Ενισχυτική Μάθηση Για Ρομποτικό Βάδισμα

Walid Soulakis

Ενισχυτική Μάθηση Για Ρομποτικό Βάδισμα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Walid Soulakis

A.M: 2003010119, Email: wsoulakis@hotmail.com

Εξεταστική Επιτροπή:

Ν. Βλάσσης, Επίκουρος Καθηγητής, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης (Επιβλέπων)

Μ. Λαγουδάκης, Επίκουρος Καθηγητής, Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης

Ι. Νικολός, Καθηγητής, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Εικόνα εξωφύλλου: NAO Aldebaran

Ενισχυτική Μάθηση Για Ρομποτικό Βάδισμα

Περίληψη

Στην παρούσα διπλωματική εργασία εξετάζουμε τη δημιουργία ενός συνόλου πρακτόρων που έχουν σκοπό να συντελέσουν μια ομάδα ποδοσφαίρου. Οι πράκτορες λειτουργούν εντός ενός περιβάλλοντος προσομοίωσης που παρέχει το Webots της εταιρίας Cyberbotics. Στην εργασία παρουσιάζονται δύο μεγάλα κεφάλαια. Στο πρώτο μέρος παρουσιάζεται ο διαγωνισμός του RoboCup και ο τρόπος λειτουργίας του Webots και του αντίστοιχου ρομπότ NAO της Aldebaran Robotics και η συμετοχή μας στο RoboCup. Το ρομπότ που χρησιμοποιήσαμε είναι το μοντέλο Nao V3R της Aldebaran Robotics. Στο δεύτερο μέρος πραγματευόμαστε τον έλεγχο κίνησης δίποδου ρομπότ ,χρησιμοποιώντας διάφορες μεθόδους ενισχυτικής μάθησης. Το ρομπότ που χρησιμοποιήσαμε είναι το μοντέλο Nao V2 της Aldebaran Robotics. Η τελική κίνηση που δημιουργήσαμε είναι ένα περπάτημα.

Ευχαριστίες

Πρώτα απ'όλα θέλω να ευχαριστήσω τον καθηγητή μου Νίκο Βλάσση, ο οποίος με βοήθησε να αποκτήσω καινούριες γνώσεις πάνω στο αντικείμενο της ενισχυτικής μάθησης και ρομποτικής. Από την αρχή της συνεργασίας μας με τις συμβουλές του κατάφερα να πιστέψω στον εαυτό μου και να πετύχω τα καλύτερα αποτελέσματα στην διπλωματική μου. Επίσης θα ήθελα να ευχαριστήσω τον κ.Λαγουδάκη και τον κ.Νικολό, μέλη της εξεταστικής επιτροπής.
Μεγάλο ευχαριστώ στον Γιώργο, για την αμέριστη βοήθεια και συμβουλές που μου προσέφερε κατά την διάρκεια της διπλωματικής μου και το RoboCup.
Στο σημείο αυτό θα ήθελα να ευχαριστήσω την οικογένειά μου (την μαμά και τα αδέλφια μου, σας αγαπώ πολύ!), οι οποίοι ήταν πάντα δίπλα μου και με στήριζαν.
Ευχαριστώ επίσης τα κολλητάρια (Ευτύχη, Νίκο, Στέφανο) με τους οποίους πέρασα πολύ ωραίες στιγμές κατά την διάρκεια των σπουδών μου. Ευχαριστώ τον Ράμι για την συνεχή υποστήριξη του (ευχαριστώ για το ελικόπτερο κουμπάρε!, θα το φτιάξω και αυτό σε RL που θα πάει!). Ευχαριστώ τον Γιάννη που ήταν ο πρώτος φίλος που απέκτησα στην αρχή των σπουδών μου. Ευχαριστώ τους φίλους μου (Αντώνης (Houssan), Βιβή, Ελευθερία, Δημήτρης, Ιφιγένια, Ιωάννα, Κατερίνα, Μαριάννα, Ραχήλ, Σήφης), ευχαριστώ και όλους τους υπόλοιπους φίλους μου..
Τέλος αυτή η διπλωματική εργασία είναι αφιερωμένη στην μνήμη του μπαμπά μου, ο οποίος πάντα ήθελε να με δει μηχανικό.

Εισαγωγή

Η ιδέα να δημιουργήσουν οι άνθρωποι μηχανές που να τους υπηρετούν, να τους διασκεδάζουν, να τους βοηθούν ή να τους αντικαθιστούν στις εργασίες τους χάνεται βαθιά στο χρόνο, καθώς ήδη από το 400 π.Χ περίπου, έχουμε αναφορές για μηχανές που λειτουργούσαν αυτόνομα (αυτόματα), αλλά έπρεπε να περιμένουμε μέχρι το 1920, οπότε ο Τσέχος συγγραφέας Karel Capek εισάγει τη λέξη ρομπότ στο θεατρικό του έργο R.U.R. (Rossum's Universal Robots) και εν αγνοία του δίνει το όνομά της σε μια από τις πιο ενεργές ερευνητικές περιοχές της σύγχρονης επιστήμης, τη ρομποτική.

Η ρομποτική σαν πεδίο έρευνας ξεκίνησε από απλούς ρομποτικούς βραχίονες και σήμερα έχει να "επιδείξει" μια πληθώρα διαφορετικών τύπων ρομπότ, όπως είναι τα βιομηχανικά, τα ιπτάμενα, τα υποθαλάσσια, τα οικιακά και πολλά άλλα. Το επόμενο βήμα είναι τα "έξυπνα ρομπότ", δηλαδή τα ρομπότ εκείνα που θα λειτουργούν αυτόνομα και θα μαθαίνουν από τα λάθη τους και εδω εισάγεται η έννοια της τεχνητής νοημοσύνης.

Η τεχνητή νοημοσύνη αποτελεί σήμερα ένα απο τα σημαντικότερα ερευνητικά πεδία σε παγκόσμιο επίπεδο. Τα έμπειρα συστήματα, τα νευρωνικά δίκτυα, η μηχανική μάθηση, οι εξελικτικοί αλγόριθμοι, οι ευφυείς πράκτορες κλπ. Η ιδέα του αυτόνομου πράκτορα (autonomous agent), δηλαδή η οντότητα εκείνη, η οποία έχει στόχους και αναλαμβάνει πρωτοβουλίες για να φέρει σε πέρας, ενώ μπορεί να εμφανίζεται σε ένα περιβάλλον και να μαθαίνει να κινείται σε αυτό , δημιούργησε έναν άλλο επιστημονικό χώρο. Στην παρούσα διπλωματική εργασία παρουσιάζονται τα εξής κεφάλαια:

Στο πρώτο κεφάλαιο περιγράφεται ο διαγωνισμός του RoboCup, και οι διάφορες κατηγορίες του.

Στο δεύτερο κεφάλαιο γίνεται μια μικρή εισαγωγή του ρομπότ που χρησιμοποιείται στα πειράματα (NAO) και του λογισμικού Webots στον οποίο γίνεται η αποποίηση του ελεγκτή μας.

Το τρίτο κεφάλαιο περιέχει μια μικρή εισαγωγή στην Ενισχυτική Μάθηση, και τους αλγορίθμους MCEM και PoWER που χρησιμοποιηθήκαν στην μάθηση.

Στο τέταρτο κεφάλαιο παρουσιάζεται συνοπτικά η ανάλυση του κατά Fourier και ο τρόπος που χρησιμοποιήθηκε στην εργασία, και η παραμετροποίηση του συστήματος για να μπορεί να χρησιμοποιηθεί η Ενισχυτική Μάθηση.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα που έδωσε η προσομοίωση του συστήματος.

Περιεχόμενα

Κεφάλαιο 1

RoboCup

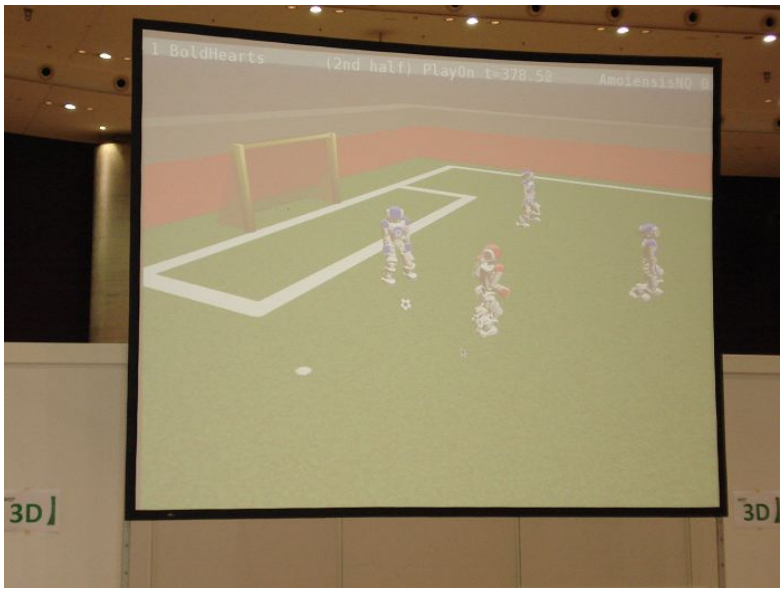
Το RoboCup είναι μια διεθνής οργάνωση που έχει ως σκοπό την εκπαίδευση και την έρευνα σε θέματα τεχνητής νοημοσύνης και ρομποτικής. Για την επίτευξη του στόχου αυτού χρησιμοποιήθηκε ένα τυποποιημένο πρόβλημα το οποίο μπορεί να προσεγγιστεί με μία μεγάλη γκάμα τεχνολογιών και μπορεί να αποτελέσει βάση για μάθηση μέσω έργων και πειραματισμού.

Για τον σκοπό αυτό το πρώτο πρόβλημα που επιλέχτηκε από την ομοσπονδία του RoboCup ήταν το παιχνίδι του ποδοσφαίρου. Το ποδόσφαιρο είναι απαιτητικό παιχνίδι σε διάφορα επίπεδα και στηρίζεται σε αρχές όπως : αυτόνομοι πράκτορες, συνεργασία πολλαπλών πρακτόρων, δημιουργία στρατηγικής, λήψη αποφάσεων σε πραγματικό χρόνο, αντιμετώπιση του θορύβου στους σένσορες και φυσικά κινηματικά προβλήματα ρομποτικής. Η φύση του προβλήματος είναι τέτοια που αποτελεί πρόκληση σε όλους τους βαθμούς δυσκολίας.

