη λειτουργίας του αλγορίθμου είτε διότι δεν βρίσκει κάποια λύση είτε διότι δεν επικοινωνούν σωστά τα nodes. Τότε είτε δεν δίνει αποτέλεσμα και σταματάει ο κώδικας είτε δίνει ως αποτέλεσμα λανθασμένες τιμές όπως την τελική θέση του τελικού σημείου δράσης που πρέπει να είναι ιδιά.

5.5 Αποτελέσματα αλγορίθμων RRT

Βάση των μετρήσεων που λάβαμε από τις παραμέτρους του χρονικού σχεδιασμού , εκτελεσης αλλά και μήκους διαδρομής των αλγορίθμων σχεδιασμού διαδρομής (RRT, RRTstar, RRTconnect) στα σενάρια 1 και 2 παρατηρούνται τα εξής:

- Αλγόριθμος RRT: Οι μετρήσεις των παραμέτρων του βασικού αλγορίθμου RRT δεν είχε ιδιαίτερα μεγάλες αποκλίσεις στην εκτέλεση του στο σενάριο 1 (Σενάριο χωρίς τοποθέτηση εμποδίου). Ενώ στο σενάριο 2 (Σενάριο με τοποθέτηση εμποδίου) παρατηρήθηκαν τεράστιες αποκλίσεις ιδιαίτερα στις μετρήσεις σχεδιασμού κίνησης και μήκος διαδρομής. Συγκεκριμένα η απόκλιση της Μέγιστης με την ελάχιστή μέτρηση στον σχεδιασμό κίνησης ισούται με (3.1005-0.5884) 2,5121 δευτερόλεπτα ενώ η απόκλιση του μήκους διαδρομής της μέγιστης σε σχέση με την ελάχιστη είχαν απόκλιση (7.2763 - 6.0402) ίση με 1,2361 μέτρα.

- Αλγόριθμος RRTstar: Ο αλγόριθμος RRTstar απεδείχθη ο ποιο σταθερός αλγόριθμος βάση των μετρήσεων που πάρθηκαν και στα δυο σενάρια. Με την μονή σημαντική απόκλιση να παρατηρείται στο σενάριο 2 στην μέτρηση του χρόνου εκτελεσης και ισούται με την διαφορά της μέγιστης με την ελάχιστη μέτρηση (20.6858 - 18.0029) = 2,6829 δευτερόλεπτα

- Αλγόριθμος RRTconnect: Ο αλγόριθμος RRTconnect δεν είχε μεγάλες αποκλίσεις στο σενάριο 1. Ενώ στο σενάριο 2 παρατηρήθηκαν αποκλίσεις στις μετρήσεις του σχεδιασμού και εκτελεσης κίνησης. Ενώ μικρές αποκλίσεις του μήκους διαδρομής και συγκεκριμένα η απόκλιση της μέγιστης μέτρησης σε σχέση με την ελάχιστη ισούται με (5.9623-5.3634) 0,5989 μέτρα.

Σύμφωνα με το μέσο ορό των τριών μετρήσεων που υπολογίστηκε για κάθε αλγόριθμο σχεδιασμού διαδρομής σε κάθε σενάριο παρατηρήθηκε ότι:

- Sum αλγορίθμου RRT: Στο σενάριο 2, ο αλγόριθμος RRT έχει το μεγαλύτερο μήκος διαδρομής.

- Sum αλγορίθμου RRTstar: Στο σενάριο 1 και 2 ο αλγόριθμος RRTstar έχει το μεγαλύτερο χρονο εκτελεσης και σχεδίασης.

- Sum αλγορίθμου RRTconnect: Στο σενάριο 1 και 2 ο αλγόριθμος RRTconnect έχει τον μικρότερο χρονο σχεδίασης και εκτελεσης ενώ στο σενάριο 2 έχει το μικρότερο μήκος διαδρομής.

