ΤΕΙ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έλεγχος ρομποτικού οχήματος με χρήση τεχνικών ευφυούς ελέγχου

ΜΑΚΡΗΣ ΝΙΚΗΤΑΣ

Πίνακας Περιεχομένων

1. Κεφάλαιο 1: Εισαγωγή

Εδώ και αρκετά χρόνια υπάρχει και αναπτύσσεται ο τομέας της ρομπότικής με υψηλούς ρυθμούς. Προγραμματισμένες μηχανές εκτελούν διάφορες εργασίες, που ο άνθρωπος δε δύναται, ή καθίσταται επικίδυνο να κάνει. Όλα τα ρομπότ μέχρι σήμερα όμως κάνουν αυτό που έχουν προγραμματιστεί να κάνουν και δεν πρόκειται κάθε φορά που εκτελούν το πρόγραμμα τους, να το κάνουν καλύτερα από την προηγούμενη φορά.

Τα τελευταία χρόνια όμως έχουν αρχίσει να αναπτύσσονται τεχνικές βελτίωσης αυτών των προγραμματισμένων εργασιών. Έχουν ήδη δημιουργηθεί ρομπότ, που μπορούν και εκτιμούν σύμφωνα με κάποιες παραμέτρους την εκτέλεση του προγράμματος τους και βελτιώνουν την απόδοσή του. Αυτή είναι μια μέθοδος αυτοβελτίωσης, η οποία μπορεί να γίνει με πολλούς διάφορους τρόπους. Ενδεικτικά μπορούν να γίνουν με εξελικτικούς αλγορίθμους οι οποίοι μπορούν να βελτιώσουν την απόδοση με τη συνεχή εκτέλεση του προγράμματος. Για παράδειγμα σήμερα υπάρχει αυτοκινούμενο αμάξι το οποίο πλοηγείται σε χαρτογραφημένες περιοχές από GPS, αλλάζει μόνο του την διεύθυνση και την ταχύτητα του και διορθώνει την συμπεριφορά του κάθε φορά που εκτελέι το συγκεκριμένο δρομολόγιο, με ή ακόμα και μερικές φορές χωρίς την βοήθεια του οδηγού.

Ένα πολύ καλό ερώτημα είναι, σύμφωνα με ποιες παραμέτρους θα βελτιώνεται ο ελεγκτής. Αυτό θα ορίζεται ανάλογα με τις ανάγκες που προκύπτουν από την χρήση για την οποία προορίζεται. Στον παράδειγμα του αυτοκινήτου, θα μπορούσε να είναι η ταχύτητα, τα χιλιόμετρα ή ακόμα και η απόσταση από ένα τρίτο σημείο.

Τέτοιας μορφής προγράμματα, που αυτοβελτιώνονται καθίστανται πολύ δύσκολα στην υλοποίηση τους κι αυτό οφείλεται στην πολυπλοκότητα και δυσκολία περιγραφής της λειτουργίας του ανθρώπινου εγκεφάλου, ώστε να μπορέσει αυτό υλοποιηθεί ως προγραμματισμένη συμπεριφορά.

Σκοπός αυτής της εργασίας είναι να υλοποιήσει ένα τέτοιου είδους πρόγραμμα / ελεγκτή που θα ελέγχει την κίνηση ενός έντροχου ρομπότ με χρήση τεχνικών που μιμούνται ή/και προσομοιώνουν την ανθρώπινη συμπεριφορά. Ένας τέτοιος τύπος ελεγκτή είναι αυτός που βασίζεται σε μια μεθοδολογία από την ευρύτερη περιοχή του ευφυούς ελέγχου και συγκεκριμένα από την ασαφή λογική.

Για να γίνει βέβαια ένα τέτοιο πρόγραμμα απαιτεί κάποιες βασικές γνώσεις που πρέπει να γίνουν γνωστές πρώτα πριν γίνει η υλοποίηση του, ώστε να κατανοηθεί η λογική με την οποία μπορεί να υλοποιηθεί κάτι τέτοιο.

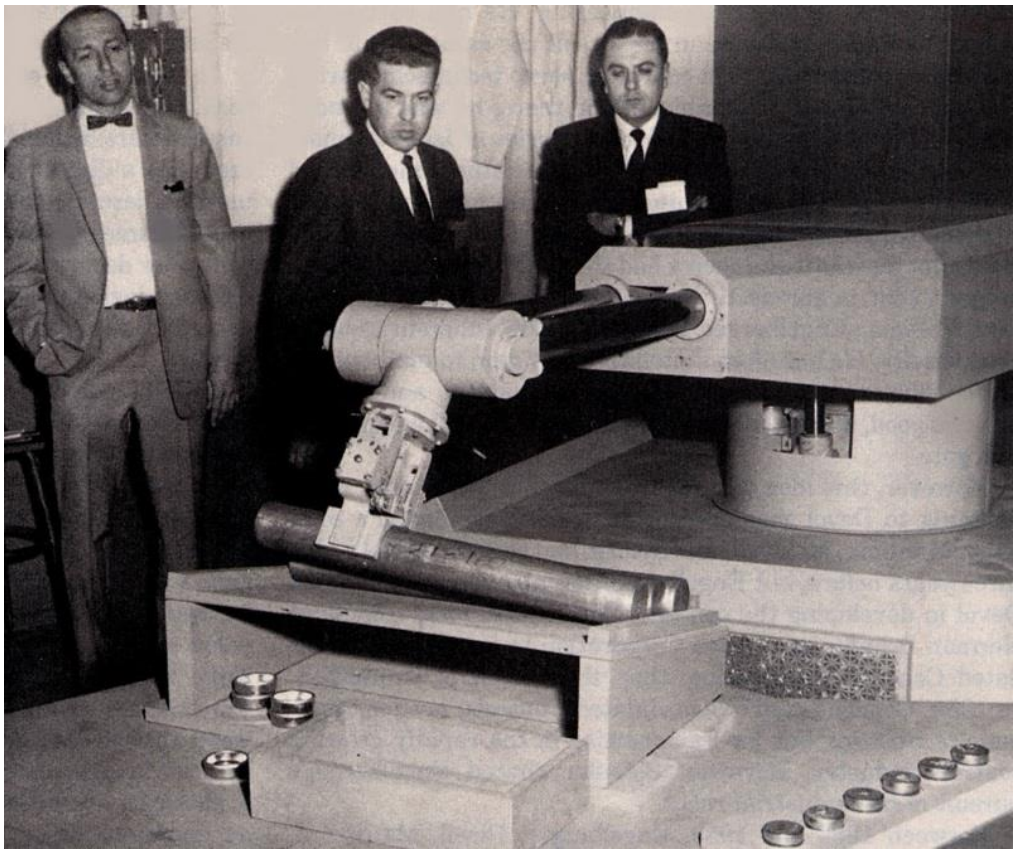
2. Κεφάλαιο 2: Ρομποτική

2.1 Γενικά για τη Ρομποτική

Γενικά η ρομποτική είναι κλάδος της τεχνολογίας που ασχολείται με τη σχεδίαση, την ανάπτυξη και τη μελέτη ρομπότ. Η επιστήμη της ρομποτικής αποτελεί συνδυασμό πολλών άλλων επιστημών, κυρίως της πληροφορικής, της ηλεκτρονικής και της μηχανολογίας. Η λέξη ρομπότ προέρχεται από το Σλάβικο robota που σημαίνει εργασία και είναι αυτόματες μηχανές με προγραμματισμένη συμπεριφορά, η χρήση των οποίων αποσκοπεί στην αντικατάσταση του ανθρώπου στην εκτέλεση έργου, τόσο σε φυσικό επίπεδο όσο και σε επίπεδο λήψης αποφάσεων.

Η ρομποτική επιστήμη έχει κάνει άλματα προόδου και έχει προσφέρει αρκετά τεχνολογικά θαύματα. Τα ρομποτικά συστήματα συνεχώς εξελίσσονται και είναι ήδη μέρος της ζωής μας σε πολλούς τομείς όπως στη βιομηχανία, την ιατρική, τη διασκέδαση και την προσωπική βοήθεια.

Ιστορικά βέβαια, ο όρος ρομπότ και η ιδέα αυτού, πρωτοεμφανίστηκε το 1921 σε ένα θεατρικό έργο ενός Τσέχου συγγραφέα και το 1961 κατασκευάζεται το πρώτο ρομπότ και τίθεται σε λειτουργία σε βιομηχανική μονάδα (Εικόνα 1**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**). Από τότε η επιστήμη της ρομποτικής έχει προχωρήσει τόσο πολύ, που σήμερα γίνονται χειρουργικές επεμβάσεις χρησιμοποιώντας ρομποτικά συστήματα με απίστευτες ακρίβειες κινήσεων (Εικόνα 2).



Εικόνα 1: Πρώτο Βιομηχανικό Ρομπότ (arduinobots.wordpress.com)

Εικόνα 2:Ρομποτική Χειρουργική (www.axiarchos.gr/robotiki-xeirourgiki)

Ένα ρομπότ συγκροτείται από δύο συστήματα, το μηχανικό και το ηλεκτρονικό. Υπάρχουν διάφορα κριτήρια διάκρισης και αντίστοιχες κατηγοριοποιήσεις των ρομπότ. Μία από αυτές είναι η διάκριση τους σε τρεις, επί του παρόντος γενιές. Στην πρώτη γενιά κατατάσσονται ρομπότ με περιορισμένη ευελιξία που διευθύνονται από τον άνθρωπο. Στη δεύτερη γενιά κατατάσσονται τα ρομπότ που είναι εφοδιασμένα με σταθερό πρόγραμμα δράσης και ρομπότ που λαμβάνουν εντολές από κάποιο αριθμητικό σύστημα ελέγχου. Και στην τρίτη γενιά κατατάσσονται τα ρομπότ που είναι εφοδιασμένα με αισθητήριες πληροφορίες από το περιβάλλον, με διάταξη επεξεργασίας πληροφορίων και με κινητήριο σύστημα εκτέλεσης εργασιών.

2.2 Έντροχα Ρομποτικά Οχήματα

Όσο αναφορά την κίνηση των ρομπότ, υπάρχουν δύο βασικές κατηγορίες. Τα ακίνητα ρομπότ, όπως βραχίονες, βιομηχανικού τύπου ή και άλλου που αποσκοπούν στην εκτέλεση εργασιών μέσα σε ένα ορισμένο χώρο και υπάρχουν και τα κινούμενα ρομποτικά συστήματα. Τα τελευταία μπορούν να χωριστούν επίσης σε επί μέρους βασικές κατηγορίες, όπως τα ανθρωποειδή ρομπότ, που διαθέτουν άκρα που μιμούνται τα ανθρώπινα άκρα, τα έντροχα, που διαθέτουν τροχούς (Εικόνα 4) ή ερπύστριες (Εικόνα 3) για την κίνηση τους, τα υποβρύχια ρομπότ, που χρησιμοποιούν προπέλες και τα ιπτάμενα ρομπότ, με έλικες όπως quad copter κτλ.

Εικόνα 3: Foster Miller TALON Robot (robotica.gr)

Εικόνα 4: Pioneer Robot (pyrorobotics.com)

Τα έντροχα ρομποτικά οχήματα δεν περιορίζονται μόνο σε επίπεδες επιφάνειες, αλλά με τον κατάλληλο εξοπλισμό μπορούν να ανέβουν και σκαλιά, να διανύσουν πολύ γρήγορα μεγάλες αποστάσεις και να διεισδύσουν σε μικρούς χώρους. Επίσης επειδή δεν βρίσκονται ούτε στον αέρα, αλλά ούτε κάτω από το νερό, δεν υπόκεινται σε περιορισμούς τόσο όσο τα ιπτάμενα και τα υποβρύχια, όπως για παράδειγμα το βάρος τους, ή το μέγεθός τους. Λόγω αυτού μπορούν να διαθέτουν βραχίονες για την διεκπεραίωση πολλών εργασιών, ταυτόχρονα κιόλας με την κίνηση τους και επίσης μπορούν να έχουν πολλούς αισθητήρες πάνω τους, γεγονός που είναι πιο δύσκολο να γίνει με ένα υποβρύχιο ρομπότ, λόγω του νερού και της πίεσης του.

Βασικά θέματα που απασχολούν και παίζουν πολύ σημαντικό ρόλο σε όλα τα κινούμενα ρομπότ είναι η αναγνώριση της θέσης τους και του χώρου που τα περιβάλλει. Εστιάζοντας στα έντροχα ρομπότ, τα οποία περιορίζονται σε (ιδανικά) επίπεδη κίνηση, για να λυθεί το πρόβλημα αυτό μπορεί να χρησιμοποιηθεί οδομετρία.
Η οδομετρία βασίζεται σε αισθητήρες που δίνουν πληροφορίες στο ρομπότ, όπως υπερηχητικοί, λέιζερ, οπτικοί, για το περιβάλλον γύρω τους και με βάση αυτές τις πληροφορίες υπολογίζονται η στροφή των τροχών και ο προσανατολισμός του ρομπότ. Βασικό πρόβλημα της οδομετρίας είναι το γεγονός ότι απαιτεί μια αναφορά, βάσει της οποίας θα μπορεί να εξάγει τις σχετικές αποκλίσεις.

Ένας συμπληρωματικός τρόπος εύρεσης θέσης είναι με χρήση γυροσκοπίων για τη μέτρηση στροφής και επιτάχυνσης. Πιο παλιά τα γυροσκόπια ενώ ήταν ακριβή, δεν ήταν

ικανά να διατηρήσουν την ακρίβεια τους, γεγονός που τα καθιστούσε μη αποδεκτή λύση στα ρομπότ. Όμως τώρα με τα λέιζερ γυροσκόπια(fiber-optic gyros), έχουν πολύ μεγάλη αντοχή, ακρίβεια και έχουν γίνει πλέον πιο φθηνά, αποτελούν μια πολύ ελκυστική λύση.
Μια τελείως διαφορετική προσέγγιση στην εύρεση θέσης, που είναι πολύ αξιόπιστη, όμως δεν εφαρμόζεται σε όλες τις περιπτώσεις, είναι ο υπολογισμός θέσης μέσω της τοποθέτησης τριών σταθερών σημείων στο χώρο και υπολογισμού της θέσης σε σχέση με αυτά. Προφανώς η αξιοπιστία της μεθόδου είναι πολύ καλή, όμως υπάρχουν χώροι που δεν διαθέτουν τέτοια σημεία. Για να μπορέσει να εφαρμοστεί η μέθοδος ακόμα και σε τέτοια περιβάλλοντα, χρησιμοποιούνται τεχνητά σημεία, γεγονός που μπορεί να μην είναι πάντα βολικό.

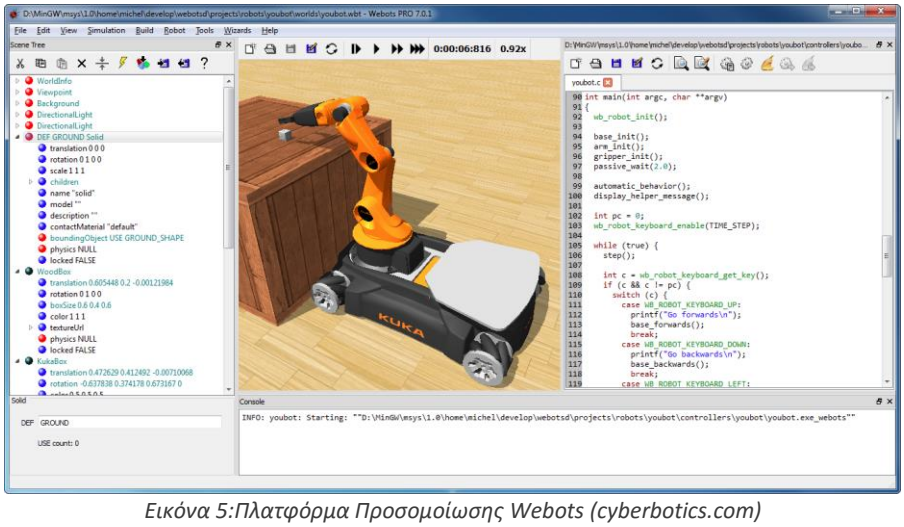
Υπάρχει επίσης η μέθοδος αναγνώρισης φυσικών σημείων, η οποία απαιτεί να είναι χαρτογραφημένη η περιοχή πριν, και δεν έχει τόσο καλή αξιοπιστία. Συνδυάζοντας όμως γεωμετρικούς χάρτες και τοπολογικούς χάρτες της περιοχής αυτής, μπορεί να γίνει αντιστοίχιση μοντέλων στους χάρτες και να γίνει η αντιστοίχιση της θέσης.

Προφανώς όλες οι παραπάνω μέθοδοι χρησιμοποιούνται, ανάλογα με τις απαιτήσεις που υπάρχουν για την κίνηση κάθε ρομπότ, αλλά και του χώρου στον οποίο βρίσκεται. (Για παράδειγμα, δεν μπορεί να χρησιμοποιηθεί εύρεση θέσης με μέθοδο αντιστοίχησης μοντέλου σε ρομπότ διάσωσης θυμάτων σεισμού.)

2.3 Προσομοίωση Ρομποτικού Συστήματος

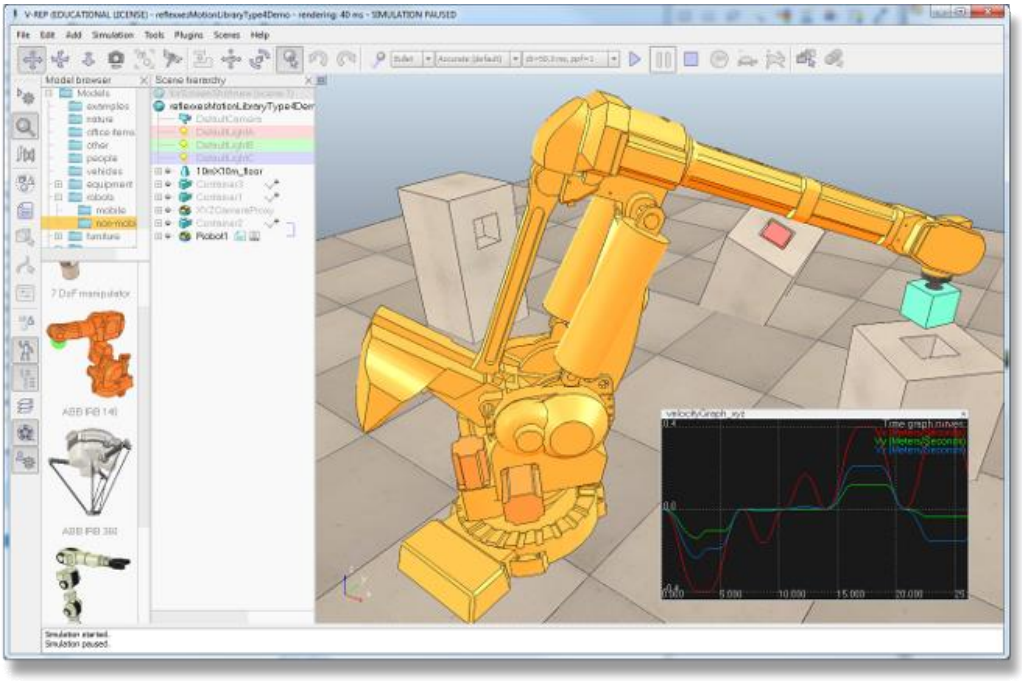
Για τις ανάγκες έρευνας πάνω στον έλεγχο των ρομποτικών συστημάτων, της οδομετρίας κ.α. έχουν δημιουργηθεί κάποια πολύ χρήσιμα εργαλεία με τα οποία μπορούμε να προσομοιώνουμε σε εικονικό περιβάλλον υπάρχοντα ρομποτικά συστήματα, με δικές μας ρουτίνες ελέγχου κίνησης. Δύο πάρα πολύ γνωστές πλατφόρμες προσομοίωσης είναι η Webots και V-Rep (Virtual Robot Exparimentation Program).