1.1 Robocup Simulation

Στην κατηγορία αυτή οι αγώνες γίνονται σε εικονικό περιβάλλον όπου ακολουθούνται δυο διαφορετικές προσεγγίσεις. Στις κατηγορίες 2d/mixed reality τα εικονικά ρομπότ είναι όμοια με αυτά των κατηγοριών Small size robot league και middle size robot league και κάθε ομάδα αποτελείται από 11 πράκτορες που βρίσκονται σε επικοινωνία με έναν server. Ο server γνωρίζει τις θέσεις των παιχτών και της μπάλας και καθορίζει την λειτουργία του παιχνιδιού. Η ευκολία της κίνησης σε αυτού του είδους τα ρομπότ λύνει προβλήματα ισορροπίας και δίνει την δυνατότητα να δοθεί έμφαση σε ομαδικές στρατηγικές όπως αυτές που πραγματοποιούνται στο πραγματικό ποδόσφαιρο.

Στην κατηγορία του 3d simulation τα μοντέλα είναι αντίγραφα του NAO που χρησιμοποιείται και στο Standard Platform League, ενώ κάθε ομάδα αποτελείται από 3 ρομπότ (2 επιθετικούς, 1 τερματοφύλακα). Αντίθετα με το 2d simulation περισσότερο βάρος δίνεται σε χαμηλού επιπέδου λειτουργίες όπως περπάτημα, οπτική αναγνώριση περιβάλλοντος, κλωτσιά, αποκρούσεις κλπ. Γενικά οι ομάδες καλούνται να επιλύσουν τα ίδια προβλήματα με τα πραγματικά ρομπότ και αυτό καθιστά τους αλγόριθμους εφαρμόσιμους και στην πραγματικότητα.



Σχήμα 1.1: Αγώνας 3d simulation

1.2 Small Size League

Σε αυτήν την κατηγορία τα ρομπότ είναι κυλινδρικά με διάμετρο 18εκ και ύψος 15 εκ τα ρομπότ είναι τροχοφόρα ενώ ενδιαφέρον προκαλεί το γεγονός ότι δεν έχουν αισθητήρια όργανα και η α-ναγνώριση της θέσης τους και της θέσης της μπάλας υπολογίζεται από την εικόνα που λαμβάνει μια κάμερα πάνω από το γήπεδο. Για αναγνώριση των ρομπότ από την κάμερα κάθε ένα έχει ένα διαφορετικό συνδυασμό 4 χρωμάτων όπου αποτελεί την ταυτότητα του. Η κάθε ομάδα έχει 5 ρομπότ συμπεριλαμβανομένου του τερματοφύλακα, ενώ ο αγώνας διεξάγεται σε γήπεδο 6m x 4m.

1.3 Middle Size League

Δυο ομάδες των 5 ρομπότ παίζουν σε ένα γήπεδο 18m x12m και κάθε ρομπότ είναι εξοπλισμένο με ένα υπολογιστή και σένσορες ώστε να υπολογίζει την συμπεριφορά του σαν παίχτης μέσα στο γήπεδο. Η επικοινωνία μεταξύ των παιχτών γίνεται ασύρματα αλλά απαγορεύεται η επικοινωνία με ανθρώπους της ομάδας. Τα ρομπότ αναπτύσσουν μεγάλες ταχύτητες και έχουν μεγλη δύναμη για "κλώτσημα" της μπάλας πράγμα που κάνει το παιχνίδι πιο θεαματικό αλλά και πιο απαιτητικό από πλευράς κατασκευής και προγραμματισμού των ρομπότ. Αξίζει να σημειωθεί ότι μέχρι και το 2008 για την αναγνώριση του γηπέδου είχαν χρησιμοποιηθεί μόνο διαφορετικού χρώματος τέρματα ενώ από το 2009 και έπειτα τα τέρματα θα είναι λευκά, κάτι που φέρνει τον διαγωνισμό πιο κοντά σε πραγματικές συνθήκες.

Σχήμα 1.2: Small size League

Σχήμα 1.3: Middle size League

1.4 Standard Platform League

Σε αυτήν την κατηγορία όλες οι ομάδες χρησιμοποιούν τον ίδιο τύπο ρομπότ το οποίο κατασκευάζεται από τρίτη ανεξάρτητη εταιρία. Μέχρι το 2008 επίσημο ρομπότ ήταν το Aibo της Sony(τετράποδο) ενώ το 2008 χρησιμοποιήθηκε το ανθρωπόμορφο ρομπότ της Aldebaran με την ονομασία Nao. Οι ομάδες επικεντρώνονται αποκλειστικά στο κομμάτι του προγραμματισμού και η τυποποίηση των ρομπότ που

Σχήμα 1.4: Aldebaran Nao, Graz 2009

χρησιμοποιούνται εξασφαλίζει την νίκη σε αυτούς που έχουν γράψει τους πιο αποτελεσματικούς αλγόριθμους.

1.5 Humanoid League

Στο humanoid league υπάρχουν 2 υποκατηγορίες με βάση το μέγεθος των ρομπότ, από 30-60 εκ για Kid Size και από 100-160 εκ για Teen Size. Τα ρομπότ όπως δηλώνει το όνομα της κατηγορίας είναι ανθρωπόμορφα με τα ίδια αισθητήρια όργανα που είναι διαθέσιμα και στον άνθρωπο. Κάθε ομάδα κατασκευάζει τα δικά της ρομπότ δίνοντας βαρύτητα τόσο στην μηχανική υπεροχή των ρομπότ όσο και στην υπεροχή από πλευράς προγραμματισμού.

1.6 Rescue Robots

Στον ετήσιο διαγωνισμό του RoboCup εκτός από το ποδόσφαιρο ομάδες διαγωνίζονται στο RoboCup Rescue και RoboCup@Home. Το RoboCup Rescue είναι ένας νέος τομέας όπου έχει σκοπό τις αποστολές εντοπισμού και διάσωσης σε μεγάλης κλίμακας καταστροφές και σε περιβάλλοντα που δεν είναι προσβάσιμα στον άνθρωπο. Κατά την διάρκεια του διαγωνισμού τα ρομποτικά οχήματα τοποθετούνται σε ειδικά διαμορφωμένες πίστες που μιμούνται δυσπρόσιτα σημεία και απαιτούν από τα οχήματα να εντοπίσουν και να διασώσουν "θύματα".

ο RoboCup Rescue πραγματοποιείται και σε simulation μορφή δίνοντας την δυνατότητα σε ομάδες που δεν έχουν τους οικονομικούς πόρους να κατασκευάσουν ένα όχημα, ώστε να συμμετάσχουν στον διαγωνισμό.

Ο πιο πρόσφατος διαγωνισμός που προστέθηκε στο RoboCup είναι το RoboCup@home Competition. Το @Home έχει στόχο να εισάγει την ρομποτική τεχνολογία στο σπίτι εκτελώντας βοηθητικές για

Σχήμα 1.5: Humanoid League

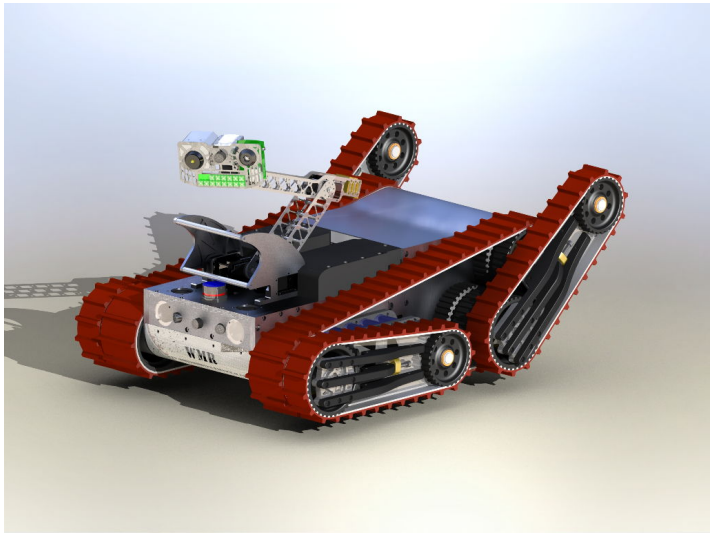
τον άνθρωπο εργασίες. Με διάφορες δοκιμασίες βαθμολογείται η ικανότητα των ρομπότ να κινούνται σε άγνωστο περιβάλλον κάνοντας κάποια λειτουργία σχετική με την καθημερινότητα διατηρώντας φιλική προσέγγιση προς τον άνθρωπο.

1.7 Robotstadium

Εκτός του RoboCup υπάρχει και το Robotstadium ,ένας διαγωνισμός που λαμβάνει χώρα μέσω internet όλο το χρόνο. Επιτρέπει μέσω του λογισμικού Webots να υλοποιεί το δικό του κώδικα και συνεπώς να επιτρέπει να παίζουν τα ρομπότ εντός προσομοίωσης.

Το Robotstadium έχει το χαρακτηριστικό να προσομοιώνει άριστα τον αγώνα βασικής πλατφόρμας του RoboCup. Αυτός ο διαγωνισμός δίνει την επιλογή μεταξύ μερικών γλωσσών προγραμματισμού, συνεπώς όποιος μπορεί να χρησιμοποιήσει τη C++, Java, Urbi μπορεί να λάβει μέρος στο διαγωνισμό.

Το Robotstadium οργανώνει αγώνες κάθε βράδυ μεταξύ διαφόρων διαγωνιζόμενων που έχουν «ανεβάσει» πρώτα τον κώδικά τους στην ιστοσελίδα. Ένας διαχωρισμός που λέγεται (Hall of fame)



Σχήμα 1.6: Rescue Robot

ενημερώνει τους 8 πρώτους να λάβουν μέρος στους τελικούς του RoboCup.

1.8 Το Όραμα

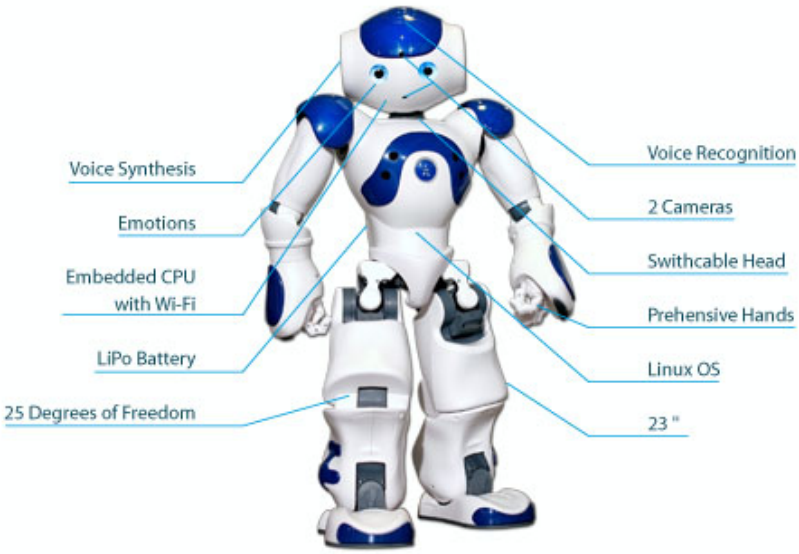
Το όραμα του RoboCup είναι: **"μέχρι το έτος 2050 να δημιουργηθεί μια ρομποτική ποδοσφαιρική ομάδα αυτόνομων πρακτόρων που θα μπορέσει να κερδίσει σε ένα αγώνα την εκάστοτε πρωταθλήτρια κόσμου"**.