Συμπεράσματα

Στην παρούσα εργασία πραγματοποιήθηκε ανάλυση των ρομποτικών συστημάτων καθώς και των αλγορίθμων σχεδιασμού διαδρομής για την αποφυγή εμποδίων και την αναζήτηση της πλησιέστερης διαδρομής μεταξύ μιας αρχικής και τελικής διαμόρφωσης. Συγκεκριμένα αναλυθήκαν οι αλγόριθμοι RRT, RRTstar και RRTconnect ως προς τον χρονο σχεδιασμού τροχιάς, χρονο εκτελεσης τροχιάς και μήκος διαδρομής. Ο έλεγχος της τροχιάς πραγματοποιήθηκε μέσω του υπολογισμού της θέσης του τελικού σημείου δράσεως στο τελικό σημείο – στόχο.

Η υλοποίηση της εφαρμογής έγινε μέσω του λογισμικού Ros με την βοήθεια του εργαλείου MoveIt και των προσομοιωτών RViz και gazebo.

Σύμφωνα με τις μετρήσεις αλλά και με τον μέσο όρο των μετρήσεων παρατηρήθηκε ότι ο αλγόριθμος RRT όταν παρουσιάζεται εμπόδιο μέσα στον χώρο εργασίας δεν είναι σταθερός ως προς τις μετρήσεις του και μπορεί να δώσει μεγάλες μετρήσεις μήκους διαδρομής. Ο RRTstar παρουσίασε μεγάλη σταθερότητα στις μετρήσεις του με το μικρότερο μέσο όρο μήκους διαδρομής στο σενάριο 2 αλλά είχε μεγάλο χρονο εκτελεσης και σχεδιασμού. Ενώ ο RRTconnect Έχει μικρό χρονο σχεδίασης και εκτελεσης αλλά ούτε εκείνος ήταν σταθερός ως προς τις μετρήσεις του.

Συμπερασματικά ως προς το σενάριο 2 που παρεμβαλλόταν εμπόδιο, η επιλογή του RRTstar είναι η καλύτερη αν ο παράγοντας είναι η επαναληψιμότητα και το μήκος διαδρομής ενώ η επιλογή του αλγόριθμου RRTconnect είναι ιδανικότερη αν ο παράγοντας είναι ο χρόνος σχεδιασμού διαδρομής και εκτελεσης.

Η καινοτομία της διπλωματικής έγκειται στην αξιολόγηση και σύγκριση της απόδοσης τριών διαφορετικών αλγορίθμων σχεδιασμού διαδρομής (RRT, RRTSTAR, RRTCONNECT) χρησιμοποιώντας το MoveIt και gazebo προσομοιωτή και προγραμματίζοντας χρησιμοποιώντας τα εργαλεία του MoveIt για έναν αρθρωτό ρομποτικό βραχίονα. Με την αξιολόγηση του χρόνου σχεδίασης, του χρόνου εκτελεσης και του μήκους διαδρομής, αποκτήθηκαν πληροφορίες για τα δυνατά και αδύνατα σημεία κάθε αλγορίθμου προσδιορίστηκε ποιος αποδίδει καλυτέρα όσον αφορά την αποτελεσματικότητα και την ακρίβεια σε περιβάλλοντα με η χωρίς παρεμβαλόμενα εμπόδια. Αυτή η συγκριτική ανάλυση μπορεί να βοηθήσει στην επιλογή αλγορίθμων σχεδιασμού κίνησης για μελλοντικές εφαρμογές.

<u>Προβλήματα εφαρμογής</u>

Για τα στατικά εμπόδια η παραπάνω μεθοδολογία εκτέλεσης αποφυγής εμποδίου λειτουργεί αλλά σε δυναμικά περιβάλλοντα δεν λειτουργεί. Η αποφυγή εμποδίου γίνεται μέσω του MoveIt δημιουργώντας και δημοσιεύοντας το εμπόδιο. Αν τεθεί το εμπόδιο στο MoveIt αλλά όχι στο gazebo, ενώ το εμπόδιο δεν θα εμφανιστεί στο gazebo η εκτέλεση τροχιάς με εμπόδιο θα συμβεί κανονικά. Ένας τρόπος λύσης αυτού του συμβάντος είναι να τοποθετηθεί στο gazebo κάποιος αισθητήρας κάμερας που θα αναγνωρίζει τα εμπόδια στο gazebo και θα τα εμφανίζει στο MoveIt. Έτσι μπορεί να εκτελεστεί μια δυναμική αποφυγή εμποδίου. Δηλαδή με την εμφάνιση κάποιου εμπόδιο στο gazebo η αρχική εκτέλεση της τροχιάς θα αλλάζει και η αποφυγή εμποδίου θα προσαρμόζεται βάση των νέων δεδομένων.