Η πλατφόρμα Webots (Εικόνα 5) είναι ένας τρισδιάστατος προσομοιωτής με αληθινά μοντέλα ρομπότ, με πιστή προσομοίωση της συμπεριφοράς τους και με πολύ καλή προσομοίωση φυσικής (physics), όπως τριβές, συγκρούσεις κτλ.



Εικόνα 5:Πλατφόρμα Προσομοίωσης Webots (cyberbotics.com)

Η πλατφόρμα V-Rep (Εικόνα 6) επίσης είναι τρισδιάστατος προσομοιωτής, επίσης με πιστά μοντέλα ρομπότ και με μεγάλη ποικιλία (από βραχίονες ως ανθρωποειδή NAO) και πολλή καλή προσομοίωση φυσικής, διάθεση πολλών αισθητήρων και πολλά άλλα.



Εικόνα 6:Πλατφόρμα V-Rep (coppeliarobotics.gr)

Και οι δυο πλατφόρμες είναι πολύ καλές και παρέχουν υποστήριξη άλλων, πιο γνωστών γλωσσών προγραμματισμού, όπως C++, Matlab, URBI, Python κτλ.
Τόσο στη Webots όσο και στο V-Rep μπορούμε να συνδέσουμε το Matlab και να δημιουργήσουμε αμφίδρομη επικοινωνία. Έτσι θα μπορούμε να προγραμματίσουμε στο matlab, και να ελέγχουμε το προσομοιωμένο ρομπότ στο V-Rep ή Webots.

3. Κεφάλαιο 3: Ασαφής Λογική

3.1 Βασικές Αρχές Ασαφούς Λογικής

Η ασαφής λογική χρησιμοποιείται κυρίως για τον έλεγχο μηχανών. Ο όρος ασαφή στην πραγματικότητα δίνεται για να αποδώσει την έννοια που δεν μπορεί να αποδοθεί με τις εκφράσεις True και False ή και Half True. Παρόλα αυτά εναλλακτικές προτάσεις όπως οι Γενετικοί Αλγόριθμοι και τα Νευρωνικά Δίκτυα μπορούν να αποδώσουν και να λειτουργήσουν τόσο καλά όσο η Ασαφής Λογική, αλλά η ασαφή λογική έχει το πλεονέκτημα και την επίλυση για προβλήματα που κοστίζουν στον άνθρωπο και τα ανθρώπινα συστήματα την δυσκολία να κατανοήσουν . Έτσι με την εμπειρία που αποκτά ο άνθρωπος από την ασαφή λογική μπορεί να σχεδιάσει και να κατασκευάσει ελεγκτές που τον βοηθούν στο να αντιμετωπίζει ευκολότερα μηχανισμούς και διαδικασίες που ίδει σήμερα έχει κατακτήσει ή ακόμα και να δημιουργήσει νέες καλύτερες.

Εφαρμογή της ασαφούς λογικής έχουμε σε πάρα πολλές εφαρμογές, όπως παρακάτω:

- ✓ Αυτόματα Κιβώτια Ταχυτήτων Αυτοκινήτων
- ✓ Ελεγκτές Ταξιδίου (GPS) Αυτοκινήτων και κινητών
- ✓ Ανελκυστήρες
- ✓ Αυτόματα Τρένα
- ✓ Βιολογικός Καθαρισμός Νερού
- ✓ Εναέρια Κυκλοφορία και πλοήγηση

και πολλές άλλες εφαρμογές.

Η ασαφής λογική δίνει μια ικανοποιητική λύση στη λεγόμενη αρχή του ασυμβίβαστου: «Καθώς η πολυπλοκότητα ενός συστήματος αυξάνει, η ικανότητα μας να προβαίνουμε σε ακριβείς και σημαντικές δηλώσεις για τη συμπεριφορά του μειώνεται μέχρι που να φθάσουμε σε ένα όριο πέρα του οποίου ακρίβεια και σημαντικότητα καθίστανται σχεδόν αμοιβαίως αποκλειόμενα χαρακτηριστικά». Είναι προφανές ότι η αρχή αυτή είναι απόρροια της κβαντικής αρχής της απροσδιοριστίας του Heisenberg.

Οι ασαφείς ελεγκτές αποτελούν συστήματα διακριτού χρόνου και επιπλέον χαρακτηρίζονται από έντονη μη γραμμικότητα. Αντιθέτως, οι ελεγκτές που χρησιμοποιούνται στις περισσότερες περιπτώσεις του αυτομάτου ελέγχου μπορεί να είναι διακριτού ή συνεχούς χρόνου και είναι κυρίως γραμμικοί. Είσοδοι και στις δύο περιπτώσεις αποτελούν το σφάλμα της εξόδου του ελεγχόμενου συστήματος με την είσοδο αναφοράς και τα χαρακτηριστικά αυτού του σφάλματος όπως η μεταβολή και ο ρυθμός μεταβολής του. Η έξοδος των ασαφών όσο και των γραμμικών ελεγκτών μπορεί να είναι το σήμα ελέγχου ή η προσαύξηση του σήματος αυτού, ανάλογα πάντα με την μορφή του ελεγκτή.

Το κύριο όμως μειονέκτημα των ασαφών ελεγκτών είναι ο μεγάλος αριθμός των παραμέτρων που πρέπει να ρυθμιστούν για να ικανοποιηθούν τα κριτήρια που έχουν τεθεί σε κάθε περίπτωση, ως προς την επιθυμητή απόκριση του ελεγχόμενου συστήματος. Η εύρεση του πεδίου τιμών των ασαφών μεταβλητών, η μορφή των συναρτήσεων συμμετοχής των ασαφών συνόλων, η επιλογή του μηχανισμού εξαγωγής συμπεράσματος και των τελεστών που χρησιμοποιεί η ασαφής λογική, ο σχεδιασμός της ασαφούς βάσης κανόνων, ο καθορισμός των πιθανών κερδών κλιμακοποίησης που μπορεί να διαθέτει ο

ελεγκτής, η επιλογή του χρόνου δειγματοληψίας και ένας αριθμός ακόμη παραμέτρων καθιστούν την διαδικασία ρύθμισης του ασαφούς ελεγκτή μια χρονοβόρα και δύσκολη διαδικασία.

Όπως κατά την εκμάθηση μιας γλώσσας προγραμματισμού τείνει να καθιερωθεί πλέον η πρώτη εφαρμογή να είναι η εκτύπωση στην οθόνη Hello World , για τους ίδιους λόγους έχει σχεδόν καθιερωθεί η πρώτη προσέγγιση στις έννοιες της ασαφούς λογικής να πραγματοποιείται με το παράδειγμα του φρεναρίσματος ενός αυτοκινήτου.

Ας υποθέσουμε λοιπόν ότι οδηγούμε ένα όχημα με κάποια ταχύτητα και σε κάποια απόσταση αντιλαμβανόμαστε ένα εμπόδιο. Για απλότητα ας θεωρήσουμε ότι το οδόστρωμα είναι επίπεδο και ομαλό. Το ερώτημα είναι ο τρόπος με τον οποίο θα χειριστούμε την κατάσταση. Ένα πολύ μαθηματικό μυαλό θα μπορούσε να αναγνωρίσει αμέσως την αναλυτική μαθηματική φόρμουλα F = f (d,u), η οποία θα του έδινε την δύναμη F με ακρίβεια κάποιων δεκαδικών ψηφίων που θα έπρεπε να ασκήσει στο φρένο του αυτοκινήτου, προκειμένου να σταματήσει ομαλά πριν από το εμπόδιο. Η σχέση βέβαια αυτή εξαρτάται από πολλούς παράγοντες, όπως ο τύπος του αυτοκινήτου, ο τύπος του φρένου, το βάρος του αυτοκινήτου η κατεύθυνση του ανέμου, κ.τ.λ. Μια τέτοια μαθηματική προσέγγιση επομένως είναι αδύνατη για τον κοινό νου.

Ακόμα και αν χρησιμοποιούσαμε ένα σύστημα κλασσικού αυτόματου ελέγχου (pid controller), τα πράγματα θα ήταν εξαιρετικά δύσκολα διότι ο κλασσικός εκλεκτής θα έπρεπε να ρυθμίζει τα κέρδη του ανάλογα με τις συνθήκες, οι οποίες μπορεί να μεταβληθούν πολλές φορές κατά τη διάρκεια της πέδησης (π.χ. ένας ξαφνικός άνεμος κατά τη φορά του αυτοκινήτου).

Στο παράδειγμα του φρεναρίσματος, δεν έχουμε ένα σταθερά διαμορφωμένο περιβάλλον η έστω ένα περιβάλλον που μεταβάλλεται με προβλέψιμο τρόπο. Οι συνθήκες μεταβάλλονται απρόβλεπτα η και αντιφατικά ορισμένες φορές (π.χ. αν ο δρόμος σε ένα σημείο έχει λάδια ο τρόπος του φρεναρίσματος θα πρέπει να ακολουθήσει εντελώς διαφορετική στρατηγική). Γεννιέται λοιπόν το ερώτημα: Πως καταφέρνει ο οδηγός να φρενάρει το όχημα ακόμα και σε δύσκολες γρήγορα μεταβαλλόμενες συνθήκες και μάλιστα κατά τρόπο ομαλό έτσι ώστε να μην ταλαιπωρηθούν επιβάτες και εμπορεύματα;

Ο οδηγός του αυτοκινήτου έχει συσσωρευμένη εμπειρία τόσο από την εκπαίδευση του όσο και από την εμπειρία του ως οδηγός. Η εμπειρία αυτή είναι οργανωμένη υπό μορφή κανόνων. Για παράδειγμα ο οδηγός γνωρίζει ότι αν το εμπόδιο είναι μακριά και η ταχύτητα του είναι μικρή δεν χρειάζεται να φρενάρει. Αν όμως το εμπόδιο είναι κοντά και η ταχύτητα του είναι μεγάλη τότε πρέπει να πατήσει πολύ φρένο. Οι όροι πολύ, λίγο, μικρή ταχύτητα μεγάλη ταχύτητα, μικρή απόσταση, μεγάλη απόσταση, κ.τ.λ. είναι λεκτικοί (linguistic) όροι. Ο οδηγός είναι σε θέση να εκτιμήσει πότε μια ταχύτητα είναι μικρή η μεγάλη.

Π.χ. ταχύτητα 50 km/h σε στεγνό οδόστρωμα μπορεί να θεωρηθεί μικρή. Η ίδια ταχύτητα όμως υπό διαφορετικές συνθήκες π.χ. σε κατοικημένη περιοχή και βρεγμένο οδόστρωμα είναι μάλλον μέτρια και υπό άλλες συνθήκες π.χ. το οδόστρωμα έχει λάδια είναι μάλλον μεγάλη. Είναι δυνατόν να μεταφέρουμε την εμπειρία του οδηγού σε ένα μηχανικό ελεγκτή

χρησιμοποιώντας τα κλασσικά μαθηματικά εργαλεία; Μια τέτοια μεταφορά θα δημιουργούσε κανόνες της μορφής:

Αν η ταχύτητα είναι από 10km/h έως 20km/h και η απόσταση από 80 έως 100μ τότε άσκησε στο φρένο πίεση 0.1 atm

Αν η ταχύτητα είναι από 21km/h έως 40km/h και η απόσταση από 30 έως 79μ τότε άσκησε στο φρένο πίεση 0.4 atm

Αυτή η αντιμετώπιση έχει σημαντικές αδυναμίες.
Τι θα συμβεί αν η ταχύτητα είναι 11 km/h?

Ο ελεγκτής θα συμπεριφερθεί με τον ίδιο τρόπο που θα συμπεριφερθεί στην περίπτωση που η ταχύτητα είναι 19km/h? Μια λύση στο πρόβλημα φαίνεται να είναι η μεγαλύτερη διακριτοποίηση κάθε μεταβλητής, π.χ. ανά 1km/h.
Τι θα συμβεί όμως αν έχουμε 10 μεταβλητές εισόδου και κάθε μια διακριτοποιείται σε 50 τμήματα? Θα δημιουργηθούν κανόνες. Σε κάθε περίπτωση ο στόχος που είναι η μεταφορά της εμπειρίας του οδηγού φαίνεται ότι δεν επιτυγχάνεται τόσο αποτελεσματικά με τον αυστηρά αριθμητικό (50 ..... 10 = crisp) τρόπο που επιχειρήσαμε. Το ανθρώπινο μυαλό χειρίζεται λεκτικούς όρους (linguistic terms) παρά αριθμούς και η μηχανή που φιλοδοξεί να μιμηθεί τον τρόπο λειτουργίας του θα πρέπει να το λάβει σοβαρά υπόψη. Ας δούμε πως θα αντιμετώπιζε ο άνθρωπος το παράδειγμα της πέδησης. Ένας έμπειρος οδηγός θα λειτουργούσε βάση λεκτικών κανόνων π.χ.

Αν η ταχύτητα είναι μικρή και η απόσταση είναι μέτρια τότε δύναμη πέδησης μικρή

Αν η ταχύτητα είναι μέτρια και η απόσταση είναι μέτρια τότε δύναμη πέδησης μέτρια

Αν η ταχύτητα είναι μεγάλη και η απόσταση είναι μικρή τότε δύναμη πέδησης μεγάλη κ.τ.λ.

Οι όροι μέτρια, μεγάλη, μικρή είναι οι λεκτικοί όροι οι οποίοι συμπλέκονται μεταξύ τους με συνδέσμους (connectives) και συνθέτουν κανόνες (rules). Πως όμως θα χειριστεί λεκτικούς όρους μια υπολογιστική μηχανή που χειρίζεται μόνο αριθμούς? Εδώ υπάρχει ασυμφωνία χαρακτήρων. Σε τέτοιες περιπτώσεις υποχωρεί ο ποιο λογικός που στην περίπτωση μας είναι ο άνθρωπος και δημιουργεί τη γέφυρα επικοινωνίας. Η γέφυρα αυτή είναι η ασαφής λογική. Ο Zadeh που θεωρείται ο ‘πατέρας’ της ασαφούς λογικής δήλωνε ότι η προσπάθεια μας να κατασκευάσουμε μηχανές με νοημοσύνη δεν μπορεί να προχωρήσει αν δεν βρεθεί ένας τρόπος ώστε οι μηχανές να σκέφτονται με παρόμοιους μηχανισμούς με τον άνθρωπο.

(Γενετικοί Αλγόριθμοι και Εφαρμογές (Λυκοθανάσης Σπυρίδων))

3.2 Το Ασαφές Σύνολο

Το ασαφές σύνολο είναι ίσως η βασικότερη έννοια του οικοδομήματος της ασαφούς λογικής. Ο Zadeh παρατήρησε ότι ο παραδοσιακός τρόπος περιγραφής ενός συστήματος που στηρίζεται στην αυστηρή λογική ότι μια κατάσταση μπορεί να έχει δύο μόνο μορφές ύπαρξη ή απουσία συνεπάγεται απώλεια πληροφορίας καθώς η πολυπλοκότητα του συστήματος αυξάνεται (δες το παράδειγμα με τους κανόνες).
Αν υιοθετηθεί λοιπόν ο τρόπος περιγραφής ενός συστήματος με τον αυστηρά αριθμητικό τρόπο υπάρχουν δύο επιλογές α) απλό μαθηματικό μοντέλο με απώλεια πληροφορίας ιδιαίτερα στις οριακές καταστάσεις (π.χ. ταχύτητα 21 km/h) β) μη απώλεια πληροφορίας με πολύπλοκο μαθηματικό μοντέλο (καταστάσεις). Ο 50 ..... 10 = 50 ..... 10 = Zadeh διατύπωσε αυτό το αδιέξοδο με την περίφημη Αρχή της Ασυμβατότητας: 5 ...καθώς η πολυπλοκότητα ενός συστήματος αυξάνεται, η ικανότητα για ακριβείς και ταυτόχρονα σημαντικές δηλώσεις που αφορούν τη συμπεριφορά του μειώνεται, και πέρα από ένα σημείο η ακρίβεια και η σημαντικότητα αποτελούν σχεδόν αμοιβαία αποκλειόμενα χαρακτηριστικά.

Ο Zadeh συνειδητοποίησε λοιπόν ότι ο πυρήνας του αδιεξόδου είναι ο δυαδικός τρόπος αναπαράστασης της πληροφορίας κατά τον οποίο μια τιμή μιας μεταβλητής είτε ανήκει είτε δεν ανήκει σε ένα υποσύνολο του πεδίου ορισμού της. Πρότεινε λοιπόν ένα διευρυμένο τρόπο αναπαράστασης όπου μια τιμή ανήκει ταυτόχρονα σε πολλά υποσύνολα, στο κάθε ένα με ένα βαθμό συμμετοχής. Κάθε τέτοιο υποσύνολο που περιλαμβάνει στοιχεία όπου κάθε ένα έχει ένα βαθμό συμμετοχής είναι το ασαφές σύνολο.

Ας θεωρήσουμε τη μεταβλητή που αντιπροσωπεύει την ταχύτητα ενός αυτοκινήτου σε ένα αυτοκινητόδρομο με πεδίο ορισμού από 0 km/h έως 120 km/h.
Κάνουμε τις έξης ερωτήσεις στον οδηγό:

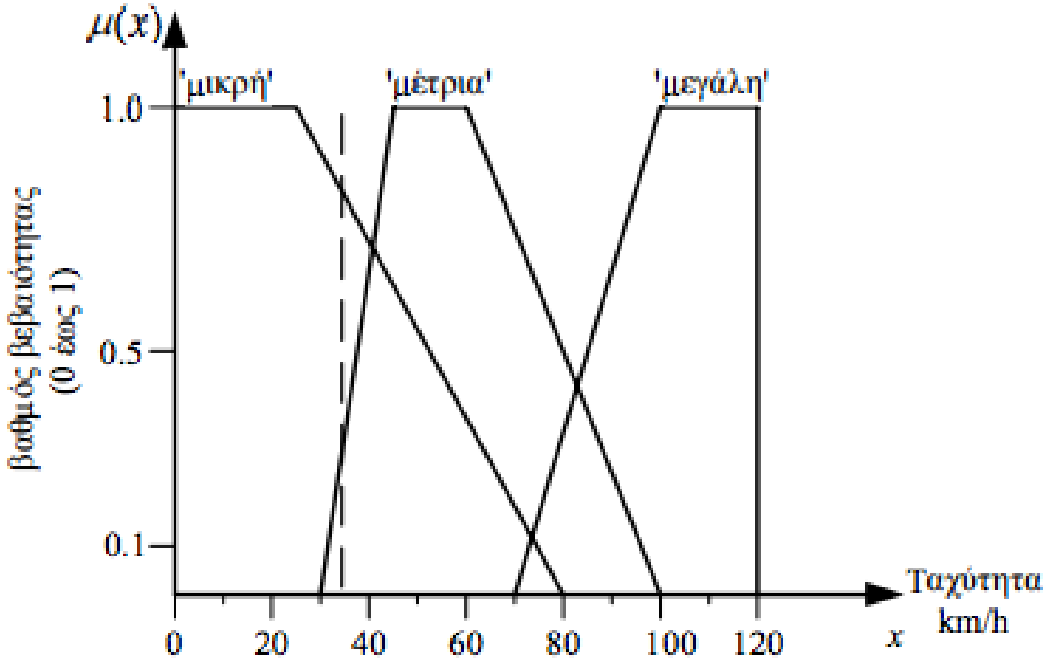
ΕΡ: Πόσο σίγουρος είσαι ότι η ταχύτητα 15 km/h είναι μικρή?
ΑΠ: 100%.