Κεφάλαιο 2

NAO και Webots

2.1 Τι είναι το NAO

Το Nao είναι ένα ανθρωποειδές ρομπότ με ύψος περίπου 60 εκατοστών. Κατασκευάστηκε από την Aldebaran Robotics, μια γαλλική εταιρεία που εδρεύει στο Παρίσι. Το σχέδιο κατασκευής του ξεκίνησε το 2005 και, σύμφωνα με το δημιουργό του Bruno Maisonnier, κατάφερε να είναι διαθέσιμο στο κοινό σε μια προσιτή τιμή, ένα ανθρωποειδές ρομπότ με μηχανικά, ηλεκτρονικά και νοητικά χαρακτηριστικά σχήμα (2.1).



Σχήμα 2.1: Το ρομπότ NAO

Το Nao χαρακτηρίζεται από 25 βαθμούς ελευθερίας, δυο κάμερες, πολλούς αισθητήρες και επίσης από σονάρ (ηχητικό εντοπιστή) για να μπορεί να κινείται με μεγάλη ακρίβεια. Έχει επίσης τη δυνατότητα με ασύρματη σύνδεση $(WiFi)$ να επικοινωνεί με άλλα εργαλεία.

Αλλά το πιο ενδιαφέρον για τους ερευνητές είναι ότι το Nao είναι πλήρως ικανό να προγραμματι-

στεί χάρη σε συγκεκριμένα εργαλεία όπως $Choregraphe$, μια γραφική εφαρμογή της $AldebaranRobotics$, ή με το Webots που μας επιτρέπει να δουλεύουμε με προσομοίωση και να αποστέλλουμε τον κώδικα κατευθείαν στο ρομπότ.

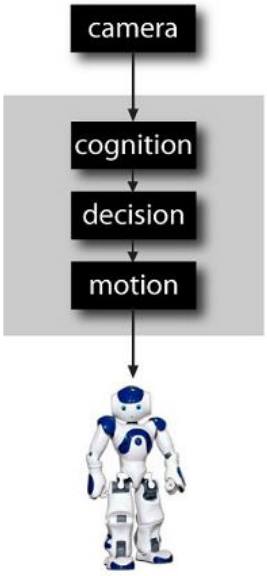
2.2 Τι είναι το Webots

Ο προσομοιωτής Webots αποτελεί τα παρακάτω στοιχεία:

- 1. Κόσμος (World) του Webots όπου ορίζεται ως ένα η παραπάνω 3D ρομπότ και το δικό τους περιβάλλον.
- 2. Ελεγκτής (Controller) των παραπάνω ρομπότ.
- 3. Ένα προαιρετικό Supervisor.

2.3 Συμπεριφορά του NAO

Με τον όρο συμπεριφορά ορίζουμε τις αποφάσεις που λαμβάνει το ρομπότ με βάση τα δεδομένα εισόδου. Η διαδικασία της απόφασης χωρίζεται σε πολλά διαφορετικά μέρη, στο δικό μας σύστημα η συμπεριφορά μας ορίζεται γύρω από τα δεδομένα που δίνει ο αισθητήρας της κάμερας. Με κάθε νέο καρέ που στέλνει το $WebCam$ καλείται μια σειρά από ρουτίνες ώστε να προσδιορίσουν την δράση του ρομπότ, σχήμα (2.2).



Σχήμα 2.2: Διάγραμμα Αποφάσεων

Διακρίνουμε ότι τα τρία βασικά μέρη του αλγορίθμου είναι:

- 1.Cognition, είναι η ανάγνωση του περιβάλλοντα χώρου.
- 2.Decision, η απόφαση με βάση τα δεδομένα που έχουμε για το περιβάλλον αλλά και τον στόχο μας.
- 3.Motion, ο προγραμματισμός για το πώς θα κινηθεί το ρομπότ αφού έχουμε λάβει την απόφαση για το τι πρέπει να γίνει.

Τώρα θα μελετήσουμε τη συμπεριφορά του ρομπότ όταν εντοπίσει την μπάλα. Πρώτα, θα αναβαθμίσει τρεις σημαντικές μεταβλητές για να μπορεί να κινείται σωστά.

- 1)$BallDist$: Το οποίο είναι η απόσταση μεταξύ της μπάλας και του ρομπότ.
- 2)$BallDir$: Το οποίο είναι η γωνία σε rad μεταξύ της μπάλας και του ρομπότ. Εάν η μπάλα είναι προς τα δεξιά του ρομπότ, η γωνία είναι θετική, αν είναι προς τα αριστερά είναι αρνητική.
- 3)$GoalDir$: Το οποίο είναι η γωνία σε rad μεταξύ του τέρματος και του ρομπότ και λειτουργεί με τον ίδιο τρόπο με το $BallDir$.

2.3.1 Τι είναι το BallDist:

Μπορούμε να χρησιμοποιήσουμε τη θέση του κεφαλιού του ρομπότ για να υπολογιστεί η απόσταση μεταξύ αυτού και της μπάλας, επιστρέφοντας έναν απλό τριγωνομετρικό υπολογισμό,σχήμα (2.3).

Μπορούμε να έχουμε πρόσβαση στη θέση από οποιαδήποτε servo μηχανή του ρομπότ, καλώντας τη μέθοδο $getServoPos$. Και σε αυτή την περίπτωση πρέπει να απασχολήσουμε μια που λέγεται $HeadPitch$.

Επιπλέον γνωρίζοντας τη γωνία (α) που σχηματίζει το κεφάλι του ρομπότ κοιτάζοντας την μπάλα και γνωρίζοντας την απόσταση (h) που υπάρχει μεταξύ του εδάφους και της κάμερας του ρομπότ, που είναι περίπου το μέγεθος του, απομένει να υπολογιστεί ένας απλός υπολογισμός τριγωνομετρίας για να προκύψει η τιμή του $BallDist$.

2.3.2 Τι είναι το BallDir:

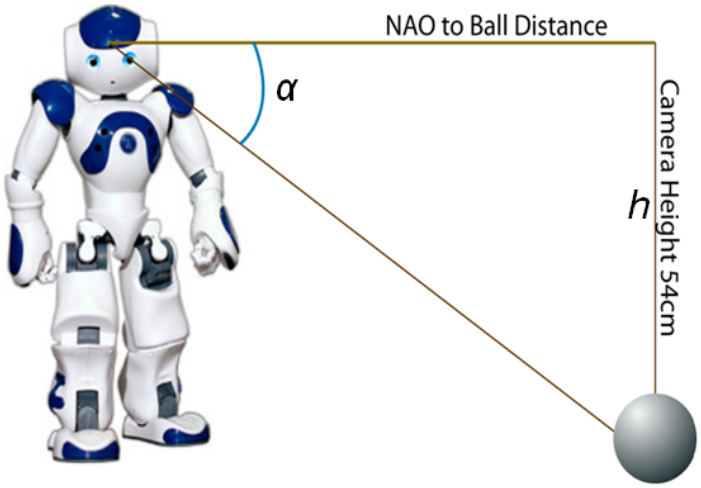
Για να ορισθεί η κατεύθυνση της γωνίας μεταξύ του ρομπότ και της μπάλας, ομοίως με την απόσταση, μόνο που αυτή τη φορά ο υπολογισμός είναι πιο απλός,σχήμα (2.4).

Πράγματι, για να ορισθεί η κατεύθυνση της μπάλας, είναι αρκετό να γνωρίζουμε τη θέση του κεφαλιού (από τη θέση του $HeadYaw$ servo) για να πάρουμε τη γωνία ανοίγματος.

2.3.3 Τι είναι το GoalDir:

Λαμβάνοντας υπόψη την κατεύθυνση του τέρματος, μπορούμε να υπολογίσουμε τη γωνία του κεφαλιού του $HeadYaw$ servo. Όμως παρατηρείται μια μεγάλη διάφορα, στην κατάσταση $"FoundBall"$ η μπάλα πάντα εντοπίζεται από το ρομπότ, όμως δεν είναι απαραίτητο να εντοπίζεται το τέρμα. Γι'αυτό είναι απαραίτητο να δράσουμε διαφορετικά ,όταν το τέρμα δεν είναι στο οπτικό πεδίο του ρομπότ.

Για να επιλυθεί αυτό το πρόβλημα, αλλάζουμε την τιμή του $GoalDir$ σύμφωνα με τις κινήσεις που κάνει το ρομπότ.



Σχήμα 2.3: Ball Distance

Σχήμα 2.4: Ball Direction

2.4 Η Όραση του NAO μέσω του Webots

Το Webots έχει το επονομαζόμενο $Camera$ node, που χρησιμοποιείται για να διαχειριστεί την κάμερα που είναι ενσωματωμένη στο ρομπότ. Αυτή η κάμερα έχει μερικές συναρτήσεις που επεξεργάζονται την εικόνα που παρατηρείται μέσω της κάμερας. Μερικές από τις συναρτήσεις είναι:

$Camera.get.image$: το οποίο επιστρέφει έναν πινάκα ακέραιων που αναπαριστούν την εικόνα που παρατηρεί το ρομπότ. Η εικόνα κωδικοποιείται σαν μια σειρά 3 bytes που αντιπροσωπεύουν το RGB ενός Pixel. Τα Pixels αποθηκεύονται σε οριζόντιες γραμμές από πάνω αριστερά της εικόνας μέχρι κάτω δεξιά.
Οι συναρτήσεις που μπορούν να χρησιμοποιηθούν κατευθείαν προσεγγίζοντας το pixel RGB είναι:

- $camera.image.get.red$
- $camera.image.get.green$
- $camera.image.get.blue$

Χρησιμοποιώντας αυτές τις συναρτήσεις θα μπορούμε να αναλύσουμε τι βλέπει το ρομπότ. Πράγματι, στον κώδικα $Java$, υπάρχει μια κλάση που ονομάζεται $Camera.java$ και χρησιμοποιείται για την ανάλυση της εικόνας. Αυτή η κλάση περιέχει κυρίως δυο μεθόδους:
Η πρώτη, που ονομάζεται $findColorBlob$ και χρησιμοποιείται για την ανεύρεση ενός αντικείμενου συγκεκριμένου χρώματος, που ορίζεται από τα RGB συστατικά του. Αυτή η συνάρτηση επαναλαμβάνει συνεχώς τον πίνακα των ακέραιων που επιστρέφει η $camera.get.image$ και θα συγκρίνει κάθε σύνολο 3 bytes αν αντιστοιχούν στα απαιτούμενα. Εάν συμβαίνει αυτό δίνει μια μεταβλητή x και y του επιθυμητού αντικειμένου της εικόνας, αλλιώς όταν το αντικείμενο δεν μπορεί να βρεθεί, επιστρέφει την τιμή που ονομάζεται UNKNOWN.