Βιβλιογραφία

[1] «wikipedia,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/R.U.R..

[2] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Three_Laws_of_Robotics.

[3] [Ηλεκτρονικό]. Available: https://cyberneticzoo.com/robots/1937-the-robot-gargantua-bill-griffith-p-taylor-australiancanadian/.

[4] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/George_Devol.

[5] [Ηλεκτρονικό]. Available: https://robotsguide.com/robots/unimate.

[6] [Ηλεκτρονικό]. Available: https://robotics.kawasaki.com/en1/anniversary/history/history_01.html.

[7] [Ηλεκτρονικό]. Available: https://www.scirp.org/journal/paperinformation.aspx?paperid=90517.

[8] [Ηλεκτρονικό]. Available: https://www.bridgemanimages.com/en/noartistknown/factory-automation-c-1980-a-unimate-puma-programmable-universal-machine-for-assembly-industrial/nomedium/asset/3400300.

[9] [Ηλεκτρονικό]. Available: https://sidhuera.com/scara-robots-the-industrial-robot-history-of-robots/.

[10] [Ηλεκτρονικό]. Available: https://www.packworld.com/machinery/robotics/article/21141606/robotics-special-report-vision-enables-tube-pickingplacing-at-160min.

[11] [Ηλεκτρονικό]. Available: https://resources.hkedcity.net/res_files/201101/20110128101339_646297.pdf.

[12] Implementation of Robot Systems.

[13] [Ηλεκτρονικό]. Available: https://www.sagerobot.com/pharmaceutical/.

[1 [Ηλεκτρονικό]. Available: https://blog.technavio.org/blog/major-types-of-industrial-robots.
4]

[1 [Ηλεκτρονικό]. Available: https://strobotics.com/cylindrical-format-robot.htm.
5]

[1 [Ηλεκτρονικό]. Available: https://technicaintl.com/products/3-axis-delta-robot-packer/.
6]

[1 [Ηλεκτρονικό]. Available: https://www.shibaura-
7] machine.co.jp/en/product/robot/lineup/th/thl1200.html.

[1 [Ηλεκτρονικό]. Available: https://www.honeywell.com/us/en/news/2022/04/how-robots-are-
8] upgrading-the-supply-chain-industry.

[1 P. S. Saha. [Ηλεκτρονικό]. Available: https://web.iitd.ac.in/~saha/ethiopia/appln.pdf.
9]

[2 [Ηλεκτρονικό]. Available: https://howtorobot.com/expert-insight/painting-robots.
0]

[2 [Ηλεκτρονικό]. Available: https://www.irjet.net/archives/V8/i9/IRJET-V8I9260.pdf.
1]

[2 [Ηλεκτρονικό]. Available: https://www.universal-robots.com/products/ur3-robot/.
2]

[2 [Ηλεκτρονικό]. Available: https://www.online-sciences.com/robotics/collaborative-robots-universal-
3] robots-ur3-importance-uses-advantages-and-disadvantages/.

[2 [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Robot_kinematics.
4]

[2 [Ηλεκτρονικό]. Available: https://forum.universal-robots.com/t/the-dimensions-of-ur3-step-file-are-
5] different-from-dh-parameter/8525.

[2 [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Kinematics_equations.
6]

[2 [Ηλεκτρονικό]. Available:
7] https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters.

[2 [Ηλεκτρονικό]. Available:
8] https://www.researchgate.net/publication/274498442/figure/fig2/AS:642913100906498@1530293836652/The-two-postures-of-2-R-Robot.png.

[2 [Ηλεκτρονικό]. Available: https://journals.sagepub.com/doi/full/10.5772/50203.
9]

[3 [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Inverse_kinematics.
0]

[3 [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/a-Configuration-of-the-mobile-robot-
1] b-the-kinematic-model_fig1_317343312.