ΕΡ: Πόσο σίγουρος είσαι ότι η ταχύτητα 20 km/h είναι μικρή?
ΑΠ: 100%.

ΕΡ: Πόσο σίγουρος είσαι ότι η ταχύτητα 40 km/h είναι μικρή?
ΑΠ: 90% είναι και λίγο μέτρια.

ΕΡ: Πόσο σίγουρος είσαι ότι η ταχύτητα 60 km/h είναι μικρή?
ΑΠ: λίγο ένα 10-20% διότι 60 km/h ταχύτητα είμαι πιο σίγουρος ότι είναι μέτρια.

Αν κάνουμε παρόμοιες ερωτήσεις για μέτρια και για μεγάλη ταχύτητα και παραστήσουμε γραφικά τη βεβαιότητα από 0 έως 1 του οδηγού συναρτήσει της ταχύτητας θα πάρουμε το παρακάτω διάγραμμα (Εικόνα 7**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**)



Εικόνα 7: Διάγραμμα Πεδίων Ταχύτητας – Βαθμού Βεβαιότητας *(Γενετικοί Αλγόριθμοι και Εφαρμογές (Λυκοθανάσης Σπυρίδων))*

Στο παραπάνω διάγραμμα παρατηρούμε μια λεκτική αναπαράσταση της ταχύτητας. Η ταχύτητα έχει 'διαμεριστεί' όπως λέμε σε λεκτικούς (linguistic) όρους.

Από το σημείο αυτό και μετά μπορούμε να μιλάμε για την ταχύτητα με λεκτικούς αντί αυστηρά αριθμητικούς όρους. Ας δούμε τι θα λέγαμε για μια ταχύτητα 32 km/h.
Η ταχύτητα αυτή είναι μικρή με βαθμό 6 βεβαιότητας 0.8 και μέτρια με βαθμό βεβαιότητας 0.3. Ο βαθμός αυτός βεβαιότητας ονομάζεται βαθμός συμμετοχής μ(x) μιας τιμής στο αντίστοιχο ασαφές σύνολο που εκφράζεται από την συνάρτηση x μ(x).
Η περιγραφή μιας μεταβλητής x με λεκτικούς όρους ονομάζεται διαμερισμός της μεταβλητής και η περιγραφή μιας αυστηρά αριθμητικής τιμής με λεκτικούς όρους όπως για παράδειγμα η ταχύτητα των 32 km/h ονομάζεται 'ασαφοποίηση' (fuzzyfication) της crisp τιμής. Ο αναγνώστης μπορεί να πειραματιστεί και να ασαφοποιήσει την ταχύτητα των 78 km/h. Στο σημείο αυτό θα πρέπει να τονιστεί κάτι που συγχέεται συχνά ακόμα και από έμπειρους χρήστες της ασαφούς λογικής και αποτελεί το σημείο στο οποίο διαχωρίζεται η ασαφής λογική από τη στατιστική ανάλυση.

Ένα ασαφές σύνολο εκφράζει κατανομή δυνατότητας (possibility distribution) και ένας βαθμός συμμετοχής μιας τιμής σε ένα ασαφές σύνολο αποτελεί το βαθμό βεβαιότητας (degree of certainty) ότι η πρόταση που διατυπώνουμε είναι αληθής (π.χ η ταχύτητα των 32 km/h είναι μικρή. Βεβαιότητα: 40%) και όχι κατανομή πιθανότητας (probability distribution).

(Γενετικοί Αλγόριθμοι και Εφαρμογές (Λυκοθανάσης Σπυρίδων))

3.3 Ο ασαφής Κανόνας

Εννοιολογικά, ο ασαφής κανόνας είναι ένας μηχανισμός αναπαράστασης της γνώσης, ο οποίος προσιδιάζει στον ανθρώπινο τρόπο σκέψης. Τα ασαφή σύνολα που εκφράζουν λεκτικούς όρους συνδυάζονται μεταξύ τους και δημιουργούν ασαφείς κανόνες που αναπαριστούν τη γνώση που έχουμε για το σύστημα.
Ένας ασαφής κανόνας αποτελείται από δύο βασικά μέρη

α) το τμήμα υπόθεσης (premise part) και
β) το τμήμα απόδοσης η απόφασης (consequent part).

Ένας απλός κανόνας είναι της μορφής:

If x is A then y is B (2-11)

Το τμήμα if x is A είναι το τμήμα υπόθεσης και το τμήμα then y is B το τμήμα απόφασης η συμπεράσματος. Όπου A και B ασαφή σύνολα. x είναι η τιμή μιας μεταβλητής εισόδου, η οποία ασαφοποιείται (fuzzyfication), δηλαδή αποκτά ένα βαθμό συμμετοχής στο ασαφές σύνολο A. y είναι η έξοδος του συστήματος που εκφράζει την απόφαση του κανόνα και παρέχεται από το μηχανισμό του συμπεράσματος (inference) σε ασαφή μορφή.
Στη συνέχεια το ασαφές συμπέρασμα από-ασαφοποιείται με το μηχανισμό της αποασαφοποίησης (defuzzyfication), και προκύπτει μια crisp τιμή που είναι το τελικό αριθμητικό συμπέρασμα που μπορεί να χειριστεί η υπολογιστική μηχανή η ένα αισθητήριο.
Ένας κανόνας μπορεί να πάρει διάφορες μορφές εκτός από αυτή της εξίσωσης (2-11). Ο κανόνας της (2-11), του οποίου η έξοδος είναι ένα ασαφές σύνολο ονομάζεται κανόνας τύπου mamdani προς τιμή του Ebrahim Mamdani που ήταν από τους πρώτους που εφάρμοσε την ασαφή λογική για να κατασκευάσει ένα ασαφές σύστημα αυτόματου έλεγχου της ταχύτητας μιας ατμομηχανής.
Άλλοι κύριοι χαρακτηριστικοί τύποι κανόνων είναι της μορφής:

If x is A then y is c (2-12)

Όπου το c είναι αριθμός η μπορεί να θεωρηθεί και crisp ασαφές σύνολο και:

If x is A then y is c0+c1 x (2-13)

Όπου η σχέση (2-12) προτάθηκε από τους Sugeno-Takagi και η επέκταση της (2-13) από τους Takagi-Sugeno-Kang. Ο ασαφής κανόνας (2-13) είναι ένας από τους κυριότερους τύπους ασαφούς κανόνα και χρησιμοποιείται ευρύτατα σε εφαρμογές ανάπτυξης ασαφών συστημάτων. Είναι γνωστός ως κανόνας T-S-K από τα αρχικά των ερευνητών TakagiSugeno-Kang.

(Γενετικοί Αλγόριθμοι και Εφαρμογές (Λυκοθανάσης Σπυρίδων))

3.4 Ασαφής Ελεγκτής

Τα βασικά δομικά στοιχεία ενός ασαφούς ελεγκτή είναι:

- Η βάση γνώσης, στην οποία είναι αποθηκευμένοι οι κανόνες για τον έλεγχο της διαδικασίας.
- Τα ασαφή σύνολα, που χρησιμοποιούνται για να αναπαραστήσουν τις μεταβλητές εισόδου και τους λεκτικούς όρους.
- Ο ασαφοποιητής, ο οποίος μετατρέπει τις πραγματικές τιμές της εισόδου σε ασαφή σύνολα.
- Ο μηχανισμός συμπερασμού, ο οποίος επεξεργάζεται τις εξόδου του ασαφοποιητή και με χρήση της βάσης γνώσης εξάγει τα ασαφή σύνολα των συμπερασμάτων.
- Ο αποσαφοποιητής, που μετατρέπει τα συμπεράσματα που εξάγει ο μηχανισμός σεμπερασμού σε πραγματικούς αριθμούς για να μπορεί να γίνει η μετάδοση της δράσης ελέγχου στην διαδικασία.

Οι είσοδοι σε έναν ασαφή ελεγκτή είναι σήματα (δηλαδή σαφείς μεταβλητές) και επομένως πρέπει ο σχεδιαστής ενός ασαφούς ελεγκτή να κάνει τα ακόλουθα βήματα:

- Λεκτική κατανομή των εισόδων: Ο σχεδιαστής πρέπει να αναπαραστήσει τις μεταβλητές εισόδου και εξόδου με τους λεκτικούς όρους.
- Διατύπωση των κανόνων: Τα ασαφή σύνολα μετά την κατανομή των εισόδων και εξόδων αποθηκεύονται υπό τη μορφή συναρτήσεων συμμετοχής στον υπολογιστή και έπειτα ακολουθεί η διατύπωση των κανόνων.
- Καθορισμό του τύπου της ασαφούς συνεπαγωγής: Μετά τη διατύπωση των κανόνων είναι απαραίτητος ο καθορισμός του ασαφούς τύπου συνεπαγωγής. Οι πιο γνωστοί τύποι ασαφούς συνεπαγωγής είναι:
    - του Mamdani, όπου χρησιμοποιείται ο τελεστής max-min, ο οποίος λαμβάνει το μικρότερο από τους βαθμούς συμμετοχής των ασαφοποιημένων τιμών και παράγει το βαθμό εκπλήρωσης (degree of fulfillment) του κάθε κανόνα. Ο βαθμός εκπλήρωσης του κανόνα δηλώνει τη βαρύτητα που έχει το αποτέλεσμα του κανόνα.
    - του Larsen, όπου χρησιμοποιείται ο τελεστής max-product, ο οποίος πολλαπλασιάζοντας τους βαθμούς συμμετοχής των ασαφοποιημένων τιμών υπολογίζει το βαθμό εκπλήρωσης του κανόνα.
- Από-ασαφοποίηση: Η από-ασαφοποίηση παράγει μία αυστηρή ή crisp τιμή από ένα ασαφές σύνολο. Είναι με λίγα λόγια, η αντίθετη διαδικασία από την ασαφοποίηση. Οι μέθοδοι από-ασαφοποίησης είναι:
    - Από-ασαφοποίηση κεντρικής τιμής (Centroid defuzzycation ή center of area ή COA), όπου υπολογίζεται το κέντρο βάρους της κατανομής του ασαφούς συνόλου της εξόδου:

$$x = \frac{\int x \cdot \mu(x)\, dx}{\int \mu(x)\, dx}$$

- Από-ασαφοποίηση μέσου όρου των μεγίστων (Mean of Maxima ή MOM), όπου υπολογίζεται ο μέσος όρος των τιμών εξόδου που έχουν τον μεγαλύτερο βαθμό συμμετοχής:

$$X = \left(\frac{1}{m}\right) * \sum^{m} \max(\mu(x))$$

- Από-ασαφοποίηση μικρότερου από τους μέγιστους (Smallest of maxima ή SOM), όπου υπολογίζεται από τις μέγιστες τιμές εξόδου εκείνη που έχει το μικρότερο βαθμό συμμετοχής
- Από-ασαφοποίηση μεγαλύτερου από τους μέγιστους (Largest of maxima ή LOM), όπου υπολογίζεται από τις μέγιστες τιμές εξόδου εκείνη που έχει το μεγαλύτερο βαθμό συμμετοχής

Η μέθοδος που χρησιμοποιείται περισσότερο είναι η μέθοδος από- ασαφοποίησης της κεντρικής τιμής ή κεντρώου (Centroid ή COA), εξαιτίας της ικανότητάς της να παρουσιάζει σε σχέση με τις άλλες μεθόδους το πιο μικρό σφάλμα.

(Εισαγωγή στην Ασαφή Λογική Δρ. Κυριάκος Δεληπαράσχος)

3.5 Συναρτήσεις Συμμετοχής

Υπάρχουν διάφοροι τύποι συναρτήσεων συμμετοχής (Membership functions ή MF's) που αναπαριστούν τα ασαφή σύνολα όπως είναι η τριγωνική μορφή (triangular mf), η τραπεζοειδή (trapezoidal mf), η καμπανοειδή (generalize bell mf ή gbell mf), η γκαουσιανή (gaussian mf), η μορφή s (s mf), η μορφή pi (pi mf), η μορφή z (z mf), η σιγμοειδή (sigmoidal mf) ή ακόμα και μια συγκεκριμένη μαθηματική τιμή.

- Η τριγωνική συνάρτηση συμμετοχής (triangular mf) χαρακτηρίζεται από τις τρεις παραμέτρους {a, b, c}, ως εξής:

$$Triangle(x, a, b, c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}, 0)$$

- Η τραπεζοειδής συνάρτηση συμμετοχής (trapezoidal mf) χαρακτηρίζεται από τις τέσσερις παραμέτρους {a, b, c, d}, ως εξής:

$$Trapezoid(x, a, b, c, d) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}),0)$$

- Η καμπανοειδής συνάρτηση συμμετοχής (generalize bell mf ή gbell mf) χαρακτηρίζεται από τις τρεις παραμέτρους {a, b, c}, ως εξής:

$$bell(x, a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|}$$

- Η γκαουσιανή συνάρτηση συμμετοχής (gaussian mf) χαρακτηρίζεται από τις δύο παραμέτρους {σ, c}, όπου το σ καθορίζει το πλάτος της συνάρτησης συμμετοχής (mf) και το c αναπαριστά το κέντρο της mf:

$$gaussian(x, \sigma, c) = e^{-(\frac{x-c}{\sigma})}$$

- Η σιγμοειδή συνάρτηση συμμετοχής (sigmoidal mf) χαρακτηρίζεται από τις δύο παραμέτρους {α, c}, ως εξής:

$$sigmoid(x, a, c) = 1/(1 + e^{-a*(x-c)})$$

Οι συναρτήσεις συμμετοχής πέρα από το ότι οριοθετούν με την λογική μας τα πεδία, αντιστοιχίζουν τιμές στα πεδία αυτά και παίζουν πολύ βασικό ρόλο στην συμπεριφορά του ελεγκτή μας.
Επίσης έχουν αναπτυχθεί και τεχνικές, οι οποίες δύνανται να βελτιώνουν τέτοιους ελεγκτές, όπως η μέθοδος γενετικών αλγορίθμων, η οποία για παραδειγμά είναι συμβατή με τριγωνικές και τραπεζοειδής συναρτήσεις συμμετοχής και όχι με γκαουσσιανές, καμπανοειδής ή σιγμοειδείς, γεγονός που πρέπει να ληφθεί σοβαρά υπόψην στην σχεδίαση των ελεγκτών.

(Εισαγωγή στην Ασαφή Λογική Δρ. Κυριάκος Δεληπαράσχος)

Κεφάλαιο 4: Λογισμικό[1]

3.6 V-REP - MATLAB

Το V-REP όπως είπαμε και στο κεφάλαιο 2 (2.3), είναι ένα πρόγραμμα προσομοίωσης ρομποτικών συστημάτων το οποίο θα χρησιμοποιηθεί εδώ σε συνεργασία με ένα πρόγραμμα μαθηματικών υπολογισμών (MATLAB). Το V-REP διαθέτει προγραμματιστικό κομμάτι, όμως εδώ θα χρησιμοποιήσουμε το MATLAB, γιατί έχει πιο γνωστή γλώσσα προγραμματισμού, γεγονός που θα διευκολύνει την προσομοίωση.

Το MATLAB είναι ένα πρόγραμμα που παρέχει στο χρήστη ένα περιβάλλον αριθμητικής υπολογιστικής και μια προγραμματιστική γλώσσα τέταρτης γενιάς. Αποθηκεύει και κάνει τις πράξεις με βάση την άλγεβρα μητρών. Χρησιμοποιείται κατά κύριο λόγο για την επίλυση μαθηματικών προβλημάτων, ωστόσο είναι πολύ "ισχυρό" και μπορεί να χρησιμοποιηθεί και για προγραμματισμό καθώς περιέχει εντολές από την C++ όπως την while, την switch και την if. Στον τομέα των γραφικών όσον αφορά τον μαθηματικό κλάδο μπορεί να υλοποιήσει συναρτήσεις πραγματικές, μιγαδικές, πεπλεγμένες συναρτήσεις δύο μεταβλητών και άλλες. Όσο αφορά τον στατιστικό κλάδο μπορεί να πραγματοποιήσει ιστογράμματα, τομεογράμματα, ραβδοδιαγράμματα, εμβαδογράμματα και άλλα.
Το προγραμματιστικό κομμάτι του MATLAB μπορεί να αναλάβει πολύπλοκες διαδικασίες εκτελώντας πολύπλοκα script, συνδυάζοντας απλότητα στον τρόπο σύνταξης τους και συνδεσιμότητα με πάρα πολλές άλλες πλατφόρμες όπως είδαμε παραπάνω.

Συνδέοντας λοιπόν τα δύο αυτά προγράμματα μεταξύ τους, θα μπορέσουμε να κάνουμε όλους τα μαθηματικούς υπολογισμούς που απαιτούνται, στην γλώσσα προγραμματισμού που είναι γνωστή και εύκολη στη σύνταξη.

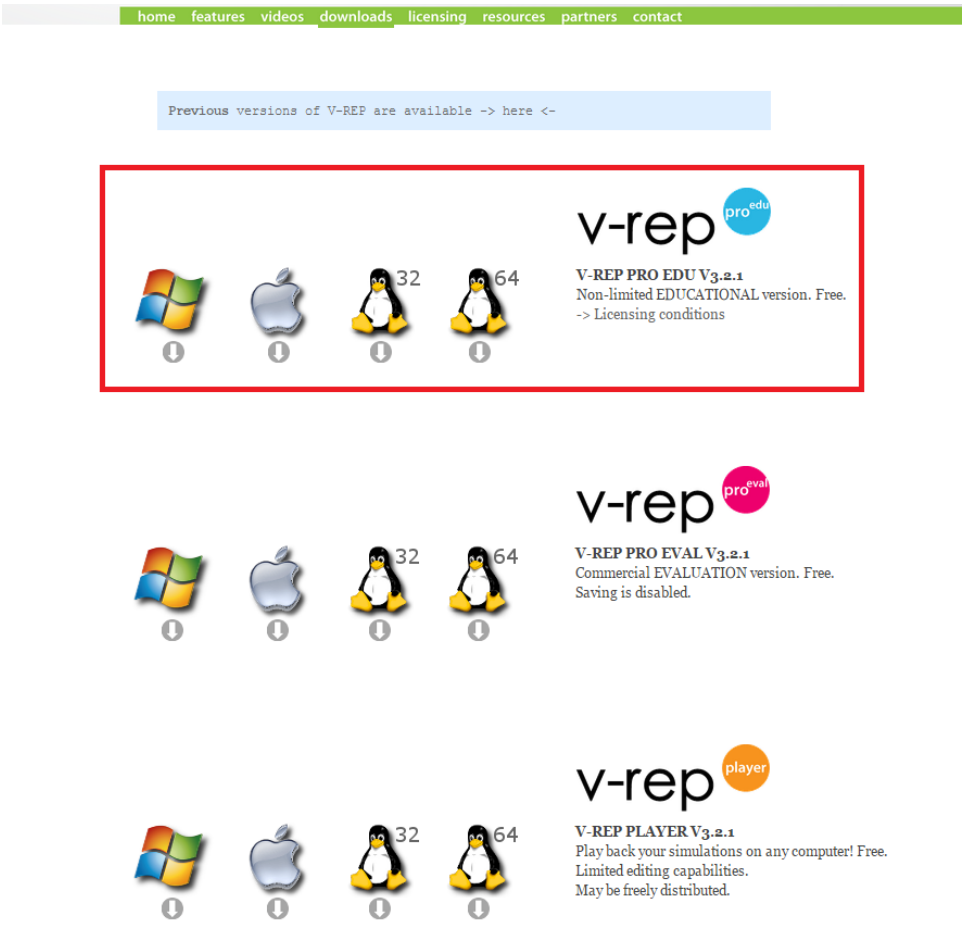
3.6.1 Σύνδεση V-REP – MATLAB

Για να πραγματοποιήσουμε τη σύνδεση θα πρέπει να έχουμε ήδη εγκατεστημένο το MATLAB στον υπολογιστή μας, κάτι που είναι πολύ εύκολο και δεν χρήζει ειδικών οδηγιών.
Έχοντας λοιπόν ήδη εγκατεστημένο το Matlab στον υπολογιστή, προχωράμε στην εγκατάσταση του V-Rep, πηγαίνοντας στην ιστοσελίδα v-rep.eu. Έπειτα πατάμε την καρτέλα downloads (Εικόνα 8).