Όσον αφόρα τη δεύτερη μέθοδο, που ονομάζεται $ProcessImage$, χρησιμοποιείται περισσότερο για να ορίσει τη θέση πάνω στην εικόνα της μπάλας και του τέρματος. Το $ProcessImage$ χρησιμοποιεί τη μέθοδο $findColorBlob$ για να έχει αυτές τις πληροφορίες. Θα καλέσει αυτή τη μέθοδο με εισόδους τα στοιχεία RGB του χρώματος της μπάλας και του τέρματος.

Στο σχήμα (2.5) και (2.6) παριστάνω δυο διαγράμματα ροής της συμπεριφοράς του (Παίκτη και Τερματοφύλακα).

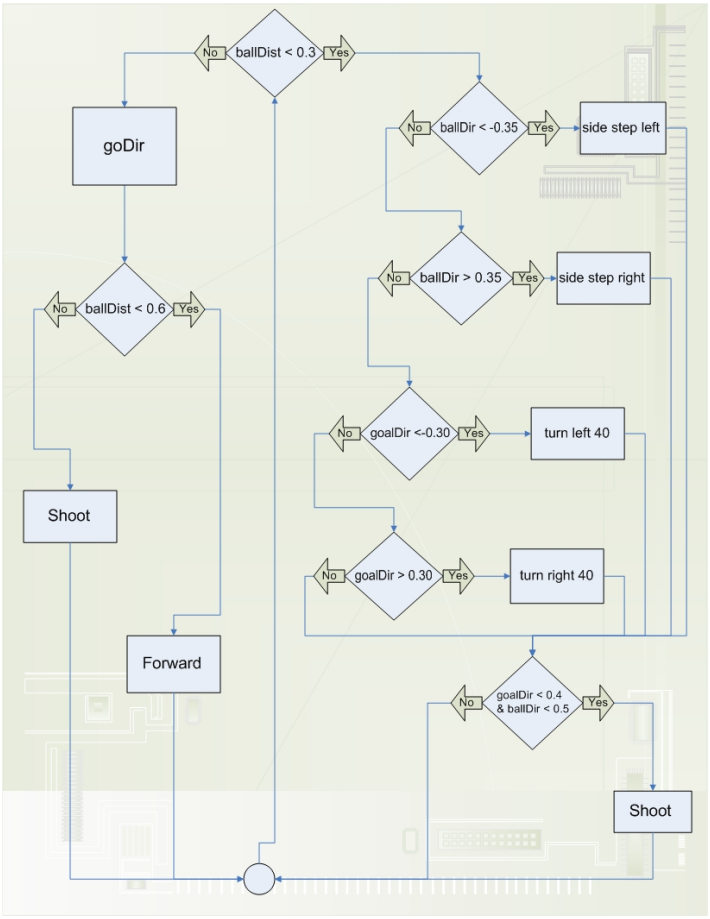
2.5 Τι είναι το Worlds

Ένας κόσμος στο Webots είναι μια τρισδιάστατη περιγραφή των ιδιοτήτων των ρομπότ και του περιβάλλοντος τους. Περιέχει μια περιγραφή του κάθε αντικειμένου: την θέση του, τον προσανατολισμό του, την γεωμετρία του, την εμφάνιση του (όπως χρώμα ή φωτεινότητα), φυσικές ιδιότητες, τύπος του αντικειμένου κλπ. Οι κόσμοι οργανώνονται ως ιεραρχικές δομές όπου αντικείμενα μπορούν να περιέχουν άλλα αντικείμενα (όπως στο VRML97). Για παράδειγμα ένα ρομπότ μπορεί να περιέχει δυο τροχούς, ένα αισθητήρα απόστασης και ένα σέρβο που το ίδιο περιέχει μια κάμερα, κλπ.

Ένα αρχείο world δεν περιέχει τον κώδικα του ελεγκτή των ρομπότ, απλώς καθορίζει το όνομα του ελεγκτή που απαιτείται για κάθε ρομπότ. Οι κόσμοι σώζονται σε αρχεία ($.wpt$). Τα αρχεία αυτά αποθηκεύονται στον υποφακελο «worlds» της κάθε εργασίας $Project$.

2.6 Τι εναι το Controller

Ένας ελεγκτής είναι ένα ηλεκτρονικό πρόγραμμα που ελέγχει ένα ρομπότ που ορίζεται σε ένα αρχείο world. Οι ελεγκτές μπορούν να γραφτούν σε οποιαδήποτε από τις γλώσσες προγραμματισμού που υποστηρίζονται από το Webots : C, C++, Java, Urbi, Python, Matlab. Όταν μια προσομοίωση ξεκινήσει, το Webots εκκινεί τους καθορισμένους ελεγκτές, τον κάθε ένα ως μια ξεχωριστή διαδικασία, και συνδέει τις διαδικασίες του ελεγκτών με τα προσομοιωμένα ρομπότ. Σημειώστε ότι αρκετά ρομπότ μπορούν να χρησιμοποιούν τον ίδιο κώδικα ελεγκτή, παρόλα αυτά για κάθε ρομπότ θα ξεκινήσει μια δική του ξεχωριστή διαδικασία.



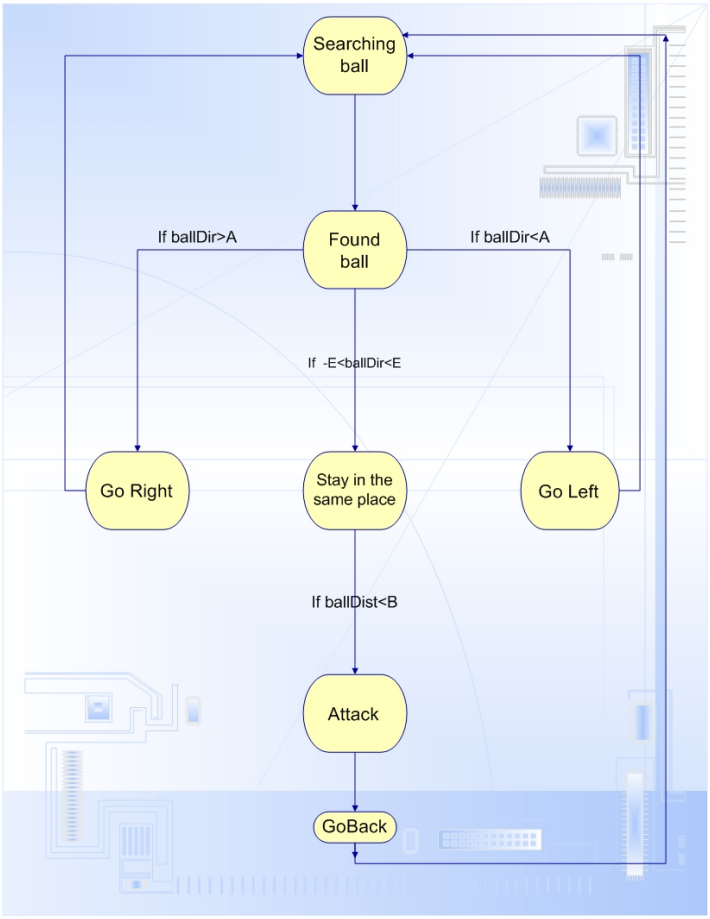
Σχήμα 2.5: Συμπεριφορά ενός Παίκτη

Μερικές γλώσσες προγραμματισμού χρειάζονται να κάνουν compile(C, C++) άλλες γλώσσες. Χρειάζονται λογισμική μετάφραση των (Urbi, Python, Matlab) και άλλες χρειάζονται και τα δυο (compile, λογισμική μετάφραση) όπως η (Java). Οι ελεγκτές των (Urbi, Python, Matlab) μεταφράζονται από τα αντίστοιχα συστήματα run-time (τα οποία πρέπει να είναι ήδη εγκατεστημένα). Ο ελεγκτής της Java πρέπει να υποστεί compile σε κώδικα byte (αρχεία .class ,ή .jar) και μετά να μεταφραστούν από μια Java Virtual Machine.

Τα αρχεία source και τα αρχεία binary του κάθε ελεγκτή αποθηκεύονται μαζί σε ένα φάκελο ελεγκτή. Αυτός ο φάκελος βρίσκεται στον υποφάκελο Controllers της κάθε εργασίας Webots.

2.7 Τι είναι το Supervisor

Ο supervisor είναι ένας προνομιακός τύπος ρομπότ που μπορεί να εκτελέσει διαδικασίες που κανονικά μπορούν να εκτελεστούν μονό από ένα ανθρώπινο χειριστή και όχι από ένα πραγματικό ρομπότ. Ο supervisor κανονικά συνδέεται με ένα πρόγραμμα ελεγκτή το οποίο μπορεί επίσης να γραφτεί σε οποιαδήποτε από τις παραπάνω γλώσσες προγραμματισμού που αναφέρθηκαν. Όμως σε αντίθεση με



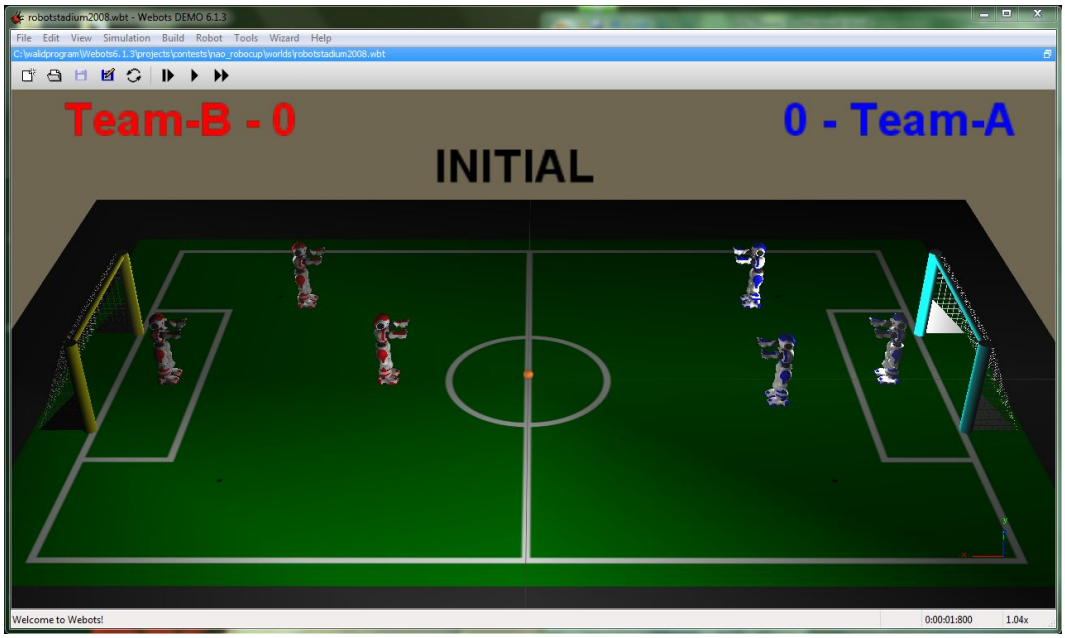
Σχήμα 2.6: Συμπεριφορά ενός Τερματοφύλακα

έναν κανονικό ελεγκτή ρομπότ, ο ελεγκτής του supervisor θα έχει πρόσβαση σε προνομιούχες διαδικασίες. Αυτές συμπεριλαμβάνουν έλεγχο προσομοίωσης, για παράδειγμα μετακίνηση των ρομπότ σε μια τυχαία θέση, να φτιαχτεί ένα βίντεο της προσομοίωσης κλπ.