[3 [Ηλεκτρονικό]. Available: https://i.stack.imgur.com/mk4eR.png.
2]

[3 J. Thushara Sandakalum and Marcelo H. Ang. [Ηλεκτρονικό]. Available:
3] https://www.mdpi.com/2075-1702/10/2/97/htm.

[3 [Ηλεκτρονικό]. Available:
4] http://hades.mech.northwestern.edu/images/archive/2/2a/20160922142253!Park-lynch.pdf.

[3 [Ηλεκτρονικό]. Available: https://www.roboticautomationsystems.com/blog/what-is-a-work-
5] envelope/.

[3 [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/Manipulators-and-their-work-
6] envelopes-a-Cartesian-manipulator-b-gantry-c_fig1_322210953.

[3 [Ηλεκτρονικό]. Available:
7] https://www.researchgate.net/publication/273897449_Obstacle_Avoidance_with_Industrial_Robots.

[3 [Ηλεκτρονικό]. Available:
8] https://www.researchgate.net/publication/283851753_A_Review_on_Motion_Planning_and_Obstacle_Avoidance_Approaches_in_Dynamic_Environments.

[3 [Ηλεκτρονικό]. Available: https://eprints.whiterose.ac.uk/168146/1/LR_7.pdf.
9]

[4 [Ηλεκτρονικό]. Available:
0] https://www.researchgate.net/publication/282054384_Path_planning_A_2013_survey.

[4 [Ηλεκτρονικό]. Available: https://blogs.mathworks.com/student-lounge/2019/11/06/robot-
1] manipulator-trajectory/.

[4 [Ηλεκτρονικό]. Available: http://mathdep.ifmo.ru/wp-content/uploads/2018/10/Intelligent-Robotics-
2] and-Autonomous-Agents-series-Choset-H.-et-al.-Principles-of-Robot-Motion_-Theory-Algorithms-and-Implementations-MIT-2005.pdf.

[43] [Ηλεκτρονικό]. Available: https://www.codingeek.com/data-structure/graph-introductions-explanations-and-applications/attachment/weighted.

[44] [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/a-An-example-of-undirected-graph-and-b-an-example-of-directed-graph_fig3_50591619.

[45] [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm.

[46] [Ηλεκτρονικό]. Available: https://link.springer.com/article/10.1007/s10845-021-01867-z.

[47] [Ηλεκτρονικό]. Available: http://www.cse.unsw.edu.au/~billw/Justsearch.html.

[48] [Ηλεκτρονικό]. Available: https://downloads.hindawi.com/journals/jr/2022/2538220.pdf.

[49] [Ηλεκτρονικό]. Available: https://media.neliti.com/media/publications/396982-greedy-a-star-and-dijkstras-algorithms-i-44c18ff5.pdf.

[50] [Ηλεκτρονικό]. Available: https://www.hindawi.com/journals/jcse/2016/7426913/.

[51] [Ηλεκτρονικό]. Available: https://awards.acm.org/athena.

[52] [Ηλεκτρονικό]. Available: http://mathdep.ifmo.ru/wp-content/uploads/2018/10/Intelligent-Robotics-and-Autonomous-Agents-series-Choset-H.-et-al.-Principles-of-Robot-Motion_-Theory-Algorithms-and-Implementations-MIT-2005.pdf.

[53] [Ηλεκτρονικό]. Available: http://motion.cs.illinois.edu/RoboticSystems/MotionPlanningHigherDimensions.html#fig:PRM.

[54] [Ηλεκτρονικό]. Available: https://www.mdpi.com/1424-8220/23/2/1041.

[55] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Steven_M._LaValle.

[56] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/James_J._Kuffner_Jr..

[57] [Ηλεκτρονικό]. Available: https://www.hindawi.com/journals/jcse/2016/7426913/.

[58] [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/A-basic-RRT-based-algorithm-for-finding-a-collision-free-path-between-q-start-and-q-goal_fig2_224141569.

[59] [Ηλεκτρονικό]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0168169918303971.