[1] Η διαδικασία που περιγράφεται μπορεί να εκτελεστεί από συμπληρωματικό λογισμικό που συνοδεύει την παρούσα εργασία.

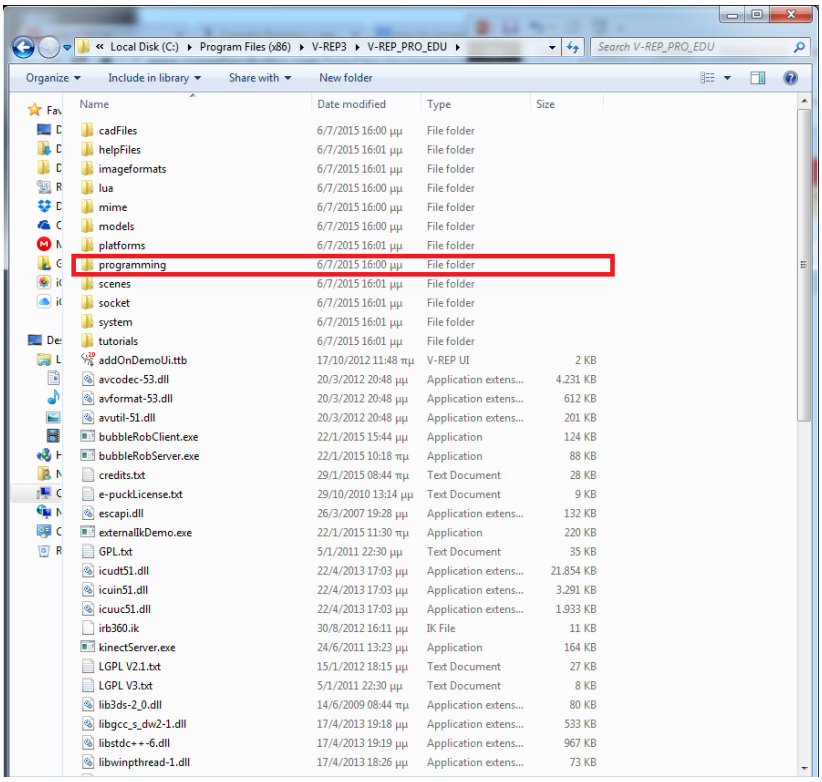
Εικόνα 8: Ιστοσελίδα v-rep.eu

Επιλέγουμε το λειτουργικό σύστημα του υπολογιστή μας, ώστε να κατεβάσουμε την αντίστοιχη συμβατή εγκατάσταση (Εικόνα 9).



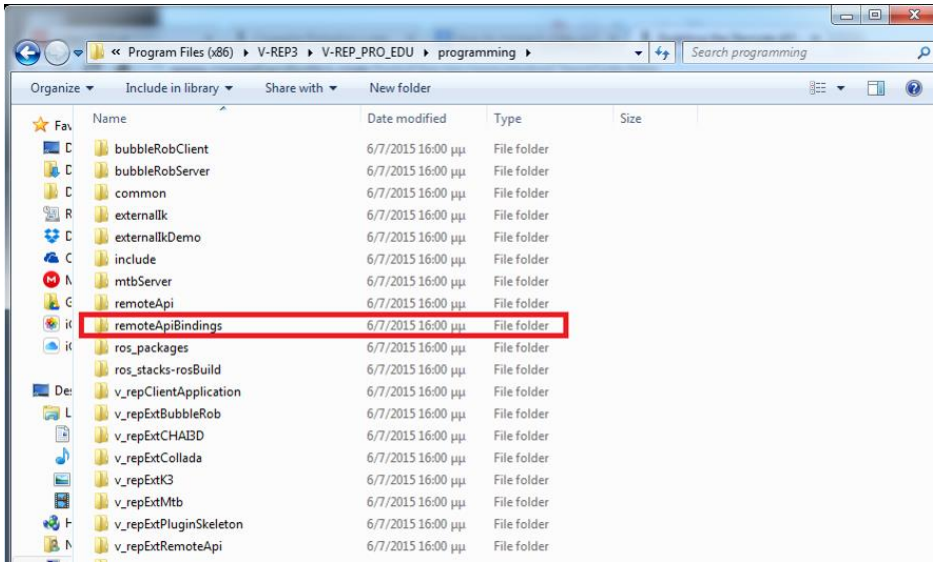
Εικόνα 9: Ιστοσελίδα v-rep.eu Downloads

Μόλις κατέβει το αρχείο και το εγκαταστήσουμε στον υπολογιστή μας, που είναι πολύ απλή διαδικασία, προχωράμε στην σύνδεση του με το matlab. Ανάλογα με το λειτουργικό σύστημα η διαδικασία είναι λίγο διαφορετική. Πηγαίνουμε στο φάκελο εγκατάστασης του V-Rep, ο οποίος στα Windows είναι C:\Program Files (x86)\V-REP_PRO_EDU, ενώ για MAC OS X είναι στο ~\Applications\V-REP_PRO_EDU και τον ανοίγουμε. Βρίσκουμε εκεί ένα φάκελο με όνομα programming και τον ανοίγουμε (Εικόνα 10).



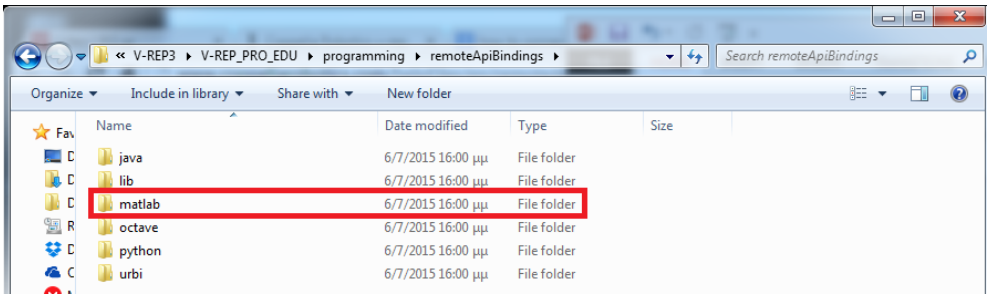
Εικόνα 10: Φάκελος Εγκατάστασης V-REP στον Υπολογιστή

Ανοίγουμε τον φάκελο remoteApiBindings (Εικόνα 11).



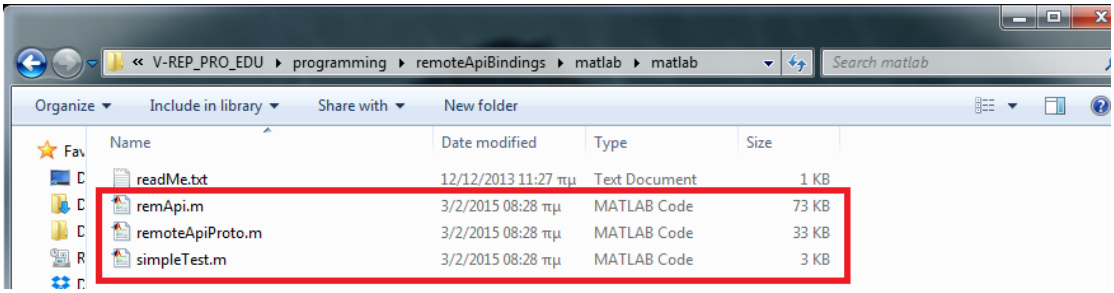
Εικόνα 11: Φάκελος programming στον φάκελο V-REP

Ανοίγουμε το φάκελο matlab, και μέσα στο φάκελο ξανανοίγουμε το φάκελο matlab (Εικόνα 12).



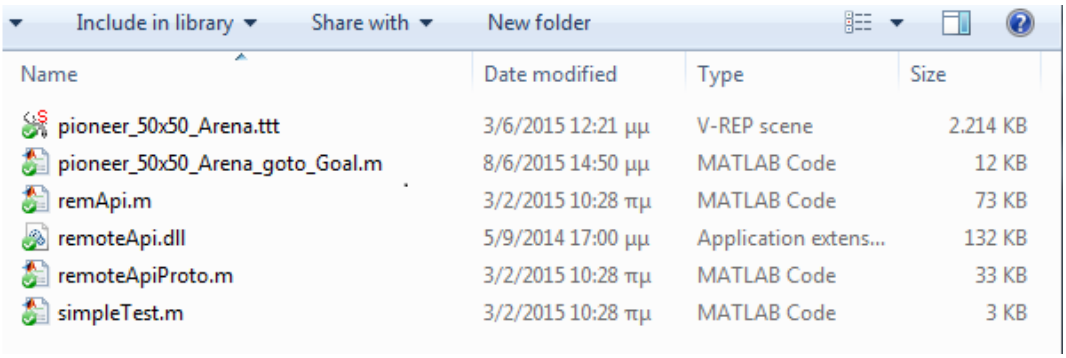
Εικόνα 12: Φάκελος matlab στον φάκελο programming

Αντιγράφουμε τα αρχεία remApi.m , remoteApiProto.m , simpleTest.m σε ένα φάκελο που θα δημιουργήσουμε στην τοποθεσία C:\ με όνομα trs, ή για MAC ~/ (Εικόνα 13).



Εικόνα 13: Αρχεία προς αντιγραφή από το φάκελο matlab

Έπειτα πηγαίνουμε στο φάκελο που παρατίθεται μαζί με την παρόν έγγραφο και μέσα εκεί ανοίγουμε το φάκελο Windows ή MAC ανάλογα με το λειτουργικό σύστημα στο οποίο βρίσκεται ο υπολογιστής, και ανοίγουμε το φάκελο trs. Αντιγράφουμε όλα τα περιεχόμενα του φακέλου αυτού στο φάκελο που δημιουργήσαμε λίγο πριν (C:\trs ή ~/trs) (Εικόνα 14).

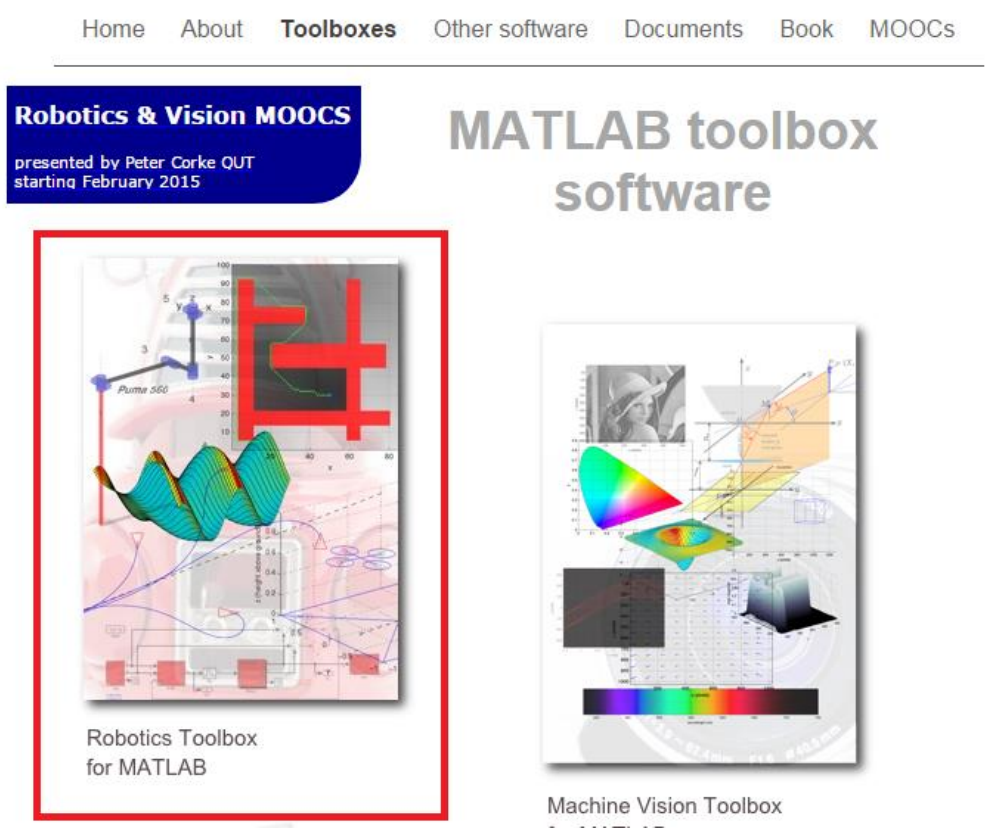


Εικόνα 14: Φάκελος Windows (που διατίθεται στο μαζί με την παρούσα εργασία).

Το επόμενο βήμα είναι να κατεβάσουμε και να εγκαταστήσουμε το robotics toolbox που είναι απαραίτητο για τις προσομοιώσεις μας μέσω matlab. Για να γίνει αυτό, πηγαίνουμ στη σελίδα petercorke.com και επιλέγουμε την καρτέλα toolboxes. (Εικόνα 15).

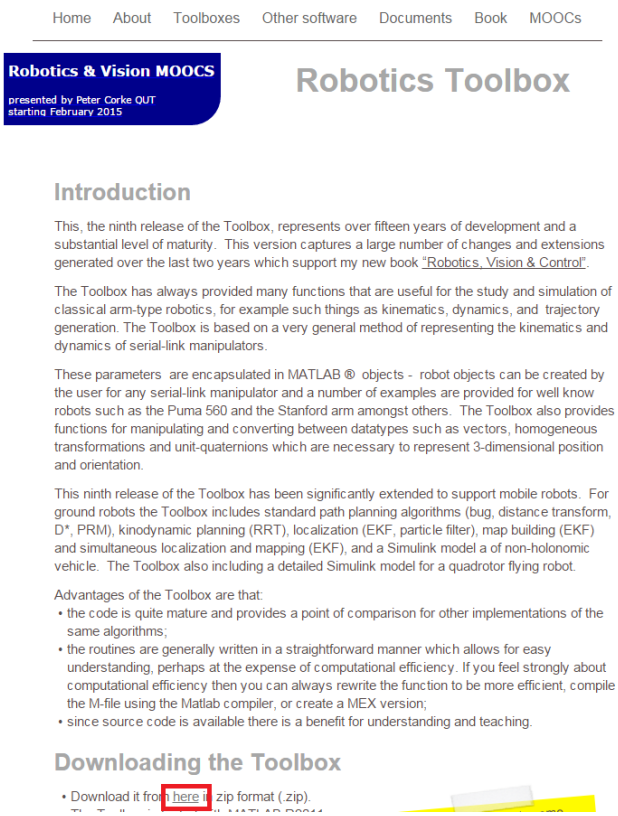
Εικόνα 15: Ιστοσελίδα petercorke.com

Επιλέγουμε το Robotics Toolbox for Matlab. (Εικόνα 16).



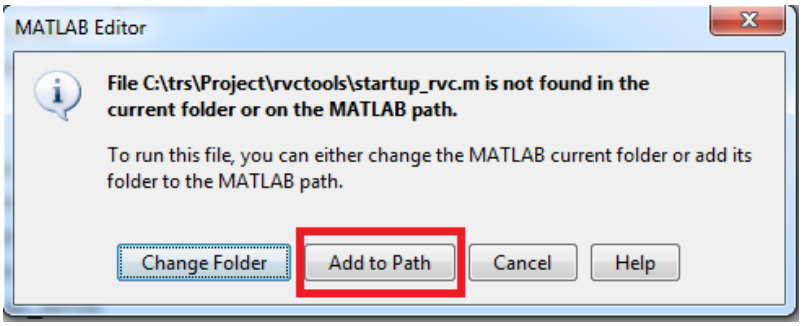
Εικόνα 16: Ιστοσελίδα petercorke.com Toolboxes

Λίγο πιο κάτω στη σελίδα έχει ένα link στην περιοχή Downloading the Toolbox (Εικόνα 17).



Εικόνα 17: Ιστοσελίδα petercorke.com Robotics Toolbox

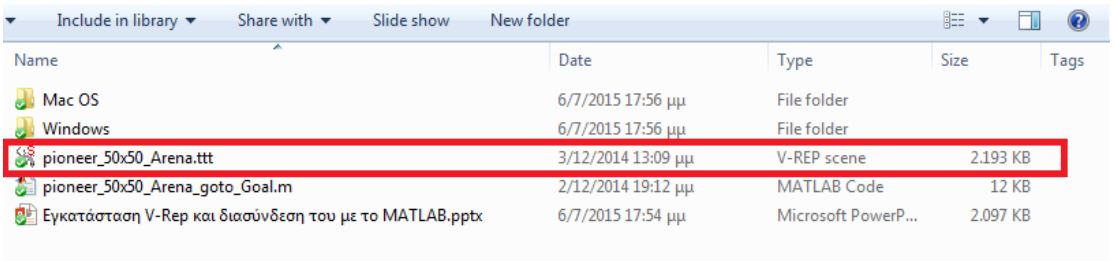
Μετά συμπληρώνουμε τα στοιχεία μας (χώρα, πανεπιστήμιο) και επιλέγουμε αν είμαστε φοιτητής, καθηγητής ή ερευνητικό πρόγραμμα και πατάμε Send. Επιλέγουμε την πιο πρόσφατη έκδοση και περιμένουμε να κατέβει.Μόλις κατέβει αποσυμπιέζουμε το zip μέσα στον φάκελο trs που φτιάξαμε. Ανοίγουμε τον φάκελο rvctools που μόλις αποσυμπιέσαμε και τρέχουμε το startup_rvc.m. Όταν το τρέξουμε στο matlab θα μας βγάλει μήνυμα αλλαγής φακέλου. Εκεί εμείς πατάμε Add To Path.(Εικόνα 18).