Μόνο ένας supervisor κατεβαίνει στον αγώνα, αποτελεί το διαιτητή του αγώνα που ορίζει τις θέσεις των ρομπότ σε κάθε έναρξη αγώνα, μετρά τη βαθμολογία, και ελέγχει το χρόνο. Επίσης, αντικαθιστά την μπάλα στο γήπεδο όταν αυτή έχει ξεφύγει. Όλες αυτές οι διαδικασίες περιέχονται στο αρχείο που ονομάζεται $nao.soccer.supervisor.c$. Επιπλέον, ο Supervisor στέλνει δεδομένα στα ρομπότ κάθε 500 ms, τα οποία περιέχουν την κατάσταση του παιχνιδιού, βαθμολογία, υπόλοιπος χρόνος του ημιχρόνου κλπ. Όλες αυτές οι πληροφορίες περιέχονται στο αρχείο που ονομάζεται $RoboCupGameControlData.h$.

2.8 Το Κύριο Παράθυρο:Menus και Buttons

Στο κύριο παράθυρο έχουμε 8 διαφορετικά Menus: File, Edit, View, Simulation, Build, Tool, Wizard και help.



Σχήμα 2.7: Το Robotstadium αποτελεί ένα world του Webots

Τα προηγούμενα Menus μας επιτρέπουν να έχουμε τον πλήρη έλεγχο στον κόσμο ή στα αντικείμενα και έχουν και τη δυνατότητα ελέγχου μέχρι και του κώδικα, καθώς και την δυνατότητα αλλαγής οποιουδήποτε μέρους της εργασίας που εφαρμόζουμε. Στο κύριο παράθυρο μπορούμε να παρατηρήσουμε 3 διαφορετικά τμήματα:

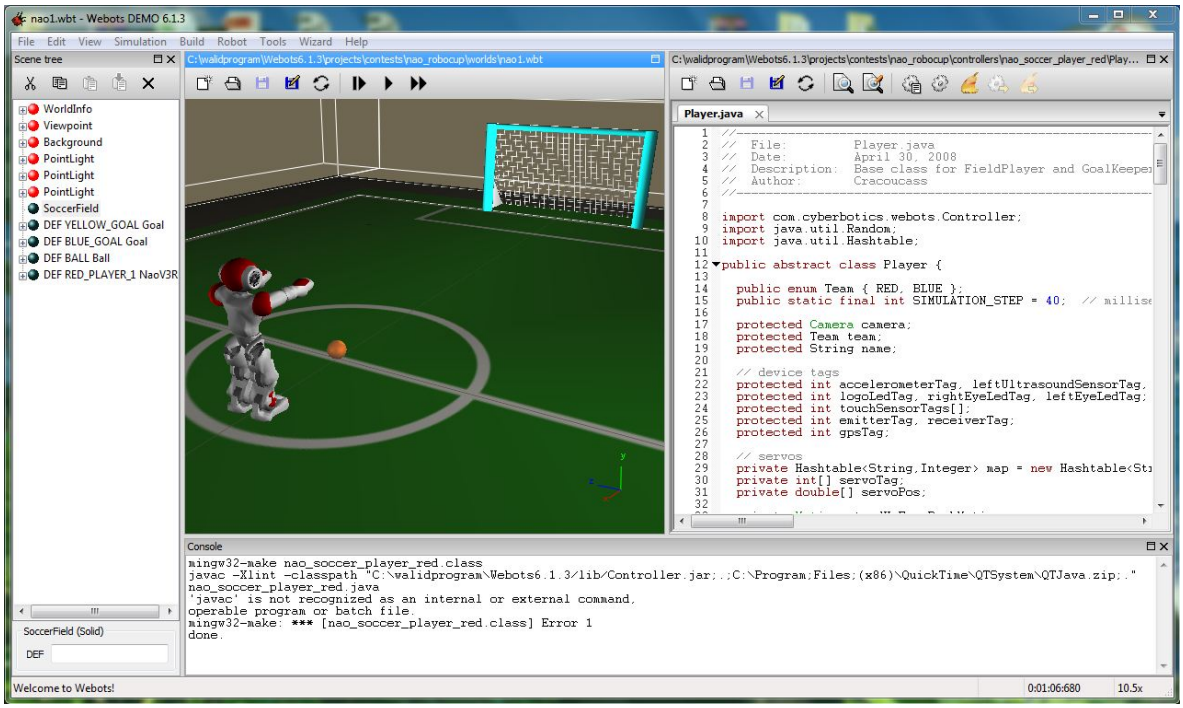
- 1. Sence Tree Window.
- 2. Source Code Editor.
- 3.Console.

2.9 Motion Editor

Ο διαχειριστής κίνησης αποδεικνύει ότι είναι ένα πολύ καλό εργαλείο που μπορεί να δημιουργεί κινήσεις εκτοπίσματος του NAO. Μάλιστα αυτό θα μας επιτρέψει να δούμε απευθείας στην οθόνη τι συμβαίνει όταν κάθε servo μηχανή κινείται.

Ο Επεξεργαστής κίνησης συνδέεται κατευθείαν με τη μηχανή προσομοίωσης κι έχει μερικά πλεονεκτήματα.
Πρώτα, διευκολύνει το σχεδιασμό αληθοφανών κινήσεων .
Δεύτερον, είναι δυνατός ο σχεδιασμός κινήσεων που αλληλεπιδρούν με το περιβάλλον.

Ο Διαχειριστής Κίνησης είναι ένα εργαλείο που εμφανίζεται στο Webots 6.0.0. Μπορεί να χρησιμοποιηθεί για να σχεδιαστούν οι κινήσεις των αρθρώσεων του ρομπότ, όπως τα ανθρωποειδή



Σχήμα 2.8: Το κύριο παράθυρο του Webots

κλπ. Οι κινήσεις μπορούν μετά να αποθηκευτούν σε αρχεία τύπου $.motion$ κι έπειτα μπορούν να αναπαραχθούν στον ελεγκτή κατά τη διάρκεια της προσομοίωσης.

Το $Motionfile$ ουσιαστικά αποτελεί μια ακολουθία γεγονότων η οποία περιγράφει πλήρως μια συγκεκριμένη κίνηση μέσα σε ένα δεδομένο χρονικό διάστημα. Το χρονικό διάστημα στο οποίο εκτελείται η κίνηση χωρίζεται σε επιμέρους μικρότερα. Σε κάθε ένα από τα επιμέρους διαστήματα υπάρχει μια συγκεκριμένη τιμή γωνίας για κάθε μια από τις 25 αρθρώσεις του Robot.

Κεφάλαιο 3

Ενισχυτική Μάθηση

3.1 Μαρκοβιανές Διεργασίες Απόφασης (Markov Decision Processes)

Για να ελέγξουμε μια διαδικασία πρέπει να μπορούμε να παρατηρήσουμε κάθε στιγμή την κατάσταση στην οποία βρίσκεται, ποιες ενέργειες θα εκτελεστούν που θα την επηρεάσουν και τι αποτέλεσμα θα έχουν οι ενέργειες αυτές. Για παράδειγμα για να βρούμε την επόμενη θέση και ταχύτητα που θα έχει ένα ρομπότ, πρέπει να γνωρίζουμε την προηγούμενη θέση του, πόσο έχουν αλλάξει οι τιμές των γωνιών των αρθρώσεων του, τι τριβές αναπτύσσονται στο δάπεδο κλπ.

Η Υπόθεση Μάρκοβ (Markov Assumption) δηλώνει πως για να βρούμε την επόμενη κατάσταση του ρομπότ δεν χρειάζεται να γνωρίζουμε τη θέση του και την ταχύτητά του όταν το εκκινήσαμε πριν κάποια λεπτά, αλλά την κατάσταση του ρομπότ την τρέχουσα χρονική στιγμή.

Ορισμός 3.1. *Υπόθεση Μάρκοβ (Markov Assumption): Η δυναμική ορισμένων συστημάτων μπορεί να περιγραφεί μόνο από την τρέχουσα κατάσταση του συστήματος και τη δράση που εφαρμόζεται σε αυτό και όχι από το πλήρες ιστορικό των καταστάσεων και των ενεργειών του συστήματος. Διαφορετικά, αν x_t είναι η τρέχουσα κατάσταση του συστήματος και u_t η τρέχουσα ενέργεια που εφαρμόζεται, τότε:* $Pr\{x_{t+1} = x' | x_t, u_t, x_{t-1}, u_{t-1}, \ldots, x_0, u_0\} = Pr\{x_{t+1} = x' | x_t, u_t\}$

Η Ενισχυτική Μάθηση χρησιμοποιείται για την επίλυση προβλημάτων, τα οποία μοντελοποιούνται με τη βοήθεια των Μαρκοβιανών Διεργασιών Απόφασης $(MarkovDecisionProcesses)$.

Ορισμός 3.2. *Μια ΜΔΑ είναι μια στοχαστική διαδικασία διακριτού χρόνου, η οποία περιγράφεται από το διάνυσμα (X, U, T, R, γ, D), όπου:*

- *X είναι ο χώρος καταστάσεων του προβλήματος, ο οποίος αντιπροσωπεύει τις διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί η διαδικασία κάθε χρονική στιγμή.*
- *U είναι ο χώρος των ενεργειών ή αποφάσεων της διεργασίας και αντιπροσωπεύει το σύνολο των διαφορετικών ενεργειών που μπορεί να εκτελέσει ο αποφασίζων κάθε χρονική στιγμή.*
- *T είναι το Μοντέλο Μετάβασης (Transition Model) της διαδικασίας και περιγράφει την πιθανότητα αν η διαδικασία βρίσκεται στην κατάσταση x τη χρονική στιγμή t και επιλεγεί η ενέργεια*

u, να βρεθεί η διαδικασία σε μια νέα κατάσταση x' ή διαφορετικά $T(x, u, x') = T(x'|x, u)$. Γίνεται κατανοητό πως το μοντέλο μετάβασης ακολουθεί την υπόθεση Μάρκοβ.