[60] [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/RRT-growth-process-diagram-Black-circles-are-obstacles-Black-points-are-RRT-path_fig2_337881983.

[61] [Ηλεκτρονικό]. Available: https://www.researchgate.net/figure/The-Connect-method-from-the-RRT-Connect-algorithm_fig2_348299368.

[62] [Ηλεκτρονικό]. Available: https://stanfordasl.github.io/aa274a/pdfs/notes/lecture2.pdf.

[63] [Ηλεκτρονικό]. Available: https://github.com/ekorudiawan/ur_description/tree/master/urdf.

[64] [Ηλεκτρονικό]. Available: https://github.com/ekorudiawan/ur_gazebo.

[65] [Ηλεκτρονικό]. Available: https://github.com/JenniferBuehler/gazebo-pkgs.

[66] [Ηλεκτρονικό]. Available: http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/motion_planning_pipeline/motion_planning_pipeline_tutorial.html.

[67] [Ηλεκτρονικό]. Available: http://wiki.ros.org/ROS/Tutorials.

[68] [Ηλεκτρονικό]. Available: https://www.packworld.com/machinery/robotics/article/21141606/robotics-special-report-vision-enables-tube-pickingplacing-at-160min.

[69]

[70]

[71] Δ. Φ. Γιάννης, «Ρομποτική».

[7 2] [Ηλεκτρονικό]. Available: https://sites.google.com/site/dilekbasaranbil/courses/ceng786/homework-2?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1.

[7 3] [Ηλεκτρονικό]. Available: http://www.robogrok.com/index.html.

[7 4] [Ηλεκτρονικό]. Available: https://automaticaddison.com/how-to-find-the-rotation-matrices-for-robotic-arms/.

[7 5] [Ηλεκτρονικό]. Available: https://automaticaddison.com/how-to-find-displacement-vectors-for-robotic-arms/.

[7 6] [Ηλεκτρονικό]. Available: https://www.researchgate.net/publication/353050713_Robotic_Obstacle_Avoidance_A_Virtual_Modeling_And_Reinforcement_Learning/link/60e64ee5b8c0d5588cde73b5/download.

[7 7] [Ηλεκτρονικό]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/152/1/012064/pdf.

[7 8] [Ηλεκτρονικό]. Available: https://www.researchgate.net/publication/277816913_Path_Planning_Based_on_Bi-RRT_Algorithm_for_Redundant_Manipulator.

[7 9] [Ηλεκτρονικό]. Available: https://www.hindawi.com/journals/jcse/2016/7426913/.

[8 0] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Inverse_kinematics.

[8 1] [Ηλεκτρονικό]. Available: https://blogs.mathworks.com/student-lounge/2019/11/06/robot-manipulator-trajectory/.

[8 2] [Ηλεκτρονικό]. Available: https://eclass.uniwa.gr/modules/document/file.php/137/%CE%A3%CE%97%CE%9C%CE%95%CE%99%CE%A9%CE%A3%CE%95%CE%99%CE%A3/5.PATH_PLANNING.pdf.

Πήγες για την εκτέλεση της εφαρμογής

Αρχεία που χρησιμοποιήθηκαν για την εκτέλεση του project

Για τα urdf μοντέλα: ρομποτ ur3, rg2, table [63]

Για την σύνδεση gazebo με MoveIt [64]

Για την βοήθεια της αρπάγης στην άπραγη αντικειμένου [65]

Πήγες που χρησιμοποιήθηκαν για την εκτέλεση του project

Για την εκμάθηση του εργαλείου MoveIt [66]

Για την εκμάθηση του “λογισμικού Ros” [67]