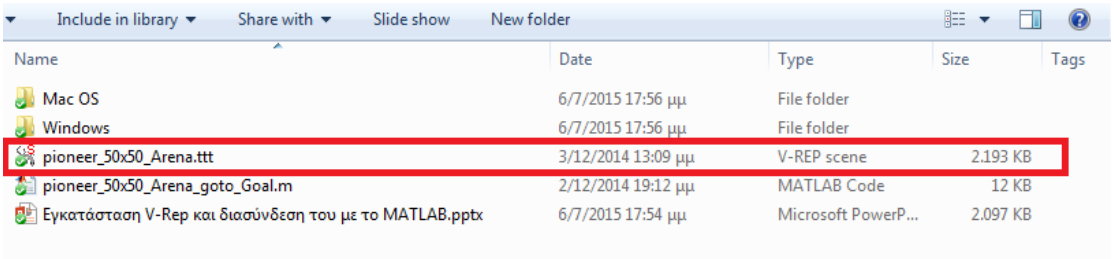
Εικόνα 18: Παράθυρο Αλλαγής Φακέλου του MATLAB

Μόλις τελειώσουμε όλα τα παραπάνω, πρέπει να επιβεβαιώσουμε αν τα κάναμε όλα σωστά. Έτσι λοιπόν πηγαίνουμε στο V-REP και στο μενού File επιλέγουμε Open Scene.

‘Επιλέγουμε το αρχείο pioneer_50x50_Arena.ttt από τον φάκελο trs που φτιάξαμε πριν. (

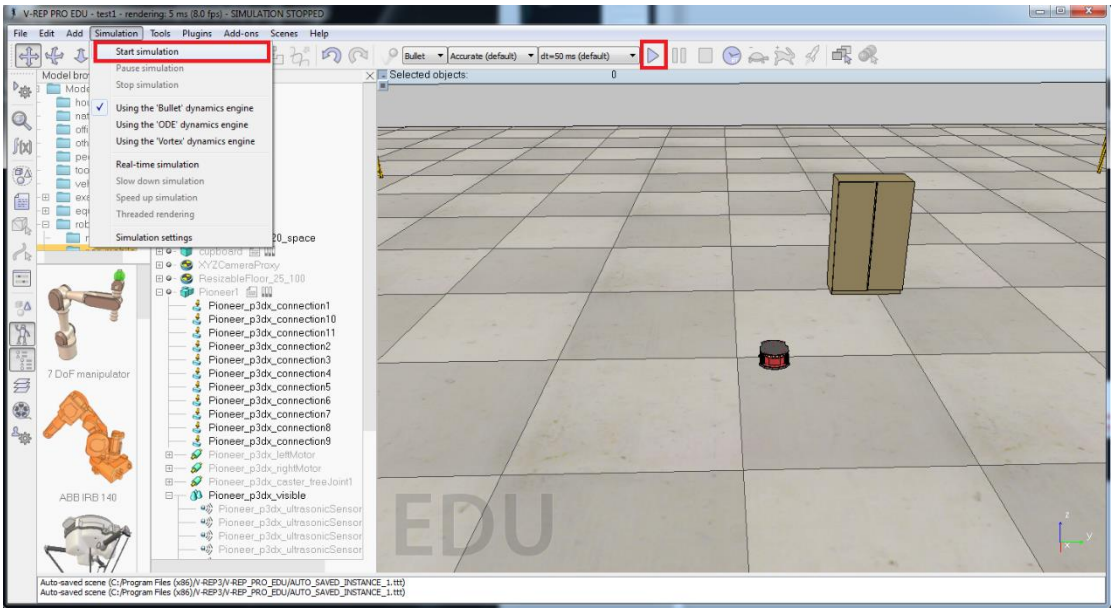


Εικόνα 19)



Εικόνα 19: Αρχεία pioneer_50x50_Arena.ttt στον φάκελο trs

Πατάμε Start Simulation ή από το menu Simulation είτε πατάμε το Play που βρίσκεται πάνω στην μπάρα συντομεύσεων του V-REP (Εικόνα 20).



Εικόνα 20: Σκηνή pioneer_50x50_Arena.ttt στο V-REP

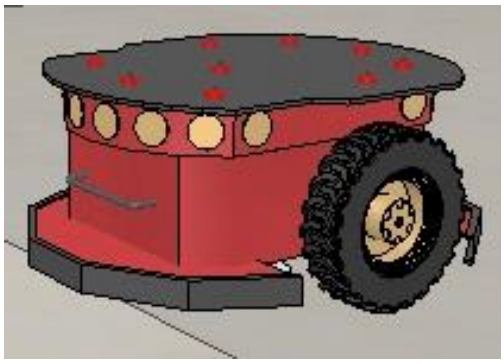
Πηγαίνουμε τώρα στο φάκελο trs που δημιουργήσαμε πριν και ανοίγουμε το αρχείο pioneer_50x50_Arena_goto_Goal.m και το τρέχουμε στο matlab. Πατάμε Change Folder στο μήνυμα που εμφανίζεται. Αν το robot στο V-REP κουνιέται, τότε σημαίνει πως η διαδικασία ήταν επιτυχής. Αν το robot δεν κουνιέται, τότε το matlab θα εμφανίσει error και θα πρέπει να ελέγξουμε τι από τα παραπάνω δεν έγινε σωστά. Εφόσον όμως έγιναν όλα όπως πρέπει

τότε έχουμε συνδέσει επιτυχώς τα δύο προγράμματα μεταξύ τους και μπορούμε πλέον να προχωρήσουμε στην προσομοίωση.

4. Κεφάλαιο 5: Κίνηση με Ελεγκτή Ασαφούς Λογικής

4.1 Σύντομη Περιγραφή του Οχήματος Μας

Φτιάχνοντας τον κώδικα με τον οποίο θα ελένχουμε την κίνηση ενός ρομπότ, θα πρέπει να γνωρίζουμε τα ιδιαίτερα χαρακτηριστικά του. Υπάρχουν έντροχα ρομπότ που χρησιμοποιούν κινητήρες για την κίνηση τους όμως στρίβουν με χρήση τρίτου, υπάρχουν άλλα που χρησιμοποιούν τέσσερις κινητήρες και στρίβουν με τους δυο μπροστινούς. Εδώ χρησιμοποιήθηκε για απλότητα ελέγχου το Pioneer, λόγω του ότι κινείται με χρήση δύο μόνο κινητήρων. Αυτό επιτρέπει τον έλεγχο της διεύθυνσης του απλά ελέγχοντας την ταχύτητα σε έναν από τους δύο κινητήρες.
Επίσης διαθέτει αρκετούς αισθητήρες με τους οποίους μπορεί να ελέγχει το περιβάλλον γύρω του για εμπόδια, όμως υπερηχητικοί αισθητήρες και οπτικό αισθητήρα εμπρός για αναγνώριση αντικειμένων.



Εικόνα 21: Pioneer Robot Model στο V-REP

4.2 Απλό Παράδειγμα Ελέγχου Ρομπότ με χρήση V-REP - MATLAB

Υπάρχουν κάποιοι κανόνες που πρέπει να τηρούμε για να μπορέσει να γίνει η σύνδεση, όπως φαίνεται κι από τον κώδικα που περιγράφηκε στη προηγούμενη παράγραφο. Στο φάκελο trs που έχει αντιγραφεί στο υπολογιστή, υπάρχει επίσης ένα script με όνομα test.m . Το επιλέγουμε και το ανοίγουμε με το MATLAB. Υπάρχει λοιπόν το script στο παράρτημα, μαζί με σχόλια δίπλα σε κάθε εντολή, ώστε να κατανοηθεί πλήρως:

Η σειρά με τη οποία συντάσσεται είναι, όπως φαίνεται στο πρώτο (Εικόνα 38) και δεύτερο κομμάτι του κώδικα (Εικόνα 39) παραπάνω είναι η εξής:

1. Φόρτωση του Robotics Toolbox
2. Ρουτίνα V-REP για έναρξη ζεύξης και έπειτα προσομοίωσης
3. Αναγνώριση όλων των αντικειμένων που θέλουμε να ελένξουμε, όπου περιλαμβάνεται το ρομπότ, όμως και τα επί μέρους κινούμενα μέρη του, όπως κινητήρες , τυχών βραχίονες κτλ. και τα αισθητήρια του.

Στο τρίτο κομμάτι το κώδικα (Εικόνα 40) ακολουθεί η ρουτίνα κίνησης που περιλαμβάνει:

- Λήψη Προσανατολισμού και Θέσης
- Μέτρηση απόστασης από το στόχο
- Υπολογισμός Σφάλματος Προσανατολισμού σε σχέση με το στόχο
- Έλεγχος εμπρόσθιου αισθητήρα για εμπόδια

Το τελευταίο κομμάτι του κώδικα (Εικόνα 41) ενσωματώνει το υπόλοιπο της ρουτίνας κίνησης, που περιλαμβάνει:

- Αλλαγή στόχου εφόσον η απόσταση είναι πολύ μικρή από τον προηγούμενο
- Υπολογισμός Νέας Ταχύτητας Κινητήρων
- Εφαρμογή Νέας ταχύτητας στους κινητήρες του ρομπότ

Επίσης ακινητοποιεί το ρομπότ και τερματίζει τη προσομοίωση.

Το παραπάνω script είναι μια απλή ρουτίνα προσομοίωσης καθώς εκτελέι απλή κίνηση σε ορισμένα σημεία στο χώρο, με έναν απλό εμπρόσθιο έλεγχο για εμπόδια που δεν είναι και πολύ αποτελεσματικός, όμως μπαίνει εδώ για να κατανοηθεί η φιλοσοφία με την οποία δομείται ένα τέτοιο πρόγραμμα.

4.3 Ελεγκτής Ασαφούς Λογικής

4.3.1 Δημιουργία Ελεγκτή Ασαφούς Λογικής στο MATLAB

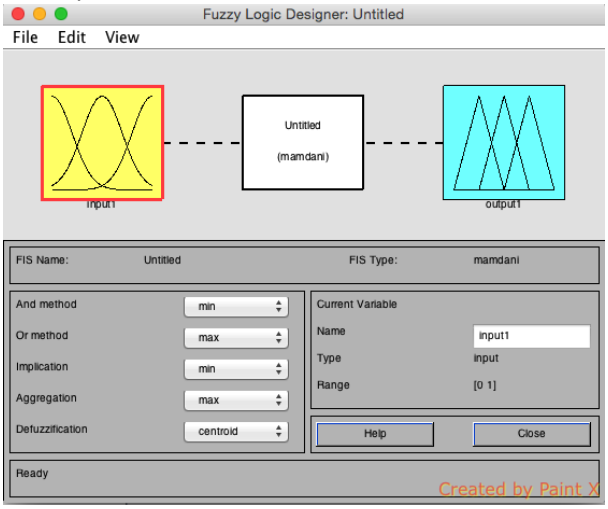
Το MATLAB μας δίνει τη δυνατότητα να δημιουργήσουμε και να επεξεργαζόμαστε ελεγκτές ασαφούς λογικής πολύ εύκολα καθώς μας παρέχει ένα πολύ εύχρηστο γραφικό περιβάλλον.

Για να μπορέσουμε όμως να εντάξουμε τον ελεγκτή μας, θα πρέπει να αλλάξουμε τον κώδικα μας αρκετά ώστε να μη γίνεται η επεξεργασία των κινήσεων στον ίδιο το κώδικα, αλλά στον ελεγκτή μας. Επειδή προφανώς αυτό χρήζει μελέτης και συνεχούς διόρθωσης, υπάρχει έτοιμος κώδικας στο φάκελο fuzzy που βρίσκεται μέσα στο φάκελο trs. Ο φάκελος περιέχει παράδειγμα κώδικα λειτουργίας κίνησης ρομπότ με ελεγκτή ασαφούς λογικής.

Φορτώνουμε λοιπόν το fuzzyexample.ttt στο V-REP με διπλό κλικ πάνω του και το fuzzyexample.m στο MATLAB. Πατώντας Start Simulation στο V-REP και έπειτα Run στο MATLAB, το ρομπότ θα πρέπει να κινείται.

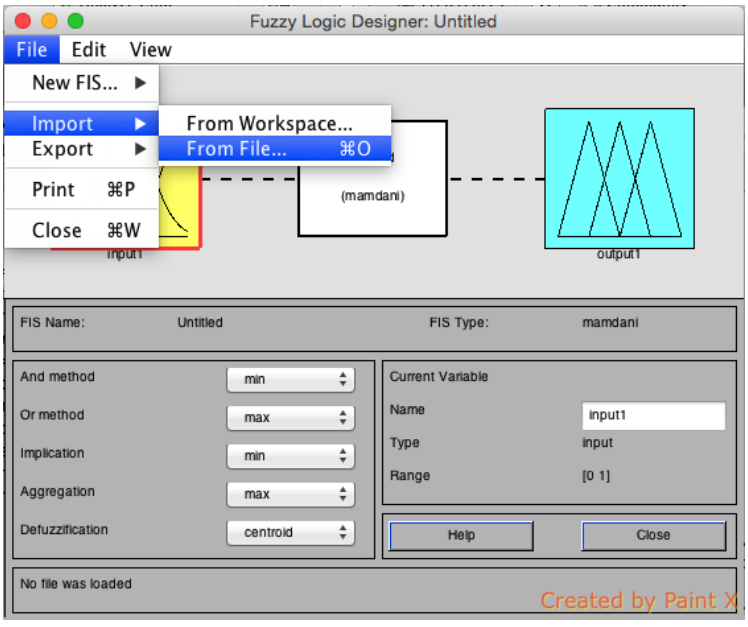
Ένα πλεονέκτημα που έχει αυτός ο κώδικας, είναι ότι ο ελεγκτής του είναι παραμετροποιήσιμος, γεγονός που αφήνει πολλά περιθώρια βελτίωσης και μάλιστα χωρίς περαιτέρω αλλαγές στον κώδικα.

Για να βελτιώσουμε τον ελεγκτή, πηγαίνουμε στο Command Window στο MATLAB και πληκτρολογούμε fuzzy και πατάμε Enter. Αμέσως εμφανίζεται ένα παράθυρο δημιουργίας ασαφούς ελεγκτή (Εικόνα 22).



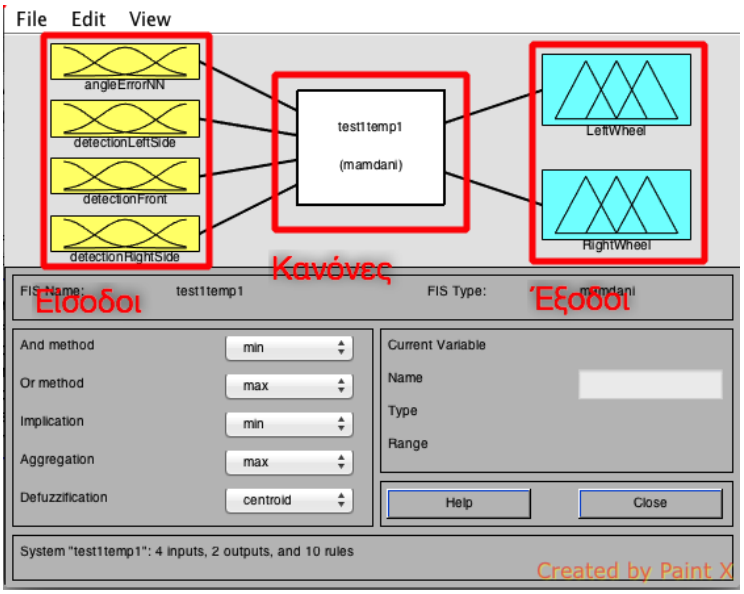
Εικόνα 22:Παράθυρο Δημιουργίας Ασαφούς Ελεγκτή

Για να φορτώσουμε έναν ήδη υπάρχον ελεγκτή, όπως στην περίπτωση μας, πατάμε File, Import, From File και επιλέγουμε το fuzzyexample.fis που έχουμε στο φάκελο fuzzy (Εικόνα 23).



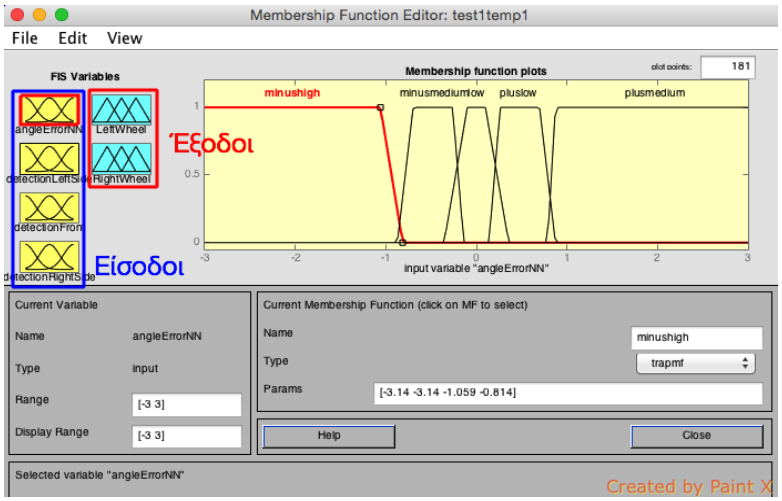
Εικόνα 23:Εισαγωγή Έτοιμου Ελεγκτή στο Παράθυρο Ασαφούς Ελεγκτή

Στην παρακάτω εικόνα βλέπουμε τον ελεγκτή μας. Αυτός ο ελεγκτής έχει 4 εισόδους και 2 εξόδους. Οι είσοδοι είναι αριστερά και οι έξοδοι δεξιά. Αυτό που είναι στη μέση περιέχει τους κανόνες του ελεγκτή μας, στους οποίους θα αναφερθούμε αργότερα (Εικόνα 24).



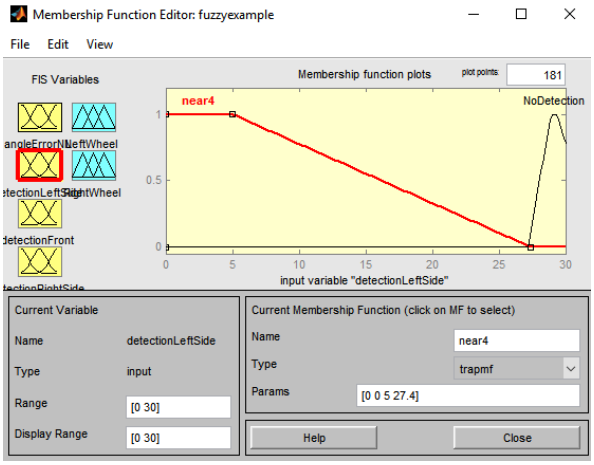
Εικόνα 24: Είσοδοι, Κανόνες και Έξοδοι στον Ελεγκτή Ασαφούς Λογικής

Αν θέλουμε να αλλάξουμε τα πεδία των εισόδων, πατάμε διπλό κλίκ πάνω τους και θα μας εμφανιστεί ένα νέο παράθυρο με όλες τις εισόδους και εξόδους μας, όπως φαίνεται παρακάτω (Εικόνα 25).