- *R είναι η Συνάρτηση Ανταμοιβής (Reward Function) ή Συνάρτηση Κόστους (Cost Function) της διαδικασίας, η οποία καθορίζει την ανταμοιβή που συλλέγει το σύστημα κάθε χρονική στιγμή. Η συνάρτηση ανταμοιβής είναι μια απεικόνιση των μεταβάσεων του συστήματος σε πραγματικούς αριθμούς, $R : X \times U \times X \to \mathbb{R}$. Και η συνάρτηση ανταμοιβής ακολουθεί την υπόθεση Μάρκοβ, δηλαδή η ανταμοιβή που λαμβάνει κάθε τιγμή το σύστημα ορίζεται ως $r_t = R(x, u, x')$ και δεν εξαρτάται από το ιστορικό του συστήματος, αλλά είναι πολύ συνηθισμένες και ανταμοιβές της μορφής $r_t = R(x, u)$ ή και $r_t = R(x)$.*

- *$\gamma \in (0, 1]$ είναι ο παράγοντας έκπτωσης (discount factor) της διαδικασίας, ο οποίος καθορίζει πόση επίδραση θα έχουν στη διαδικασία ανταμοιβές που συλλέγονται μετά από t βήματα της διαδικασίας.*

- *D είναι μια πιθανοτική κατανομή πάνω στο χώρο των καταστάσεων X, η οποία καθορίζει την αρχική κατάσταση της διαδικασίας.*

3.2 Το βασικό μοντέλο της Ενισχυτικής Μάθησης

Στην Ενισχυτική Μάθηση θεωρούμε πως το πρόβλημα μπορεί να διαχωριστεί στο σύστημα (ή περιβάλλον) και στον αποφασίζοντα (decision maker). Θεωρούμε πως σε κάθε χρονική στιγμή το σύστημα βρίσκεται σε μια κατάσταση $x \in X$ και ο αποφασίζων εφαρμόζει πάνω του μια ενέργεια $u \in U$, η οποία έχει σαν συνέπεια τη μεταφορά του συστήματος σε μια επόμενη κατάσταση $x' \in X$, και την συλλογή μιας ανταμοιβής $r \in \mathbb{R}$. Στην παρούσα εργασία, όπου ασχολούμαστε με πραγματικά ρομποτικά συστήματα οι καταστάσεις και οι ενέργειες θα ανήκουν σε συνεχείς χώρους, δηλαδή $X \subseteq \mathbb{R}^N$ και $U \subseteq \mathbb{R}^M$.[1]

Για να μπορέσουμε να ελέγξουμε το σύστημα πρέπει σε κάθε χρονική στιγμή να επιλέγουμε την κατάλληλη ενέργεια για την εργασία που θέλουμε να εκτελέσουμε. Η συνάρτηση η οποία επιλέγει κάθε στιγμή μια ενέργεια, ανάλογα με την κατάσταση του συστήματος ονομάζεται πολιτική.

Ορισμός 3.3. *Πολιτική (Policy) ή ελεγκτής (controller) ονομάζεται η αντιστοίχιση $\pi(s) : S \to A$ από καταστάσεις σε ενέργειες. Για την παρούσα εργασία θα θεωρούμε ότι η πολιτική είναι της μορφής $u_t \sim \pi_\theta(u_t|x_t) = p(u_t|x_t, \theta)$, όπου το διάνυσμα $\theta \in \mathbb{R}^L$ δηλώνει τις L παραμέτρους της πολιτικής π_θ. Η πολιτική εδώ είναι στοχαστική, χρησιμεύοντας στην εξερεύνηση νέων καταστάσεων του συστήματος.*

Συνοψίζοντας όλα τα προηγούμενα έχουμε:

Ορισμός 3.4. *Ένα Μαρκοβιανό Σύστημα ορίζεται από τις εξισώσεις:*

[1] Στην παρούσα εργασία θεωρούμε $M = 1$

$$x_0 \sim p(x_0), \tag{3.1}$$
$$x_{t+1} \sim p(x_{t+1}|x_t, u_t), \tag{3.2}$$
$$r_{t+1} = R(x_t, u_t, x_{t+1}), \tag{3.3}$$
$$u_t \sim \pi_\theta(u_t|x_t) = p(u_t|x_t, \theta), \tag{3.4}$$

με καταστάσεις $x_t \in X \subseteq \mathbb{R}^N$, ενέργειες $u_t \in U \subseteq \mathbb{R}^M$ για όλες τις χρονικές στιγμές $t \in \mathbb{N}$ και παραμέτρους $\theta \in \mathbb{R}^L$.

Εκτός από τις παραπάνω εξισώσεις πρέπει να γνωρίζουμε για πόσο χρονικό διάστημα θα λειτουργεί το σύστημά μας, πρέπει να γνωρίζουμε δηλαδή αν η διεργασία είναι άπειρη ή διακόπτεται μετά από κάποια βήματα. Ο ορίζοντας μιας διεργασίας είναι ο "χρόνος ζωής" της διεργασίας. Ο ορίζοντας μπορεί να είναι άπειρος (η διεργασία εκτελείται για πάντα) ή πεπερασμένος (η διεργασία τερματίζει με πιθανότητα 1 μετά από κάποιο αριθμό βημάτων, ο οποίος δεν είναι γνωστός εκ των προτέρων) ή δεδομένος (η διεργασία εκτελείται για συγκεκριμένο αριθμό βημάτων).

Μια πλήρης εκτέλεση των εξισώσεων 3.1 - 3.4 έως τον ορίζοντα H ονομάζεται επεισόδιο ή τροχιά (rollout ή trajectory ή episode) και αποτελείται από μια ακολουθία από καταστάσεις, ενέργειες και ανταμοιβές:

$$x_0 \xrightarrow[r_0]{u_0} x_1 \xrightarrow[r_1]{u_1} x_2 \xrightarrow[r_2]{u_2} x_3 \xrightarrow[r_3]{u_3} x_4 \xrightarrow[r_4]{u_4} \quad \dots \quad x_{H-1} \xrightarrow[r_{H-1}]{u_{H-1}} x_H \tag{3.5}$$

Ας υποθέσουμε τώρα πως έχουμε ένα σύστημα, το οποίο αλληλεπιδρά με το περιβάλλον και δημιουργεί επεισόδια και πως έχουμε επιλέξει μια παραμετροποιημένη πολιτική, η οποία παράγει τις ενέργειες του συστήματος. Σκοπός μας είναι να μεγιστοποιήσουμε τις ανταμοιβές που συλλέγει το σύστημα για όλες τις πιθανές τροχιές που μπορεί να παραχθούν. Το μέτρο που χρησιμοποιείται είναι η αναμενόμενη ανταμοιβή (expected total discounted reward ή expected cumulative reward ή value of policy ή expected return). Πλέον μπορούμε να ορίσουμε το πλαίσιο της Ενισχυτικής Μάθησης:

Ορισμός 3.5. *Σκοπός της Ενισχυτικής Μάθησης είναι να βρει την πολιτική π_θ^* με παραμέτρους $\boldsymbol{\theta}^* \in \mathbb{R}^L$, η οποία είναι βέλτιστη ως προς μια αντικειμενική συνάρτηση (evaluation function) $J(\theta)$, χρησιμοποιώντας μόνο δειγματική εμπειρία από το σύστημα. Η πιο συνηθισμένη αντικειμενική συνάρτηση είναι η αναμενόμενη ανταμοιβή, δηλαδή θα έχουμε:*

$$J(\theta) = \mathbb{E}\left\{\sum_{t=0}^{H} \gamma^t r_t; \theta\right\}, \tag{3.6}$$

όπου η αναμενόμενη τιμή $\mathbb{E}\{\cdot\}$ αφορά όλες τις πιθανές τροχιές που μπορούν να προκύψουν ξεκινώντας από την κατάσταση x_0 και ακολουθώντας την π_θ.

3.3 Ενισχυτική Μάθηση με Συναρτήσεις Αξιολόγησης

Πολλοί αλγόριθμοι Ενισχυτικής Μάθησης στηρίζονται στην εκτίμηση κάποιων συναρτήσεων αξιολόγησης. Για μια Μαρκοβιανή Διεργασία Απόφασης η συνάρτηση αξιολόγησης κατάστασης (state

value function) V αποδίδει μια τιμή σε μια κατάσταση x του συστήματος. Η τιμή $V^{\pi}(x)$ μιας κατάστασης x με πολιτική π είναι η αναμενόμενη ανταμοιβή που συλλέγει το σύστημα όταν ξεκινά από την κατάσταση x και ακολουθεί πολιτική π:

$$V^{\pi}(x) = \mathbb{E}_{\pi}\left\{\sum_{t=0}^{H}\gamma^{t}r_{t}|x_{0}=x\right\} \tag{3.7}$$

Παρόμοια, η συνάρτηση αξιολόγησης κατάστασης - ενέργειας (state - action value function) αποδίδει μια τιμή σε κάθε ζεύγος (x, u) καταστάσεων και ενεργειών. Η τιμή $Q^{\pi}(x, u)$ της επιλογής να εκτελεστεί η ενέργεια u όταν το σύστημα βρίσκεται σε κατάσταση x και ακολουθεί πολιτική π είναι η αναμενόμενη ανταμοιβή που συλλέγεται όταν το σύστημα βρίσκεται σε κατάσταση x, εκτελείται η ενέργεια u στο πρώτο βήμα και ακολουθείται στη συνέχεια η πολιτική π:

$$Q^{\pi}(x,u) = \mathbb{E}_{\pi}\left\{\sum_{t=0}^{H}\gamma^{t}r_{t}|x_{0}=x,u_{0}=u\right\} \tag{3.8}$$

Η συνάρτηση αξιολόγησης κατάστασης και η συνάρτηση αξιολόγησης κατάστασης - ενέργειας συνδέονται με την ακόλουθη σχέση:

$$Q^{\pi}(x,u) = r(x,u) + \gamma\sum_{x'\in X}T(x,u,x')V^{\pi}(x'), \tag{3.9}$$

Δεδομένης μιας πολιτικής π, η άπληστη πολιτική (Βέλτιστη ως προς τις άλλες)(greedy policy) π' πάνω στην π (over π) είναι μια αιτιοκρατική πολιτική η οποία μπορεί να υπολογιστεί απ' ευθείας από την Q^{π} ή τη V^{π}. Ειδικότερα, είναι η πολιτική η οποία μεγιστοποιεί την Q^{π} σε κάθε κατάσταση:

$$\pi'(x) = \arg\max_{u\in U}Q^{\pi}(x,u) \tag{3.10}$$

και ισχύει

$$\forall x\in X,\quad Q^{\pi'}(x,\pi'(x)) \geq Q^{\pi}(x,\pi(x)) \tag{3.11}$$

Για κάθε πολιτική π η γραμμική εξίσωση Bellman συνδέει τις τιμές Q^{π} μεταξύ ζευγών καταστάσεων και ενεργειών. Για αιτιοκρατικές πολιτικές ισχύει:

$$Q^{\pi}(x,u) = r(x,u) + \gamma\sum_{x'\in X}T(x,u,x')Q^{\pi}(x',\pi(x')) \tag{3.12}$$

Οι τιμές για τις Q^{π} για όλα τα ζεύγη καταστάσεων - ενεργειών μπορούν να βρεθούν αν λυθεί το γραμμικό σύστημα διαστάσεων $|X||U| \times |X||U|$, που προκύπτει από τις εξισώσεις Bellman για όλα τα ζεύγη καταστάσεων - ενεργειών.