Παραρτήματα

ΚΩΔΙΚΑΣ movement1.py

```
1. import sys
2. import copy
3. import rospy
4. import moveit_commander
5. import moveit_msgs.msg
6. import geometry_msgs.msg
7. import math as m
8. from tf.transformations import *
9. from gazebo_msgs.srv import SpawnModel
10. from gazebo_msgs.srv import DeleteModel
11. from geometry_msgs.msg import Pose
12.
13. moveit_commander.roscpp_initialize(sys.argv)
14. # node for adding pick and place
15. rospy.init_node('pick_and_place', anonymous=True)
16.
17. # moveitcommander
18. robot = moveit_commander.RobotCommander()  # motion on movit
19. scene = moveit_commander.PlanningSceneInterface()
20.
21. group = moveit_commander.MoveGroupCommander("arm")  # wrist
22. group2 = moveit_commander.MoveGroupCommander("hand")  # gripper
23. display_trajectory_publisher = rospy.Publisher(
24.     '/move_group/display_planned_path',
    moveit_msgs.msg.DisplayTrajectory, queue_size=20)
25.
26. # set joint values to move robot arm
27. group_variable_values = group.get_current_joint_values()
28. group_variable_values[0] = -0.0915172139475     # joint 1
29. group_variable_values[1] = -0.176953153593  # joint 2
30. group_variable_values[2] = 0.379329819822  # joint 3
31. group_variable_values[3] = -0.199541716426  # joint 4
32. group_variable_values[4] = -0.0916237938249  # joint 5
33. group_variable_values[5] = -0.00286632744277   # joint 6
34. group.set_joint_value_target(group_variable_values)
35. plan1 = group.plan()
36.
37. group.execute(plan1, wait=True)
38.
```

```
# shut down moveit_commander
moveit_commander.roscpp_shutdown()
```

ΚΩΔΙΚΑΣ objects.py

```
import rospy
from moveit_msgs.msg import CollisionObject
from geometry_msgs.msg import PoseStamped, Pose, Point, Quaternion
from shape_msgs.msg import SolidPrimitive
from moveit_commander import MoveGroupCommander
from gazebo_msgs.srv import SpawnModel
from gazebo_msgs.msg import ModelState
from moveit_msgs.msg import ContactInformation
rospy.init_node('collision_object_publisher')

# Initialize a MoveGroupCommander object
group_name = "arm"
move_group = MoveGroupCommander(group_name)

# Create a CollisionObject message
collision_object = CollisionObject()
collision_object.header.frame_id = 'world'  # Set the frame ID of the
object

# Set the ID of the collision object (should be unique)
collision_object.id = 'box'

# Set the dimensions of the box
box_size = [0.1, 0.2, 0.8]

# Set the pose of the box
box_pose = PoseStamped()
box_pose.header.frame_id = 'world'
box_pose.pose.position = Point(x=0.0, y=0.04, z=1.28)

# Set the orientation of the box
box_orientation = Quaternion(x=0.0, y=0.0, z=0.0, w=1.0)
box_pose.pose.orientation = box_orientation

    # Assign the pose to the collision object
    collision_object.primitives.append(SolidPrimitive(type=SolidPrimiti
ve.BOX, dimensions=box_size))
    collision_object.primitive_poses.append(box_pose.pose)
```

```
104.
105.       # Publish the collision object
106.       co_pub = rospy.Publisher('/collision_object', CollisionObject,
    queue_size=1)
107.       rospy.sleep(2)  # Allow time for the publisher to connect
108.       co_pub.publish(collision_object)
109.
110.       # Create a model state message for the box object
111.       model_state = ModelState()
112.       model_state.model_name = 'box'
113.       model_state.pose = box_pose.pose
114.
115.       # Create a service proxy for the spawn_sdf_model service
116.       rospy.wait_for_service('/gazebo/spawn_sdf_model')
117.       spawn_sdf_model = rospy.ServiceProxy('/gazebo/spawn_sdf_model',
    SpawnModel)
118.
119.       # Open the model SDF file and read the XML content
120.       bin_file = open('/home/user/model_editor_models/project/model.sdf')
121.       bin_xml = bin_file.read()
122.
123.       # Set the pose of the object
124.       bin_pose = Pose()
125.       bin_pose.position.x = 0
126.       bin_pose.position.y = 0.04
127.       bin_pose.position.z = 1.28
128.
129.       # Set the name of the model and the frame ID
130.       bin_name = 'bin'
131.       reference_frame = 'world'
132.
133.       # Call the spawn_sdf_model service to spawn the object
134.       spawn_sdf_model(bin_name, bin_xml, '', bin_pose, reference_frame)
```