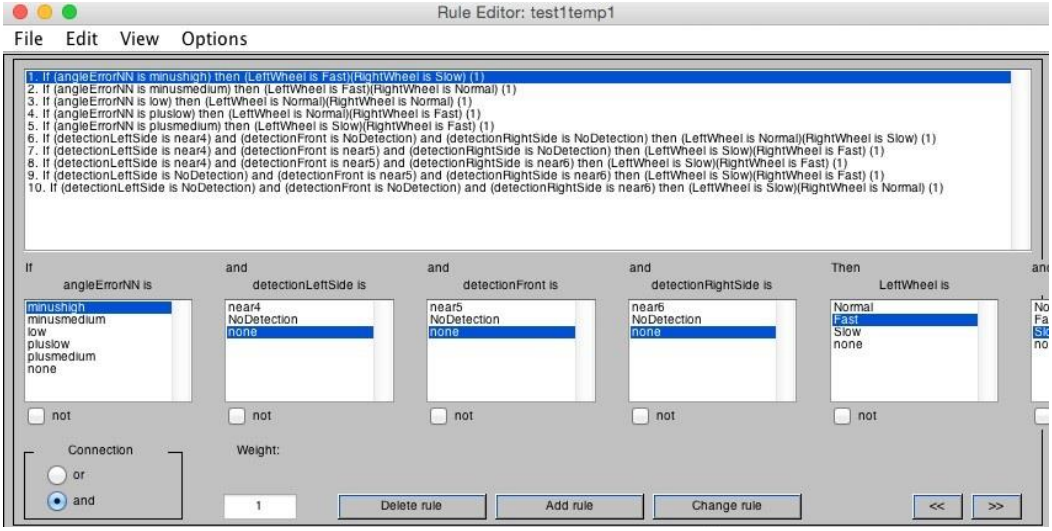
Εικόνα 25: Παράθυρο Επεξεργασίας Πεδίων Εισόδων – Εξόδων Ελεγκτή

Στη παρακάτω εικόνα φαίνονται τα πεδία της εισόδου detectionLeftSide της οποίας μπορούμε να τα αλλάξουμε απευθείας με το Ποντίκι πάνω στις γραμμές, ή από κάτω στο πεδίο Params αλλάζοντας τους αριθμούς για περισσότερη ακρίβεια. Μπορούμε να αλλάξουμε και την μορφή της γραμμής από το πεδίο Type (Εικόνα 26).



Εικόνα 26: Επεξεργασία Πεδίου Εισόδου Ελεγκτή

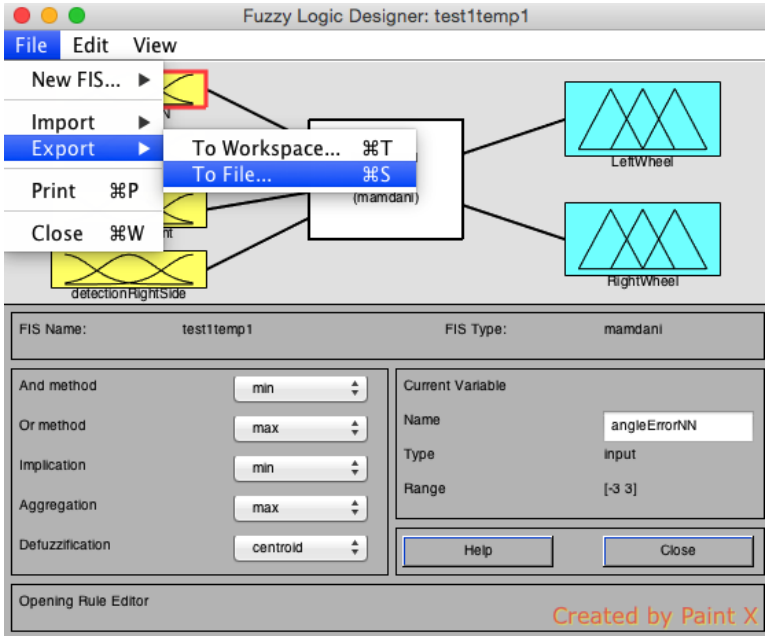
Μπορούμε να κάνουμε το ίδιο και στις εξόδους μας, καθώς μπορούμε να μετονομάσουμε όλα τα πεδία έτσι ώστε να θυμόμαστε τι αντιπροσωπεύει το καθένα.
Πατώντας διπλό κλικ στους κανόνες εμφανίζεται το παρακάτω παράθυρο (Εικόνα 27).



Εικόνα 27: Παράθυρο Επεξεργασίας Κανόνων Ελεγκτή

Εδώ έχουμε διαθέσιμους όλους τους συνδυασμούς εισόδων – εξόδων και μπορούμε να φτιάξουμε κανόνες με λογικές πράξεις AND – OR – NAND – NOR. Μας παρέχεται η δυνατότητα επίσης να ορίζουμε την βαρύτητα του κάθε κανόνα με βάση τη λογική μας. Αυτό μπορούμε να το ρυθμίσουμε από το πεδίο Weight το οποίο έχει ως προεπιλογή το 1, όμως εμείς μπορούμε να του βάλουμε οποιαδήποτε τιμή από 0 ως 1.
Μόλις τελειώσουμε την βελτίωση, ή την εκ νέου δημιουργία του ελεγκτή, τότε πατάμε στο κυρίως παράθυρο του ελεγκτή μας File , Export , To File και αποθηκεύουμε τον ελεγκτή μας εκεί που θέλουμε (Εικόνα 28).

Προσοχή στο όνομα που θα του δώσουμε γιατί ο κώδικας φορτώνει τον ελεγκτή βάσει δεδομένου ονόματος που του ορίζουμε εμείς, οπότε ενδεχομένως να χρειαστεί να επέμβουμε και στον κώδικα.



Εικόνα 28: Απόθήκευση / Εξαγωγή Καινούργιου Ελεγκτή

4.4 Ελεγκτής Ασαφούς Λογικής (Fuzzy Control) και V-Rep

Εκτελώντας το fuzzyexample.m που είχαμε δεί στην ενότητα 5.3.1 (4.3.1), παρατηρούμε την κίνηση του ρομπότ στο χώρο. Μπορούμε αμέσως να καταλάβουμε ότι δεν παραγματοποιεί ευθείες κίνησεις, δεν σταθεροποιείται έυκολα και επίσης κάποιες φορές δεν φτάνει ποτέ το στόχο. Αυτό οφείλεται στο ότι ο ασαφής ελεγκτής μας δεν είναι μελετημένος τόσο ώστε να είναι σταθερός, αλλά και ένα πολύ μικρό ποσοστό ευθύνης οφείλεται και στην καθυστέρηση του MATLAB να υπολογίσει τις κινήσεις με αποτέλεσμα να υπάρχει μεγάλο διάστημα μεταξύ των ανανεόσεων των ταχυτήτων των κινητήρων.

Για να εξαλείψουμε προβλήματα τέτοιου είδους, δηλαδή που οφείλονται στην προσομοίωση και όχι στο πρόγραμμα, κατεβάζουμε την ταχύτητα προσομοίωσης από το V-REP πατώντας το εικονίδιο με την χελώνα δυο φορές. Εφόσον το κάνουμε αυτό θα παρατηρήσουμε μικρότερες διακυμάνσεις στη κίνηση του ρομπότ. Για να μπορέσουμε να ελέγξουμε το ρομπότ με ελεγκτή ασαφούς λογικής, θα πρέπει να λάβουμε υπόψην πρώτα κάποιες παραμέτρους που ακολουθούν παρακάτω.

4.5 Έλεγχος με βάση τον ελεγκτή Fuzzy

4.5.1 Πραγματοποίηση Απλής Μετάβασης σε σημεία στο χώρο

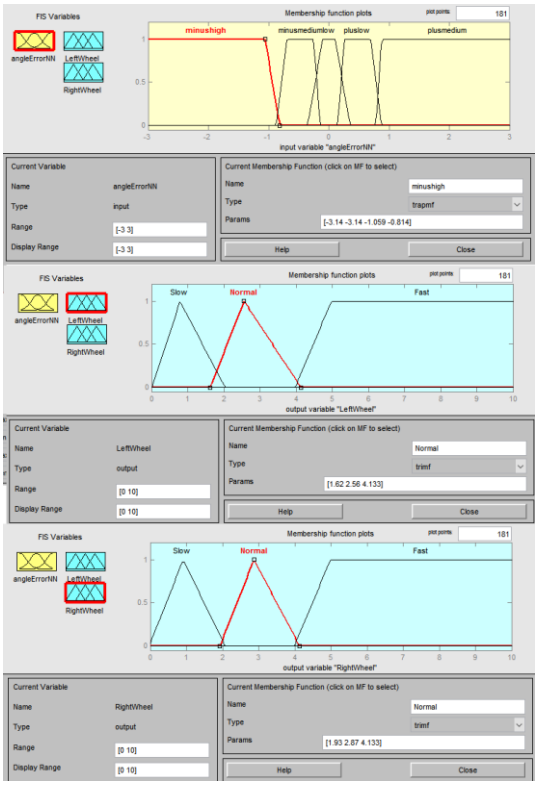
Η πιο απλή ρουτίνα που μπορούμε να φτιάξουμε είναι ρουτίνα απλής κίνησης, κάνοντας τα εξής βήματα στο MATLAB:
Πατάμε δημιουργία – Νέο m-file και αρχίζουμε να φτιάχνουμε την ρουτίνα αρχικοποίησης σύνδεσης του MATLAB – VREP, η οποία φαίνεται στην παρακάτω εικόνα (Εικόνα 42) και παρατίθεται και στο παράρτημα έτοιμη.

Το αποθηκεύουμε με όνομα Arxikopoihsh ώστε να ξέρουμε τι ρόλο θα παίζει όταν το καλέσουμε στο κυρίως πρόγραμμα μας.

Στη συνέχεια φτιάχνουμε ένα νέο m-file με το οποίο θα γίνουν γνωστά στο MATLAB, όλα εκείνα τα κομμάτια που θα ελένξει στο ρομπότ, όπως κινητήρες, αισθητήρια κτλ. Για παράδειγμα το V-REP πρέπει να συνεννοηθεί με το MATLAB για το που θα του στέλνει τις ταχύτητες για τους κινητήρες, ώστε το V-REP να τις διαβάζει από εκεί. Για να το κάνουμε αυτό το VREP έχει κάποιες εντολές με τις οποίες δίνει πληροφορίες ελέγχου πίσω στο MATLAB. Παρακάτω ακολουθεί η ρουτίνα HandleObject_v2 η οποία υπάρχει έτοιμη στο παράρτημα και παρατίθεται στην παρακάτω εικόνα (Εικόνα 43).

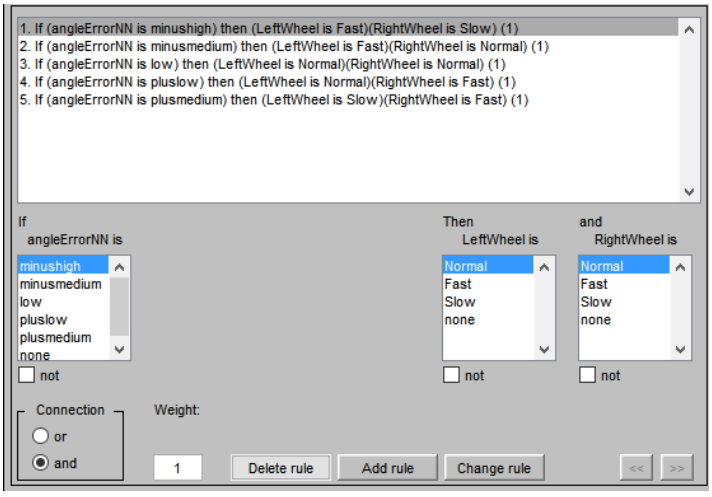
Στην πρώτη γραμμή αναγνωρίζεται το ρομπότ. Στην ένατη ως την ενδέκατη ρυθμίζονται οι ιδιότητες του, οι οποίες από προεπιλογή γράφονται έτσι. Έπειτα ορίζεται η αρχική του θέση στο χώρο και ο προσανατολισμός του. Και μετά αναγνωρίζονται ξεχωριστά τα μοτέρ κίνησης. Ακολουθούν εντολές που δίνουν πληροφορίες για την θέση και τον προσανατολισμού του, ώστε να μπορέσουμε αργότερα αν θελήσουμε να ελένξουμε αν έγινε επιτυχώς ο ορισμός τους και ενδιάμεσα στο κώδικα φαίνονται και εντολές ελέγχου (if) που ελέγχουν αν όλα βαίνουν καλώς, καθώς το VREP θα επιστρέψει μη μηδενική τιμή στην μεταβλητή errorCode αν υπάρξει πρόβλημα και τότε ανάλογα με τον κωδικό μπορούμε να διαπιστώσουμε την φύση του προβλήματος.

Πριν προχωρήσουμε στην δημιουργία του κυρίου προγράμματος, μπορούμε να δημιουργήσουμε τον ελεγκτή μας. Αυτό μπορεί να φαίνεται λίγο ανορθόδοξο, όμως συνήθως βοηθάει στην υλοποίηση του κώδικα ένας έτοιμος ελεκτής καθώς καθοδηγεί τον τρόπο σκέψης.
Έτσι λοιπόν στο Command Window του MATLAB, γράφουμε fuzzy ώστε να ανοίξει το παράθυρο δημιουργίας, και δημιουργούμε τον ελεγκτή, συμφωνα με αυτά που είπαμε παραπάνω, λαμβάνοντας υπόψην ότι έχουμε δύο εξόδους (Αριστερό και Δεξί Κινητήρα) και μόνο μία είσοδο η οποία θα είναι μια μεταβλητή στην οποία θα παίρνει τιμές σύμφωνα με το σφάλμα διεύθυνσης του ρομπότ ως προς το στόχο (AngleError). Ο ελεγκτής λοιπόν θα πρέπει να μοιάζει με αυτόν στις παρακάτω εικόνες (Εικόνα 29).



Εικόνα 29: Πεδία Εισόδων Εξόδων Ελεγκτή Ασαφούς Λογικής

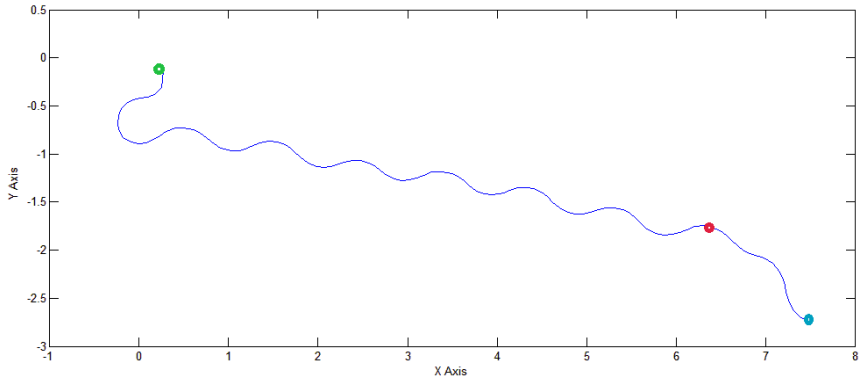
Θα πρέπει να έχει τους κανόνες όπως φαίνονται στην Εικόνα 30.



Εικόνα 30: Κανόνες Ελεγκτή Ασαφούς Λογικής

Έχοντας αποθηκεύσει τώρα τον ελεγκτή μας ως FIS File είμαστε έτοιμοι να προχωρήσουμε στο κυρίως πρόγραμμα μας. Το πρόγραμμα θα πρέπει να καλεί τις ρουτίνες που φτιάξαμε (Αρχικοποίηση και Αναγνώριση Αντικειμένων), να φορτώνει τον ελεγκτή ασαφούς λογικής και μέσα στην ρουτίνα επανάληψης κίνησης που μπορούμε να την φτιάξουμε με μια While, να ενημερώνεται για την παρούσα θέση και προσανατολισμό του, να υπολογίζει το σφάλμα διεύθυνσης από το στόχο, να εισάγει τα δεδομένα στον ελεγκτή, να παίρνει τα αποτελέσματα από τον ελεγκτή και να ενημερώνει την ταχύτητα των κινητήρων του ρομπότ. Όλα αυτά μέχρι να υπάρχει σήμανση τέλους από το πρόγραμμα. Το πρόγραμμα test3_v2.m που φαίνεται στην Εικόνα 44, κάνει ακριβώς τα παραπάνω με χρήση ελεκτή

ασαφούς λογικής με όνομα test3_v2.fis και η πορεία που πραγματοποιεί φαίνεται στην παρακάτω εικόνα (Εικόνα 31: Πορεία Ρομπότ στο Χώρο (Πράσινο Σημείο Εκκίνησης , Κόκκινο Ενδιάμεσος Στόχος , Μπλέ Τελικός Στόχος)Εικόνα 31).



Εικόνα 31: Πορεία Ρομπότ στο Χώρο (Πράσινο Σημείο Εκκίνησης , Κόκκινο Ενδιάμεσος Στόχος , Μπλέ Τελικός Στόχος)

5. Κεφάλαιο 6: Πραγματοποίηση Απλής Μετάβασης σε σημεία στο χώρο με αποφυγή εμποδίων

Τώρα εφόσον θελήσουμε τα κινείται το ρομπότ, όμως να μπορεί να αντιλαμβάνεται ένα εμπόδιο και να τροποποιεί την πορεία του έτσι ώστε να το αποφεύγει, μπορούμε να τροποποιήσουμε τον παραπάνω κώδικα ως εξής:
Στη ρουτίνα αναγνώρισης αντικειμένων που φτιάξαμε παραπάνω θα πρέπει να εισάγουμε ακριβώς πριν τις εντολές αναγνώρισης κινητήρων και όχι πιο πριν τις εντολές της παρακάτω εικόνας (Εικόνα 32), ώστε να συνδέσουμε το MATLAB με τους υπερηχητικούς αισθητήρες του Pioneer.

```
end
[errorCode, h.US1]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor1', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US2]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor2', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US3]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor3', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US4]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor4', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US5]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor5', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US6]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor6', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US7]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor7', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US8]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor8', vrep.simx_opmode_oneshot_wait);
[errorCode, h.US16]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor16', vrep.simx_opmode_oneshot_wait);
```

Εικόνα 32: Γραμμές Κώδικα που πρέπει να εισαχθούν στο προηγούμενο πρόγραμμα Σύνδεσεις Αντικειμένων (HandleObject_v2)

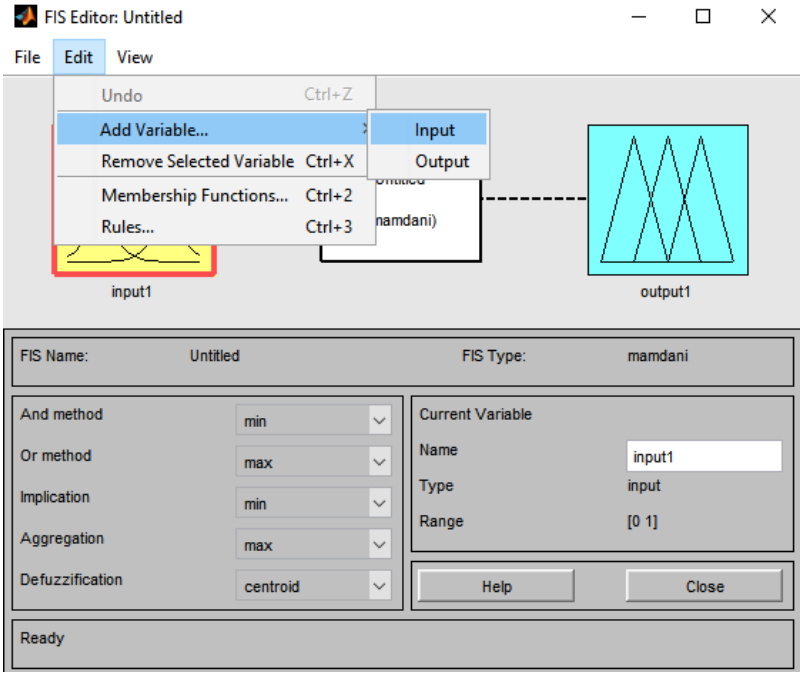
Μόλις αποθηκεύσουμε το παραπάνω, πάμε στο κυρίως πρόγραμμα μας και εισάγουμε, μέσα στην ρουτίνα επανάληψης οπουδήποτε θέλουμε αρκεί να είναι πριν το κομμάτι που ενημερώνεται ο ελεγκτής μας, τις εξής γραμμές:

```
[errorCode,detectionState3,detectedPoint3,detectedObjectHandle3,detectedSurfaceNormalVector3]=vrep.simxReadProximitySensor(clientID, h.US3, vrep.simx_opmode_oneshot_wait);

[errorCode,detectionState4,detectedPoint4,detectedObjectHandle4,detectedSurfaceNormalVector4]=vrep.simxReadProximitySensor(clientID, h.US4, vrep.simx_opmode_oneshot_wait);

[errorCode,detectionState5,detectedPoint5,detectedObjectHandle5,detectedSurfaceNormalVector5]=vrep.simxReadProximitySensor(clientID, h.US5, vrep.simx_opmode_oneshot_wait);

[errorCode,detectionState6,detectedPoint6,detectedObjectHandle6,detectedSurfaceNormalVector6]=vrep.simxReadProximitySensor(clientID, h.US6, vrep.simx_opmode_oneshot_wait);

[errorCode,detectionState7,detectedPoint7,detectedObjectHandle7,detectedSurfaceNormalVector7]=vrep.simxReadProximitySensor(clientID, h.US7, vrep.simx_opmode_oneshot_wait);
```

Τώρα παίρνουμε πληροφορίες από πέντε αισθητήρες που καλύπτουν το πλάγιο και μπροστινό μέρος του ρομπότ.