Αντίθετα, η εξίσωση βελτιστότητας Bellman (Bellman optimality equation), η οποία συνδέει τις τιμές της βέλτιστης συνάρτησης Q^{*} της βέλτιστης πολιτικής π^{*} είναι μη γραμμική και καταλήγει σε ένα μη γραμμικό σύστημα εξισώσεων:

$$Q^{*}(x,u) = r(x,u) + \gamma\sum_{x'\in X}T(x,u,x')\max_{u'\in U}Q^{*}(x',u') \tag{3.13}$$

Η εξίσωση δείχνει πως οι επιλογές των ενεργειών της βέλτιστης πολιτικής μεγιστοποιούν την αναμενόμενη ανταμοιβή.

Η επίλυση μιας ΜΔΑ είναι η εύρεση της βέλτιστης πολιτικής π^* για την ΜΔΑ. Οι κλασικές μέθοδοι για την εύρεση της π^* είναι:

- Value Iteration: Ξεκινά με ένα αρχικό διάνυσμα Q και εφαρμόζοντας συνεχώς την εξίσωση βελτιστότητας Bellman προσεγγίζει την τιμή της βέλτιστης Q^*. Στην συνέχεια υπολογίζει την π^*, η οποία είναι η άπληστη πολιτική πάνω στην Q^*.

- Policy Iteration: Ξεκινά με μια αρχική πολιτική π, υπολογίζει τις τιμές τις Q^π επιλύοντας τις γραμμικές εξισώσεις Bellman και εξάγει την νέα πολιτική π', η οποία είναι η άπληστη πολιτική πάνω στην Q^π. Τα δύο αυτά βήματα επαναλαμβάνονται μέχρι $\pi' = \pi^*$.

Οι κλασικές αυτές μέθοδοι, οι οποίες στηρίζονται στις εξισώσεις Bellman απαιτούν γνώση του συστήματος, αφού χρησιμοποιούν το μοντέλο μετάβασης T. Για να ξεπεραστεί το εμπόδιο αυτό έχουν αναπτυχθεί πολλοί αλγόριθμοι (Q-Learning, Fitted-Q, LSPI κ.ά), οι οποίοι επιχειρούν να μάθουν τις τιμές Q από δείγματα που συλλέγονται κατά την εκτέλεση της ΜΔΑ.

3.4 Ο αλγόριθμος Monte Carlo Expectation Maximization για Ενισχυτική Μάθηση

Η κεντρική ιδέα του αλγορίθμου είναι να θεωρήσουμε τις ανταμοιβές σαν πιθανότητες κάποιων φανταστικών γεγονότων (για το λόγο αυτό απαιτείται $r \in [0, 1]$). Συγκεκριμένα, η ανταμοιβή r_T που συλλέχθηκε σε κάποιο βήμα T της άπειρου ορίζοντα ΜΔΑ, είναι η πιθανότητα κάποιο φανταστικό γεγονός R να συμβεί στο τελευταίο βήμα μιας T-ορίζοντα ΜΔΑ, η οποία έχει το ίδιο δυναμικό μοντέλο με την άπειρου ορίζοντα ΜΔΑ.

Είδαμε πως σκοπός μας είναι να βρούμε τις βέλτιστες παραμέτρους, οι οποίες μεγιστοποιούν τη συνάρτηση ανταμοιβής:

$$J(\theta) = \mathbb{E}\left\{\sum_{t=0}^{H} \gamma^t r_t; \theta\right\}, \tag{3.14}$$

Διαφορετικά, αν $p_{\boldsymbol{\theta}}(\xi)$ δηλώνει την πιθανοφάνεια μιας πλήρους τροχιάς $\xi \in \Xi$, θα ισχύει:

$$J(\theta) = \mathbb{E}_\xi\left\{r(\xi)\right\} = \sum_{\Xi} p_\theta(\xi) r(\xi) \tag{3.15}$$

Στον αλγόριθμο MCEM χρησιμοποιούμε τις εξής παραμέτρους:

$$\alpha(t) = (1-\delta)\delta^t, \tag{3.16}$$

$$b(t) = (1-\gamma/\delta)(\gamma/\delta)^t, \tag{3.17}$$

που είναι γεωμετρικές κατανομές με $\gamma < \delta < 1$.

Χρησιμοποιούμε τον ελεγκτή:

$$u_t = (\theta + \varepsilon_t)\phi(x_t), \tag{3.18}$$

όπου $\boldsymbol{\phi} : \mathbb{R}^n \mapsto \mathbb{R}^d$ είναι σταθερές συναρτήσεις βάσης και ε_t είναι λευκός Gaussian θόρυβος $\varepsilon_t \sim \mathcal{N}(\varepsilon_t; 0, \sigma^2 I_d)$, ο οποίος χρησιμοποιείται για την εξερεύνηση του χώρου των παραμέτρων. Το σ μπορεί να είναι σταθερό, να μεταβάλλεται σε κάθε επανάληψη του αλγορίθμου ή να είναι κι αυτό μια από τις παραμέτρους προς βελτιστοποίηση, ετσι προκύπτει η παρακάτω σχέση:

$$\theta_{k+1} = \theta_k + \frac{\sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} Q_{it}\boldsymbol{\varepsilon}_{it}}{\sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} Q_{it}} \tag{3.19}$$

όπου

$$Q_{it} = \sum_{\tau=t}^{|\xi_i|} b(\tau) r_{i\tau} \tag{3.20}$$

Συνοπτικά ο αλγόριθμος MCEM φαίνεται στον πίνακα 3.1.

Πίνακας 3.1: Μάθηση πολιτικής με τον αλγόριθμο MCEM

Είσοδος: αρχικές παράμετροι πολιτικής $\boldsymbol{\theta}$

Επανάλαβε
  Επανάλαβε για $i = 1 : m$
    Επίλεξε ένα τυχαίο μήκος τροχιάς T_i από κατάλληλη γεωμετρική κατανομή
    Συνέλεξε δειγματική τροχιά με $\boldsymbol{u}_t = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^T \boldsymbol{\phi}(\boldsymbol{x}_t)$, όπου $[\boldsymbol{\varepsilon}_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$
    και αποθήκευσε όλα τα $(t, \boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}, \boldsymbol{\varepsilon}_t, r_{t+1})$ για $t = 1, 2, \ldots, T_i$
  Έως $i = m$
  Υπολόγισε τα $Q_{it} = \sum_{t=\tau}^{T_i} b(t) r_{it}$
  Βρες νέες παραμέτρους $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + (\sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} Q_{it}\boldsymbol{\varepsilon}_{it})/(\sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} Q_{it})$
Έως $\boldsymbol{\theta}_{k+1} \simeq \boldsymbol{\theta}_k$

Έξοδος: θ_k

3.5 Ο αλγόριθμος Policy learning by Weighting Exploration with the Returns (PoWER)

Αν στον αλγόριθμο MCEM θέσουμε $\alpha(T) = 1$ για $T = H$ και 0 για οποιοδήποτε άλλο μήκος τροχιάς και επιπλέον θέσουμε $b(t) = 1/(1 + H)$ για $t = 1, 2, \ldots, H$, τότε η εξίωση 3.19 παραμένει αμετάβλητη, με τη μόνη απλοποίηση ότι θα έχουμε το ίδιο μήκος H για όλα τα επεισόδια και ο όρος $b(t)$ θα είναι μια σταθερά, οπότε $Q_{it} = \sum_{t=\tau}^{H} r_{it}$, και ο νέος αλγόριθμος που προκύπτει είναι ο PoWER.

Ο αλγόριθμος PoWER είναι σχεδιασμένος για προβλήματα πεπερασμένου ορίζοντα, ενώ ο αλγόριθμος MCEM είναι δομημένος έτσι ώστε να ανταποκρίνεται σε προβλήματα άπειρου ορίζοντα στα

οποία η ανταμοιβή του συστήματος μειώνεται με το χρόνο. Αυτό δεν σημαίνει πως ο PoWER δεν μπορεί να επιλύσει προβλήματα άπειρου ορίζοντα, απλά απαιτεί την εύρεση κατάλληλου μήκους τροχιάς H. Το πρόβλημα είναι πως δεν υπάρχει κάποιος κανόνας για το βέλτιστο μήκος H , το οποίο μπορεί να διαφέρει αρκετά από πρόβλημα σε πρόβλημα, αλλά και για διαφορετικές αρχικές τιμές του ίδιου προβλήματος.

Η δεύτερη διαφορά ανάμεσα στον MCEM και στον PoWER είναι οι ποσότητες $b(t)$ οι οποίες μειώνονται με το χρόνο, δίνοντας μεγαλύτερο βάρος στις ανταμοιβές που λαμβάνονται στα πρώτα βήματα της τροχιάς. Το γεγονός αυτό μπορεί να βοηθήσει σημαντικά στη σύγκλιση του MCEM όταν η δυναμική της ΜΔΑ έχει αρκετό θόρυβο, καθώς στην περίπτωση αυτή οι ανταμοιβές που λαμβάνει το σύστημα σε βάθος χρόνου έχουν εκφυλιστεί από το θόρυβο του περιβάλλοντος.

Συνοπτικά ο αλγόριθμος PoWER φαίνεται στον πίνακα 3.2.