ΚΩΔΙΚΑΣ randomcodeevaluationrrt.py

```
1. import sys
2. import copy
3. import rospy
4. import moveit_commander
5. import moveit_msgs.msg
6. import geometry_msgs.msg
7. import math as m
8. from tf.transformations import *
9. from gazebo_msgs.srv import SpawnModel
10. from gazebo_msgs.srv import DeleteModel
11. from geometry_msgs.msg import Pose
12. import numpy as np
13. import time
14.
15. moveit_commander.roscpp_initialize(sys.argv)
16. # node for adding pick and place
17. rospy.init_node('pick_and_place', anonymous=True)
18.
19. # moveitcommander
20. robot = moveit_commander.RobotCommander()
21. scene = moveit_commander.PlanningSceneInterface()
22.
23. group = moveit_commander.MoveGroupCommander("arm")
24. group2 = moveit_commander.MoveGroupCommander("hand")
25. display_trajectory_publisher = rospy.Publisher(
26.     '/move_group/display_planned_path',
    moveit_msgs.msg.DisplayTrajectory, queue_size=20)
27.
28.
29. # thesi prin koytaki
30. group_variable_values = group.get_current_joint_values()
31. group_variable_values[0] = -0.0915172139475
32. group_variable_values[1] = -0.176953153593
33. group_variable_values[2] = 0.379329819822
34. group_variable_values[3] = -0.199541716426
35. group_variable_values[4] = -0.0916237938249
36. group_variable_values[5] = -0.00286632744277
37. group.set_joint_value_target(group_variable_values)
38. plan1 = group.plan()
39. rospy.sleep(0.1)
40. group.go(wait=True)
41.
42. # THESI telos
```

```
group_variable_values = group.get_current_joint_values()
group_variable_values[0] = 0.0118702462598   # joint 0
group_variable_values[1] = -3.14153438889
group_variable_values[2] = -0.0175243176233  # joint 0
group_variable_values[3] = -3.1415700392
group_variable_values[4] = 0.0237287330466  # joint 0
group_variable_values[5] = -0.0952336759661
group.set_joint_value_target(group_variable_values)

# Plan the trajectory from the start to goal configuration
plan_start_time = time.time()
plan1 = group.plan()
plan_time = time.time() - plan_start_time

# Execute the planned trajectory
exe_start_time = time.time()
group.go(wait=True)
exe_time = time.time() - exe_start_time

# Calculate the path length
path_len = 0
previous_point = None
for point in plan1.joint_trajectory.points:
    if previous_point is not None:
        path_len += np.linalg.norm(np.array(point.positions) -
np.array(previous_point.positions))
    previous_point = point

# Get the end-effector position
end_effector_pose = group.get_current_pose().pose

# Output
print("Planning time: {:.4f} s".format(plan_time))
print("Execution time: {:.4f} s".format(exe_time))
print("Path length: {:.4f} m".format(path_len))
print("End-effector position: ({:.4f}, {:.4f},
{:.4f})".format(end_effector_pose.position.x,
end_effector_pose.position.y, end_effector_pose.position.z))
print(group.get_planner_id())

moveit_commander.roscpp_shutdown()
```