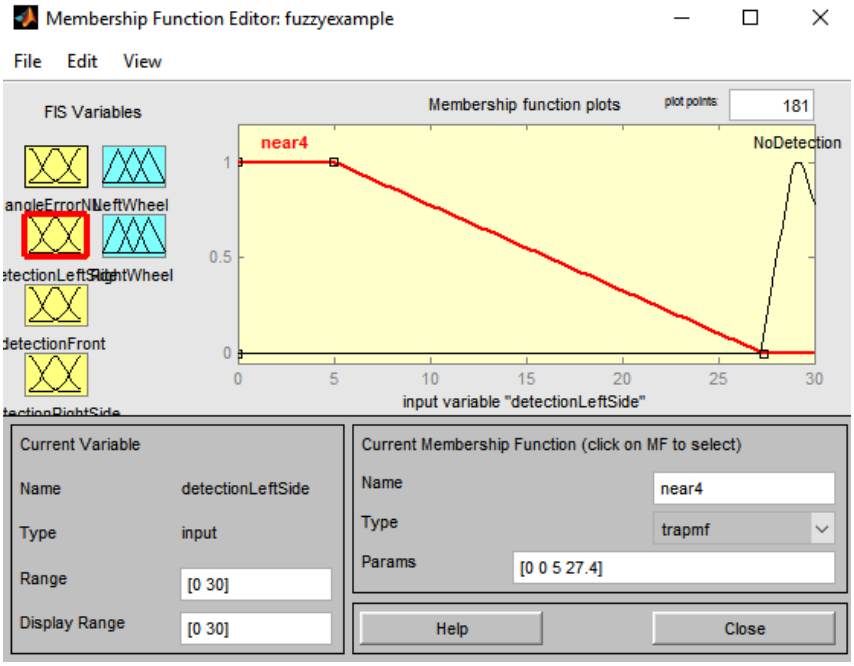
Μετά από τα παραπάνω θα πρέπει να κάνουμε τον υπολογισμό θέσης πιθανού αντικειμένου. Για παράδειγμα, αν είναι αριστερά θα το ανιχνεύσουν σίγουρα οι αισθητήρες 3 και 4. Ενώ αν είναι δεξιά οι 6 και 7. Αν είναι ευθεία τότε θα ανιχνεύσει ο αισθητήρας 5 το εμπόδιο. Πρέπει λοιπόν να μειώσουμε τα αποτελέσματα από πέντε , σε τρία, ώστε να έχουμε 3 επιπλέον εισόδους μετά στον ελεγκτή που θα τροποποιήσουμε για να απλοποιηθεί και η διαδικασία δημιουργίας κανόνων του.
Έτσι λοιπόν προσθέτουμε τις εξής εντολές (Εικόνα 45).

Τώρα μπορούμε να αποθηκεύσουμε το κυρίως πρόγραμμα και το μόνο που μένει είναι να τροποποιήσουμε τον ελεγκτή μας.
Πληκτρολογούμε λοιπόν fuzzy στο Command Window και ανοίγουμε τον ελεγκτή μας.
Προσθέτουμε τρείς εισόδους πατώντας στο μενού Edit – Add Variable – Input (Εικόνα 33) και τις ονομάζουμε detectionLeftSide , detectionFront , detectionRightSide ή όπως θέλουμε αρκεί να έιναι με την σειρά Αριστερή , Εμπρόσθια και Δεξιά Μεριά. Αν αλλάξουμε την σειρά θα πρέπει να αλλάξουμε και την σειρά με την οποία θα εισάγουμε και τα δεδομένα.



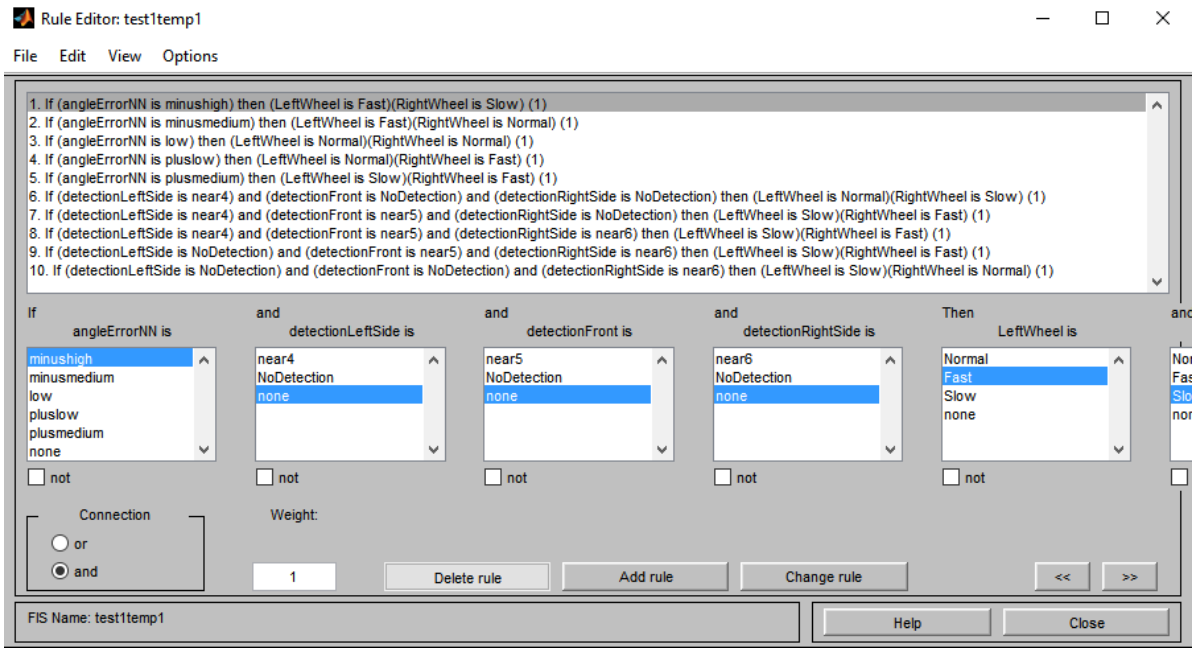
Εικόνα 33: Εισαγωγή Μεταβλητών Εισόδου στον Ελεγκτή

Στη συνέχεια με διπλό κλικ πάνω σε ένα τα επεξεργαζόμαστε έχοντας υπόψην ότι όσο πιο μικρή τιμή έχουμε τόσο πιο κοντά είναι το εμπόδιο. Επομένως μπορούμε για παράδειγμα να τα φτιάξουμε όλα όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 34) έχοντας δυο πεδία, ένα για κοντά και ένα για καμία αναγνώριση.



Εικόνα 34: Διαμόρφωση Πεδίων Εισόδων Αισθητήρων

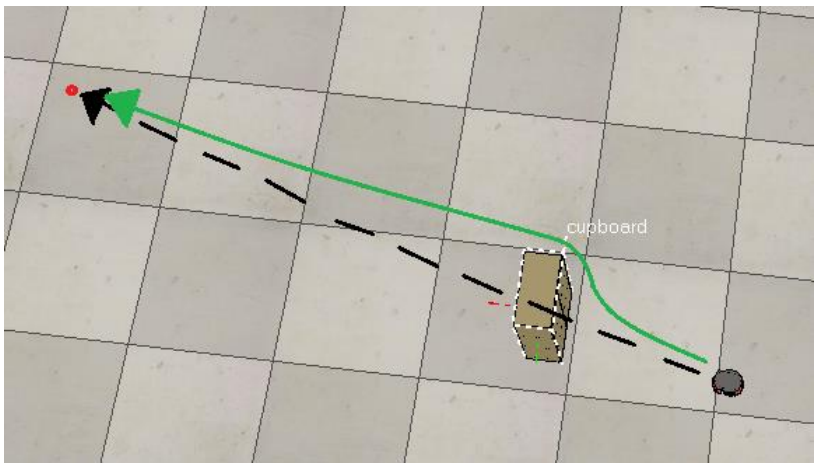
Έπειτα προσθέτουμε τους απαραίτητους κανόνες ώστε να λαμβάνονται υπόψη τα νέα δεδομένα που εισάγονται (Εικόνα 35).



Εικόνα 35: Ανανεομένοι Κανόνες Ελεγκτή Ασαφούς Λογικής

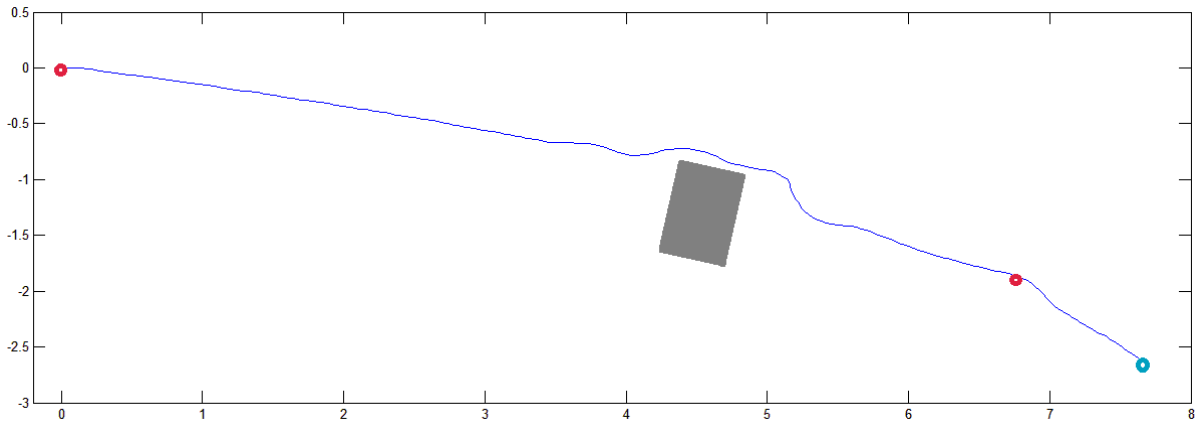
Αποθηκεύουμε τον ελεγκτή και το πρόγραμμα είναι έτοιμο για προσομοίωση. Αν προσομοιώσουμε τώρα θα παρατηρήσουμε ότι το ρομπότ αποφεύγει τα εμπόδια, όμως έχει πρόβλημα όταν αυτά εμφανιστούν ξαφνικά πολύ μπροστά του. Αυτό συμβαίνει γιατί αν είναι μεγάλο το αντικείμενο, το ρομπότ θα προσκρούσει πάνω του και λόγω της προσομοίωσης φυσικής του προγράμματος θα γίνει καθετο με το αντικείμενο κι έτσι θα κολλήσει. Παρόλα αυτά όμως επειδή το ρομπότ δύναται να εντοπίζει από αρκετή απόσταση

τα αντικείμενα δεν υπάρχει πρόβλημα σε χώρο που υπάρχουν στατικά ή σχετικά αργά κινούμενα εμπόδια (Εικόνα 36).



Εικόνα 36: (Μαύρο: Πορεία Χωρίς Εμπόδιο, Πράσινο: Πορεία Με Αποφυγή Εμποδίου)

Προσομοιώνοντας λοιπόν τον κώδικά με αποφυγή αντικειμένων, τοποθετώντας στο χώρο ένα μεγάλο ακίνητο εμπόδιο, η τροχιά του ρομπότ φαίνεται να αλλάζει ώστε να το αποφύγει, όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 37).



Εικόνα 37: Πορεία Ρομπότ με Αποφυγή Εμποδίου (Γκρί Εμπόδιο , Κόκκινο Αρχικό Σημείο και Ενδιάμεσος Στόχος, Μπλέ Τελικός Στόχος)

Η παραπάνω προσομοίωση έδειξε ότι όντως γίνεται αποφυγή εμποδίων, όμως δεν γίνεται σε πολύ μεγάλο βαθμό. Επι τοις ουσίας, ο ελεγκτής λαμβάνει υπόψη του το εμπόδιο εφόσον το εντοπίσει και δρα από την πρώτη στιγμή. Όταν όμως διαπιστωθεί εμπόδιο ξαφνικά σε πολύ μικρή απόσταση από το ρομπότ, τότε δεν προλαβαίνει να αντιδράσει. Επίσης πρόβλημα παρατηρήθηκε όσο αναφορά τον εντοπισμό εμποδίου με τον εμπρόσθιο αισθητήρα. Εκεί έχουμε αντίδραση, απλά θα πρέπει να γίνει κάποια βελτιστοποίηση ώστε να έχουμε πιο δραστική αλλαγή πορείας, ώστε να αποφεύγουμε τις κρούσεις.

Με πρώτη ματιά, ο ελεγκτής ασαφούς λογικής δεν δουλέυει τόσο καλά, όσο θα δούλευε ένας απλός κώδικας ελέγχου με συμβατική μέθοδο που είδαμε στην ενότητα 5.2 (4.2), όμως είναι πιο ευέλικτος ο κώδικας στην βελτίωση, καθώς μια απλή αλλαγή του ελεγκτή

τροποποιεί πλήρως το αποτέλεσμα, καθώς υπάρχουν και τεχνικές βελτίωσης των ελεγκτών ασαφούς λογικής, όπως για παράδειγμα η μέθοδος γενετικών αλγορίθμων.

6. ΚΕΦΑΛΑΙΟ 7: Επίλογος

Το πρόγραμμα / ελεγκτής λειτουργεί αρκετά καλά αν ληφθεί υπόψη και ο τρόπος με τον οποίο προσομοιώνεται. Σε πραγματικές συνθήκες το πρόγραμμα ελεγκτής θα τρέχει αρκετές φορές γρηγορότερα, λόγω του ότι υπάρχουν ταχύτερες γλώσσες προγραμματισμού από την MATLAB και μέρος της υπολογιστικής δύναμης δεν θα καταναλώνεται για την προσομοίωση του. Επομένως θα υπάρχει καλύτερη και ποιοτικότερη εκτέλεση του. Σίγουρα θα υπάρχουν και άλλου είδους προβλήματα, που όμως θα είναι πιο πρακτικής φύσης.
Η προσομοίωση αυτή μας επιτρέπει να προσομοιώσουμε τον έλεγχο ενός έντροχου ρομποτικού συστήματος με την συμβατική μέθοδο, αλλά και οποιοδήποτε ελεγκτή ασαφούς λογικής . Αυτό βέβαια υποστηρίζεται για ελεγκτές ασαφούς λογικής που θα λαμβάνουν υπόψη ως δεδομένα εισόδων το σφάλμα προσανατολισμού και τα δεδομένα των αισθητήρων. Επιλέγοντας έναν ελεγκτή με παραπάνω εισόδους, θα πρέπει να μεριμνήσουμε να υπολογίζουμε τις τιμές των μεταβλητών αυτών στο κυρίως πρόγραμμα, κάτι που είναι πάρα πολύ απλό , δεδομένου ότι έχουμε ήδη τον ελεγκτή μας.

Άρα παίρνοντας υπόψη τα παραπάνω, το πρόγραμμα ελεγκτής καθίσταται αρκετά ευέλικτο σε αλλαγές και τροποποιήσεις, κάτι που πηγάζει και από την γλώσσα στην οποία είναι γραμμένο (MATLAB). Επίσης πραγματοποιώντας ένα τέτοιο πρόγραμμα με διαφορετικές τεχνικές ελέγχου κίνησης μπορεί να δώσει πάρα πολλές πληροφορίες σχετικά με την κάθε τεχνική. Διαπιστώθηκε δηλαδή ότι η τεχνική ελέγχου με ασαφή λογική είναι λιγότερο σταθερή όσο αναφορά το τελικό αποτέλεσμα σε σχέση με την συμβατική τεχνική ελέγχου, όμως είναι πάρα πολύ πιο ευέλικτη σε τροποποιήσεις, επομένως και πιο φιλική σε βελτιώσεις.

Ανάλογα με την χρήση για την οποία προορίζεται το πρόγραμμα, μπορεί μια μέθοδος να λειτουργεί καλύτερα από την άλλη. Για παράδειγμα, εφόσον ένα έντροχο ρομπότ μεταφέρει κάτι που απαιτεί τεράστια σταθερότητα από ένα σημείο σε ένα άλλο, μια λογική προσέγγιση , όπως η συμβατική τεχνική θα λειτουργούσε καλύτερα, ενώ σε περίπτωση που ο στόχος είναι μια ταχύτερη μετάβαση, η τεχνική ασαφούς λογικής, λόγω του ότι μπορούν να αξιοποιηθούν κι άλλες μεθόδοι για την βελτίωση του ελεγκτή, είναι η πλέον κατάλληλη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) http://edurobotics.weebly.com (Τελευταία Πρόσβαση 15/1/2016)
2) http://users.sch.gr (Τελευταία Πρόσβαση 15/1/2016)
3) https://www.cyberbotics.com (Τελευταία Πρόσβαση 15/1/2016)
4) http://www.robotica.gr (Τελευταία Πρόσβαση 15/1/2016)
5) https://www.cyberbotics.com (Τελευταία Πρόσβαση 15/1/2016)
6) www.axiarchos.gr/robotiki-xeirourgiki (Τελευταία Πρόσβαση 15/1/2016)
7) Arduinobots.wordpress.com (Τελευταία Πρόσβαση 15/1/2016)
8) Cyberbotics.com (Τελευταία Πρόσβαση 15/1/2016)
9) Coppeliarobotics.com (Τελευταία Πρόσβαση 15/1/2016)
10) Petercorke.com (Τελευταία Πρόσβαση 15/1/2016)
11) Mathworks.com (Τελευταία Πρόσβαση 15/1/2016)

12) Ασαφή Συστήματα (Παπαδάκης Στέλιος , Αδαμίδης Παναγιώτης) (ΤΕΙ Θεσσαλονίκης Τμήμα Πληροφορικής)

13) Goldberg D.E.,GENETIC ALGORITHMS in Search, Optimization and Machine Learning, Addison Wesley Publishing Company, Inc., 1989.

14) Holland J.H., Adaptation in Natural and Artificial Systems, M.I.T. Press, 1975.

15) Michalewich Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer–Verlag, 2nd ed., 1992.

16) Mitchel, Melanie, An Introduction to Genetic Algorithms, MIT Press, 1996.

17) Davis L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

18) Γενετικοί Αλγόριθμοι και Εφαρμογές (Λυκοθανάσης Σπυρίδων)

19) Fitness Function in Evolutionary Robotics: A Survey and Analysis (Andrew L. Nelson , Gregory J. Barlow, Lefteris Doitsidis)

20) Bio-Inspired Artificial Intelligence: Theories, Methods and Technologies (Dario Floreano , Claudio Mattiussi)

21) Where Am I? Sensors and Methods for Mobile Robot Positioning (J. Borenstein, H. R. Everett, L. Feng)

22) Ευφυής Έλεγχος (Ελευθέριος Δοϊτσίδης)

23) http://en.wikipedia.org/wiki/Fuzzy_control_system (Τελευταία επίσκεψη: 8/2/2016)

24) Οικονόμου Παναγιώτης Δρ. Ε. Παπαγεωργίου http://www.terrapapers.com/wp-content/uploads/2014/02/fuzzy_logic.pdf (Τελευταία επίσκεψη: 8/1/2016)

25) http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/Fuzzy/Fuzzy1.html (Τελευταία επίσκεψη: 8/1/2016)

26) Εισαγωγή στην Ασαφή Λογική Δρ. Κυριάκος Δεληπαράσχος

27) Ασαφή Συστήματα Θεωρία και Εργαστηριακές Ασκήσεις Παπαδάκης Στέλιος Αδαμίδης Παναγιώτης Θεσσαλονίκη

ΠΑΡΑΡΤΗΜΑ

Κώδικας Απλής Κίνησης στο Χώρο

```
try
    clear all                   %Καθαρίζει όλες τις μεταβλητές
    close all                   %Κλείνει όλα τα παράθυρα
    clc                         %Καθαρίζει την οθόνη
    disp('Initializing Robotic Toolbox');   %Μύνημα
    initialiseRoboticsToolbox   %Ρουτίνα φόρτωσης Robotics Toolbox στο MATLAB
    vrep=remApi('remoteApi');   %Ρουτίνα του V-REP για έναρξη ζεύξης με το MATLAB
    vrep.simxFinish(-1);        %Σήμανση στο V-REP για έναρξη προσομοίωσης
    clientID=vrep.simxStart('127.0.0.1',19999,true,true,2000,5);    %Ρουτίνα V-REP για έναρξη προσομοίωσης
    if clientID==-1
        error(['Connection Error:Failed to establish Connection to VREP!\n                ',...
                               'Is VRep server listening port 19999?'], 'Is VRep server listening @ port 19999?')
    else
        disp('Connected to remote API server');
        [errorCode]=vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot_wait);
        if errorCode ~= 0
            error(['Error Returned:vrep.simxStartSimulation Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
        end
    end
    i=0; j=1;
    goalxyz=[7 -2 0; -4 -10 0;10 10 0; -1 -1 0; -10 10 0; -10 -6 0];        %Ορισμός Κάποιων Σημείων Στόχων στο Χώρο
    [maxj, tmp_]=size(goalxyz);
    motorspeeds=[5 5];                                                      %Αρχικοποίηση της μεταβλητής των κινητήρων
    [errorCode, h.pioneer1] = vrep.simxGetObjectHandle(clientID, 'Pioneer1', vrep.simx_opmode_oneshot_wait);%Αναγνώριση του ρομπότ
    if errorCode ~=0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, h.goal]     = vrep.simxGetObjectHandle(clientID, 'goal' , vrep.simx_opmode_oneshot_wait);%Αναγνώριση προσομοιωμένου στόχου
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, sensor]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor5', vrep.simx_opmode_oneshot_wait);
    %Αναγνώριση εμπρόσθιου αισθητήρα
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
```

Εικόνα 38: Πρώτο Κομμάτι Απλού Κώδικα Κίνησης

```
    end
    [errorCode, sensor]=vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_ultrasonicSensor5', vrep.simx_opmode_oneshot_wait);
    %Αναγνώριση εμπρόσθιου αισθητήρα
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, prop]=vrep.simxGetModelProperty(clientID, h.pioneer1, vrep.simx_opmode_oneshot_wait );%Αρχικοποίηση Ρομπότ
    prop=bitor(prop,vrep.sim_modelproperty_not_dynamic+vrep.sim_modelproperty_not_respondable)
    [errorCode] = vrep.simxSetModelProperty(clientID, h.pioneer1, prop, vrep.simx_opmode_oneshot_wait )
    position=[0 0 0.1386]
    [errorCode]=vrep.simxSetObjectPosition(clientID, h.pioneer1,-1, position, vrep.simx_opmode_oneshot)
    [errorCode]=vrep.simxSetObjectOrientation(clientID, h.pioneer1,-1,[0 0 0], vrep.simx_opmode_oneshot)
    prop=prop-(vrep.sim_modelproperty_not_dynamic+vrep.sim_modelproperty_not_respondable)
    [errorCode] = vrep.simxSetModelProperty(clientID, h.pioneer1, prop, vrep.simx_opmode_oneshot_wait)
    [errorCode]=vrep.simxSetObjectPosition(clientID, h.goal, -1,goalxyz(j,:), vrep.simx_opmode_oneshot_wait);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxSetObjectPosition Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, h.leftMotor]  = vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_leftMotor', vrep.simx_opmode_oneshot_wait);
    %Αναγνώριση Αριστερού Κινητήρα
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Pioneer_p3dx_leftMotor Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, h.rightMotor] = vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_rightMotor', vrep.simx_opmode_oneshot_wait);
    %Αναγνώριση Δεξιού Κινητήρα
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Pioneer_p3dx_rightMotor Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, curr_position]    = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_streaming);%Λήψη Αρχικής Θέσης Ρομπότ
    pause(0.1)
    [errorCode, curr_position]    = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectPosition Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_streaming);
```

Εικόνα 39: Δεύτερο Κομμάτι Κώδικα Απλής Κίνησης

```
    [errorCode, curr_position]    = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectPosition Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_streaming);
    %Λήψη Αρχικού Προσανατολισμού Ρομπότ
    pause(0.1)
    [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectOrientation Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end

    while (vrep.simxGetConnectionId(clientID)~=-1)  %Ρουτίνα Κίνησης Ρομπότ
        i=i+1;
        [errorCode, curr_position] = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);%Λήψη Θέσης Ρομπότ
        [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);%Λήψη Προσανατολισμού
        rad2deg(curr_orientation(1)+curr_orientation(3));
        distance2goal=sqrt( (goalxyz(j,1)-curr_position(1))^2+(goalxyz(j,2)-curr_position(2))^2 );%Υπολογισμός Απόστασης Από Στόχο
        angle2goal=atan2(goalxyz(j,2)-curr_position(2), goalxyz(j,1)-curr_position(1));
        angleError=angle2goal-(curr_orientation(1)+curr_orientation(3));%Υπολογισμός Σφάλματος Προσανατολισμόυ σε σχέση με το στόχο
        angleErrorN=normaliseInCircle(angleError);
        angleErrorND=rad2deg(angleErrorN);
        k=angleErrorN/pi;
        [errorCode, detectionState]=vrep.simxReadProximitySensor(clientID, sensor, vrep.simx_opmode_oneshot_wait);%Έλεγχος αισθητήρα για εμπόδια
        if errorCode ~= 0
        error('Error Returned:vrep.simxReadProximitySensor');
        else
            simulationTime=vrep.simxGetLastCmdTime(clientID);
        end
        if (detectionState~=0)  %Έλεγχος για αντικείμενα σύμφωνα με τον εμπρόσθιο αισθητήρα
            motorspeeds(1)=3.1415*0.5;
            motorspeeds(2)=3.1415*0.2;
        else
            motorspeeds=[5 5];
        end
```

Εικόνα 40: Τρίτο Κομμάτι Κώδικα Απλής Κίνησης

```
                [errorCode]=vrep.simxSetObjectPosition(clientID, h.goal, -1,goalxyz(j,:), vrep.simx_opmode_oneshot_wait);
            else
                disp('prepare to break');
                 break
            end
        elseif (distance2goal < 0.1 )
            motorspeeds=[1 1];
        else
            motorspeeds=[5 5];
        end
         angleErrorNN=angleErrorN;
        if abs(angleErrorN) > 1
            angleErrorNN=abs(angleErrorN)/angleErrorN;
        end
        motorspeeds(1)=motorspeeds(1)*(1-angleErrorNN); %Αποθήκευση νέας ταχύτητας κινητήρων στις μεταβλητές κινητήρων
        motorspeeds(2)=motorspeeds(2)*(1+angleErrorNN);
        fprintf('(1-angleError) %2.2f, Motorspeeds:(%2.2f,%2.2f), detectionState:%2d \n', (1-angleError), motorspeeds(1),motorspeeds(2),detectionS
        [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.leftMotor,  motorspeeds(1), vrep.simx_opmode_streaming);%Ορισμός Νέας Ταχύτητας
        [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.rightMotor, motorspeeds(2), vrep.simx_opmode_streaming);%Κινητήρων
    end
    disp('break');
    [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.leftMotor,  0, vrep.simx_opmode_oneshot_wait);%Μηδενισμός Ταχύτητας Κινητήρων
    [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.rightMotor, 0, vrep.simx_opmode_oneshot_wait);
    [errorCode, curr_position] = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);%Λήψη Τελικής Θέσης Ρομπότ
    distance2goal=sqrt( (goalxyz(maxj,1)-curr_position(1))^2+(goalxyz(maxj,2)-curr_position(2))^2 )%Λήψη Τελικής Απόστασης από το στόχο
    vrep.simxFinish(clientID);%Τερματισμός Προσομοίωσης
    vrep.delete();
    catch err
    msgString = getReport(err)
        fprintf('error: %s',msgString)
        vrep.simxFinish(clientID);
        vrep.delete();
        disp('Abnormal program termination ! ');
end;
disp('Program end');
```

Εικόνα 41: Τελευταίο Κομμάτι Κώδικα Απλής Κίνησης

Κώδικας Απλής Κίνησης στο Χώρο Χωρίς Απουφυγή Εμποδίων (Ρουτίνες Αρχικοποίησης, Σύνδεσεις Αντικειμένων και Κυρίος Κώδικας)

```
clear all
close all
clc
disp('Initializing Robotic Toolbox');
initialiseRoboticsToolbox
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,2000,5);
if clientID==-1
    error(['Connection Error:Failed to establish Connection to VREP!\n                ',...
                            'Is VRep server listening port 19999?'], 'Is VRep server listening @ port 19999?')
else
    disp('Connected to remote API server');
    [errorCode]=vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot_wait);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxStartSimulation Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
end
```

Εικόνα 42: Κώδικας Ρουτίνας Αρχικοποίησης

```
[errorCode, h.pioneer1] = vrep.simxGetObjectHandle(clientID, 'Pioneer1', vrep.simx_opmode_oneshot_wait);
    if errorCode ~=0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, h.goal]     = vrep.simxGetObjectHandle(clientID, 'goal' , vrep.simx_opmode_oneshot_wait);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectHandle Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
    [errorCode, prop]=vrep.simxGetModelProperty(clientID, h.pioneer1, vrep.simx_opmode_oneshot_wait );
     prop=bitor(prop,vrep.sim_modelproperty_not_dynamic+vrep.sim_modelproperty_not_respondable);
    [errorCode] = vrep.simxSetModelProperty(clientID, h.pioneer1, prop, vrep.simx_opmode_oneshot_wait );
     position=[0 0 0.1386];
    [errorCode]=vrep.simxSetObjectPosition(clientID, h.pioneer1,-1, position, vrep.simx_opmode_oneshot) % change the position of the model
    [errorCode]=vrep.simxSetObjectOrientation(clientID, h.pioneer1,-1,[0 0 0], vrep.simx_opmode_oneshot);
     prop=prop-(vrep.sim_modelproperty_not_dynamic+vrep.sim_modelproperty_not_respondable);
    [errorCode] = vrep.simxSetModelProperty(clientID, h.pioneer1, prop, vrep.simx_opmode_oneshot_wait );
    [errorCode]=vrep.simxSetObjectPosition(clientID, h.goal, -1,goalxyz(j,:), vrep.simx_opmode_oneshot_wait);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxSetObjectPosition Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end

    [errorCode, h.leftMotor]  = vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_leftMotor', vrep.simx_opmode_oneshot_wait);
    [errorCode, h.rightMotor] = vrep.simxGetObjectHandle(clientID, 'Pioneer_p3dx_rightMotor', vrep.simx_opmode_oneshot_wait);
    [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_streaming);
    pause(0.1) %seconds
    [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);
    [errorCode, curr_position]    = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_streaming);
    pause(0.1) %seconds
    [errorCode, curr_position]    = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);
    if errorCode ~= 0
        error(['Error Returned:vrep.simxGetObjectOrientation Returned error. ErrorCode:',errorCode],['ErrorCode:',errorCode]);
    end
```

Εικόνα 43: Κώδικας Σύνδεσης – Ελέγχου Αντικειμένων στο V-REP από το MATLAB

```
try
    Arxikopoihsh %Ρουτίνα Αρχικοποίησης
    j=1;
    goalxyz=[7 -2 0; -4 -10 0;10 10 0; -1 -1 0; -10 10 0; -10 -6 0];    %Στόχοι
    [maxj, tmp_]=size(goalxyz); %Όριο Αριθμού Στόχων
    motorspeeds=[5 5];
    HandleObject_v2        %Ρουτίνα Αναγνώρισης Αντικειμένων
    initfis=readfis('test3.fis');   %Φόρτωση Ελεγκτή Ασαφούς Λογικής
    while (vrep.simxGetConnectionId(clientID)~=-1)  %Ρουτίνα Επανάληψης Κίνησης
        [errorCode, curr_position] = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);     %Ενημέρωση Θέσης
        [errorCode, curr_orientation] = vrep.simxGetObjectOrientation(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer);   %Ενημέρωση Προσανατολισμού
        distance2goal=sqrt( (goalxyz(j,1)-curr_position(1))^2+(goalxyz(j,2)-curr_position(2))^2 ); %Απόσταση Από Στόχο
        angle2goal=atan2(goalxyz(j,2)-curr_position(2), goalxyz(j,1)-curr_position(1)); %Υπολογισμός Διαφοράς Γωνίας από Στόχο
        angleError=angle2goal-(curr_orientation(1)+curr_orientation(3)); %Υπολογισμός Γωνιακού Σφάλματος
        angleErrorN=normaliseInCircle(angleError);
        angleErrorND=rad2deg(angleErrorN);
        if ( distance2goal < 0.5 ) %Έλεγχος Ελάχιστης Απόστασης Από Στόχο
           j=j+1;
           if j < maxj+1  %Αλλαγή Στόχου
               [errorCode]=vrep.simxSetObjectPosition(clientID, h.goal, -1,goalxyz(j,:), vrep.simx_opmode_oneshot_wait);
           else
               disp('prepare to break');   %Τέλος Στόχων
                break
           end
        end
        angleErrorNN=angleErrorN;
        if abs(angleErrorN) > 1                    %Έλεγχος Μεγάλου Γωνιακού Σφάλματος
            angleErrorNN=abs(angleErrorN)/angleErrorN;
        end
        angleErr=double(angleErrorNN);
        update=[angleErr];  %Ενημέρωση Ελεγκτή
        [motorspeeds]=evalfis(update,initfis); %Υπολογισμός Ταχυτήτων Κινητήρων από τον ελεγκτή ασαφούς λογικής
        [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.leftMotor,  motorspeeds(1), vrep.simx_opmode_streaming);%Ενημέρωση ταχυτήτων κινητήρων
        [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.rightMotor, motorspeeds(2), vrep.simx_opmode_streaming);
    end
    disp('break');
    [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.leftMotor,  0, vrep.simx_opmode_oneshot_wait); %Μηδενισμός Κινήτηρων
    [errorCode]=vrep.simxSetJointTargetVelocity( clientID, h.rightMotor, 0, vrep.simx_opmode_oneshot_wait);
    [errorCode, curr_position] = vrep.simxGetObjectPosition(clientID, h.pioneer1, -1, vrep.simx_opmode_buffer); %Ενημέρωση Θέσης
    distance2goal=sqrt( (goalxyz(maxj,1)-curr_position(1))^2+(goalxyz(maxj,2)-curr_position(2))^2 );
    vrep.simxFinish(clientID); %Τέλος Προσοποίωσης
    vrep.delete();
    catch err
    msgString = getReport(err);
        fprintf('error: %s',msgString)
        vrep.simxFinish(clientID);
        vrep.delete();
        disp('Abnormal program termination ! ');
end;
disp('Program end');
```

Εικόνα 44: Κώδικας Κύριου Προγράμματος Απλής Κίνησης στο Χώρο Χωρίς Αποφυγή Εμποδίων

Κομμάτι Κώδικα Απλής Κίνησης στο Χώρο με Αποφυγή Εμποδίων

```
det3=sqrt( (detectedPoint3(1)-curr_position(1))^2+(detectedPoint3(2)-curr_position(2))^2 );
detection3=double(det3);
det4=sqrt( (detectedPoint4(1)-curr_position(1))^2+(detectedPoint4(2)-curr_position(2))^2 );
detection4=double(det4);
if detectionState3~=0 || detectionState4~=0
    detectionLeftSide=sqrt(((detection3)^2)+((detection4)^2));
else
    detectionLeftSide=29;
end
det5=sqrt( (detectedPoint5(1)-curr_position(1))^2+(detectedPoint5(2)-curr_position(2))^2 );
detection5=double(det5);
if detectionState5~=0
    detectionFront=detection5;
else
    detectionFront=29;
end
det6=sqrt( (detectedPoint6(1)-curr_position(1))^2+(detectedPoint6(2)-curr_position(2))^2 );
detection6=double(det6);
det7=sqrt( (detectedPoint7(1)-curr_position(1))^2+(detectedPoint7(2)-curr_position(2))^2 );
detection7=double(det7);
if detectionState6~=0 || detectionState7~=0
    detectionRightSide=sqrt(((detection6)^2)+((detection7)^2));
else
    detectionRightSide=29;
end
```

Εικόνα 45: Γραμμές Κώδικα που πρέπει να εισαχθούν στο προηγούμενο κυρίως πρόγραμμα

Συννημένα Αρχεία

Μαζί με την παρούσα εργασία, παρατίθενται και τα εξής:

- ✓ Φάκελος Mac OS που περιέχει όλα τα απαραίτητα αρχεία για εγκατάσταση σύνδεσης MATLAB – VREP σε περιβάλλον Macintosh
- ✓ Φάκελος Windows που περιέχει όλα τα απαραίτητα αρχεία για εγκατάσταση σύνδεσης MATLAB – VREP σε περιβάλλον Windows
- ✓ Φάκελος test3 που περιέχει κώδικα ελέγχου κίνησης στο χώρο χωρίς εμπόδια (test3_v2.m) και κώδικα ελέγχου κίνησης στο χώρο με αποφυγή εμποδίων (test3.m).
- ✓ Φάκελος Final Script που περιέχει το τελικό πρόγραμμα με τον κώδικα αποφυγής εμποδίων.

Υπόψην ότι τα παραπάνω χρειάζεται να αντιγραφούν στον φάκελο trs που έχει δημιουργηθεί στο υπολογιστή σας, ώστε να καλεστούν οι απαραίτητες ρουτίνες, αλλιώς δεν θα λειτουργήσουν.