ΚΩΔΙΚΑΣ pick_and_place.py

```
1. #!/usr/bin/env python
2.
3. import sys
4. import copy
5. import rospy
6. import moveit_commander
7. import moveit_msgs.msg
8. import geometry_msgs.msg
9. import math as m
10.
11.from tf.transformations import *
12.from gazebo_msgs.srv import SpawnModel
13.from gazebo_msgs.srv import DeleteModel
14.from geometry_msgs.msg import Pose
15.
16.moveit_commander.roscpp_initialize(sys.argv)
17.# node for adding pick and place
18.rospy.init_node('pick_and_place', anonymous=True)
19.
20.# moveitcommander
21.robot = moveit_commander.RobotCommander()  # motion on movit
22.scene = moveit_commander.PlanningSceneInterface()
23.
24.group = moveit_commander.MoveGroupCommander("arm")  # wrist
25.group2 = moveit_commander.MoveGroupCommander("hand")  # gripper
26.display_trajectory_publisher = rospy.Publisher(
27.    '/move_group/display_planned_path',
   moveit_msgs.msg.DisplayTrajectory, queue_size=20)
28.
29.
30.# spawn object
31.can_file = open("/home/user/.gazebo/models/coke_can/model.sdf")
32.can_xml = can_file.read()
33.
34.#bin_file = open("/home/user/model_editor_models/project/model.sdf")
35.#bin_xml = bin_file.read()
36.
37.rospy.wait_for_service("/gazebo/spawn_sdf_model")
38.
39.#bin_pose = Pose()
40.#bin_pose.position.x = 0
41.#bin_pose.position.y = 0.04
42.#bin_pose.position.z = 1.28
```

```
43.
44.can_pose = Pose()
45.can_pose.position.x = 0.452  # 0.459  + aristera robot
46.can_pose.position.y = 0.06  # 0.085 + pisw robot
47.can_pose.position.z = 1.12  # 1.12
48.
49.q = quaternion_from_euler(0, 0, m.radians(180))
50.can_pose.orientation.x = q[0]
51.can_pose.orientation.y = q[1]
52.can_pose.orientation.z = q[2]
53.can_pose.orientation.w = q[3]
54.
55.spawn_object_srv = rospy.ServiceProxy("/gazebo/spawn_sdf_model",
   SpawnModel)
56.#spawn_object_srv("bin", bin_xml, "", bin_pose, "world")
57.spawn_object_srv("can", can_xml, "", can_pose, "world")
58.
59.# open hand
60.group2_variable_values = group2.get_current_joint_values()
61.group2_variable_values[0] = 1.52526710353
62.group2.set_joint_value_target(group2_variable_values)
63.plan2 = group2.plan()
64.rospy.sleep(0.5)
65.group2.go(wait=True)
66.
67.# position before grasp the object
68.rospy.sleep(0.5)
69.# position before grasp the object
70.group_variable_values = group.get_current_joint_values()
71.group_variable_values[0] = -0.0915172139475    # joint 0
72.group_variable_values[1] = -0.176953153593
73.group_variable_values[2] = 0.379329819822  # joint 0
74.group_variable_values[3] = -0.199541716426
75.group_variable_values[4] = -0.0916237938249  # joint 0
76.group_variable_values[5] = -0.00286632744277
77.group.set_joint_value_target(group_variable_values)
78.plan1 = group.plan()
79.rospy.sleep(0.1)
80.group.go(wait=True)
81.
82.spawn_object_srv("can", can_xml, "", can_pose, "world")
83.
84.# open hand
85.group2_variable_values = group2.get_current_joint_values()
```

```
group2_variable_values[0] = 1.52526710353     # joint 0
group2.set_joint_value_target(group2_variable_values)
plan2 = group2.plan()
rospy.sleep(0.5)
group2.go(wait=True)

# the position that the can exist
group.set_named_target("zero")
plan1 = group.plan()
rospy.sleep(0.5)
group.execute(plan1, wait=True)


#  the hand close to grasp the object
    group2_variable_values = group2.get_current_joint_values()
    group2_variable_values[0] =  0.1    # joint 0 - 0.282852425575 , - 2
    group2.set_joint_value_target(group2_variable_values)
    plan2 = group2.plan()
    rospy.sleep(0.5)
    group2.go(wait=True)


    # last position of robotic arm
    group_variable_values = group.get_current_joint_values()
    group_variable_values[0] = 0.0118702462598    # joint 0
    group_variable_values[1] = -3.14153438889
    group_variable_values[2] = -0.0175243176233  # joint 0
    group_variable_values[3] = -3.1415700392
    group_variable_values[4] = 0.0237287330466  # joint 0
    group_variable_values[5] = -0.0952336759661
    group.set_joint_value_target(group_variable_values)
    plan1 = group.plan()
    rospy.sleep(0.1)
    group.go(wait=True)

    delete_model_prox = rospy.ServiceProxy('gazebo/delete_model',
DeleteModel)
    delete_model_prox('can')

    moveit_commander.roscpp_shutdown()
```