Πίνακας 3.2: Μάθηση πολιτικής με τον αλγόριθμο PoWER

Είσοδος: αρχικές παράμετροι πολιτικής $\boldsymbol{\theta}$

Επανάλαβε

Συνέλεξε m δειγματικές τροχιές με $\boldsymbol{u}_t = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^T \boldsymbol{\phi}(\boldsymbol{x}_t)$, όπου $[\boldsymbol{\varepsilon}_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$

και αποθήκευσε όλα τα $(t, \boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}, \boldsymbol{\varepsilon}_t, r_{t+1})$ για $t = 1, 2, \ldots, H$

Υπολόγισε τα $\hat{Q}^{\pi}_{it}(\boldsymbol{x}, \boldsymbol{u}, \tau) = \sum_{t=\tau}^{T_i} r_{it}$

Βρες νέες παραμέτρους $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + (\sum_{i=1}^{m} \sum_{t=0}^{H} Q_{it}(\boldsymbol{x}, \boldsymbol{u}, t)\boldsymbol{\varepsilon}_{it})/(\sum_{i=1}^{m} \sum_{t=0}^{H} Q_{it}(\boldsymbol{x}, \boldsymbol{u}, t))$

Έως $\boldsymbol{\theta}_{k+1} \simeq \boldsymbol{\theta}_k$

Έξοδος: θ_k

Κεφάλαιο 4

ΕΦΑΡΜΟΓΗ ΣΕ ΡΟΜΠΟΤ NAO

4.1 Εισαγωγή

Αυτό που μας ενδιαφέρει είναι η εύρεση και η βελτίωση των τροχιών που θα διαγράψουν κάποιες απο τις αρθρώσεις του NAO κατά την εκτέλεση μιας κίνησης σε ένα ορισμένο χρονικό διάστημα. Έτσι λοιπόν κατασκευάζουμε τις τροχιές αυτές, τις οποίες θα ονομάζουμε $q(i)$ όπου $i = 1, 2, 3, ..., 22$ το πλήθος των αρθρώσεων. Η προσέγγιση των τροχιών αυτών μπορεί να γίνει με πολλούς τρόπους, είτε με $Splines$, είτε με $CentralPatternGenerators(CPGs)$. Σε μια πρώτη προσέγγιση χρησιμοποιήσαμε για την παραμετροποίηση βασικές τριγωνομετρικές συναρτήσεις της μορφής όπως αυτής της εξίσωσης (5.2). Επειδή το ανθρώπινο περπάτημα είναι μια περιοδική διαδικασία μας βολεύει να χρησιμοποιήσουμε τον μετασχηματισμό Fourier για την παραμετροποίηση των τιμών των γωνιών που παίρνουμε από το $MotionFile$.

Το K είναι ένας σταθερός αριθμός $(K = 9)$. Με δοκιμή και σφάλμα διαπιστώθηκε ότι μια τιμή του K μεταξύ 6 και του 10 μπορεί να προσεγγίζει τις περισσότερες τροχιές ικανοποιητικά. Οι αρχικές τιμές των παραμέτρων a_i και b_i μπορούν να υπολογιστούν με αντίστροφο μετασχηματισμό $Fourier$.

4.2 Ο Μετασχηματισμός Fourier

Η ανάλυση Fourier αποτελεί θέμα κεντρικής σημασίας στα Εφαρμοσμένα Μαθηματικά. Παραδείγματος Χάριν, είναι απαραίτητη για την μελέτη *Περιοδικών* η *Σχεδόν Περιοδικών* φαινομένων ως προς το χρόνο ή τον χώρο, όπως είναι διαφόρων ειδών κυματισμοί (κύματα επιφάνειας στο νερό, παλίρροιες, ακουστικά, ηλεκτρομαγνητικά και ελαστικά κύματα κ.α.), οι ταλαντώσεις και οι περιοδικές κινήσεις των σωμάτων, το εναλλασσόμενο ηλεκτρικό ρεύμα κλπ.

Ειδικότερα, η αριθμητική ανάλυση Fourier ασχολείται με την προσέγγιση συνεχών *περιοδικών* φαινομένων από διακριτά και με την ανάλυση και διαχείριση περιοδικών ή σχεδόν περιοδικών συναρτήσεων, των οποίων οι τιμές είναι γνωστές μόνο σ'ένα πεπερασμένο σύνολο σημείων *«δειγματοληψίας»* και οι σχετικοί υπολογισμοί γίνονται με οικονομία πράξεων, αν χρησιμοποιούμε αλγορίθμους Ταχέων Μετασχηματισμών Fourier (TMF).

λέμε ότι μια συνάρτηση $f : \mathbb{R} \to \mathbb{C}$ είναι περιοδική με περίοδο T (ή T-περιοδική), αν ισχύσει:

$$f(t + T) = f(t), \quad t \in R \tag{4.1}$$

Για απλούστευση του συμβολισμού θα υποθέσουμε ότι οι περιοδικές συναρτήσεις μας έχουν περίοδο $T = 1$. (Είναι προφανές ότι η f είναι T-περιοδική, αν και μόνο αν η συνάρτηση $g, g(t) = f(Tt)$, $\boldsymbol{t} \in R$, είναι 1-περιοδική. Έτσι αν $p(t)$ είναι προσέγγιση της τιμής $g(t)$, τότε η $p(t/T)$ είναι προσέγγιση της $f(t)$).

Μια 1-περιοδικη συνάρτηση f μπορεί να παρασταθεί, υπό ορισμένες συνθήκες, με την αντίστοιχη *(τριγωνομετρική)* σειρά $Fourier$

$$f(t) \sim \frac{1}{2}a_0 + \sum_{k=0}^{\infty}(a_k cos2\pi kt + b_k sin2\pi kt) \tag{4.2}$$

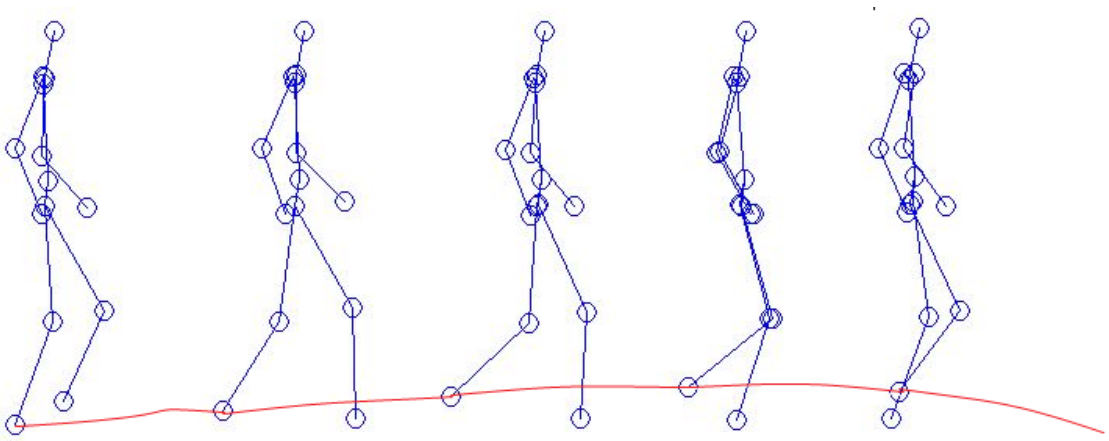
Όπου a_κ και b_κ λέγονται συντελεστές Fourier της f και δίνονται από τους τύπους:

$$\begin{cases} \alpha_k = 2\int_0^1 f(\tau)cos2\pi\tau d\tau, & \kappa\text{=0,1,2,...,} \\ b_k = 2\int_0^1 f(\tau)sin2\pi\tau d\tau, & \kappa\text{=0,1,2,...,} \end{cases} \tag{4.3}$$

a_κ και b_κ μπορούν να υπολογιστούν με αντίστροφο μετασχηματισμό $Fourier$.

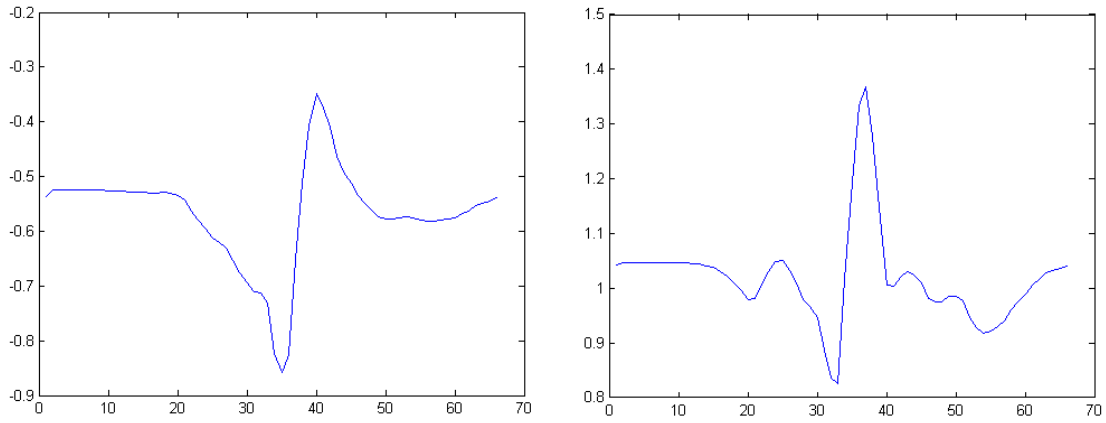
4.3 Παραμετροποίηση του συστήματος με χρήση Fourier

Από την φύση γνωρίζουμε ότι κάθε σημείο που κινείται στο χώρο τότε αυτό το σημείο διαγράφει μια τυχαία τροχιά η οποία δείχνει τα σημεία που έχει περάσει από πάνω τους αυτό το σώμα, σχήμα(4.1). Κάθε γωνία διαγράφει μια τροχιά. Κάθε τροχιά ακολουθεί ένα επαναλαμβανόμενο μοτίβο. Σκοπός είναι η δημιουργία μιας συνάρτησης $f(t)$, με χρήση $Fourier$, για κάθε γωνία από τις 25 η οποία θα προσεγγίζει όσο το δυνατό καλύτερα την φυσική κίνηση(στην περίπτωσή μας το περπάτημα).



Σχήμα 4.1: Η τροχιά που διαγράφει μια άρθρωση

Με την βοήθεια του προγράμματος Matlab εφαρμόζεται ένα $FFT(FastFourierTransform)$ στις τιμές που διαβάζεται από ένα $Motionfile$ για κάθε μια γωνία ξεχωριστά. Για να είναι εφικτός ο FFT πραγματοποιούνται οι κατάλληλες αλλαγές στους χρόνους και στις σταθερές. Στο σχήμα(4.2) παρουσιάζονται οι τροχιές δυο αρθρώσεων του αριστερού ποδιού. Το σχήμα(4.3) μας δείχνει τς τροχιές που διαγράφει η κάθε μια άρθρωση σε ένα ορισμένο χρονικό διάστημα .



Σχήμα 4.2: Η τροχιά που διαγράφει το αριστερό πόδι

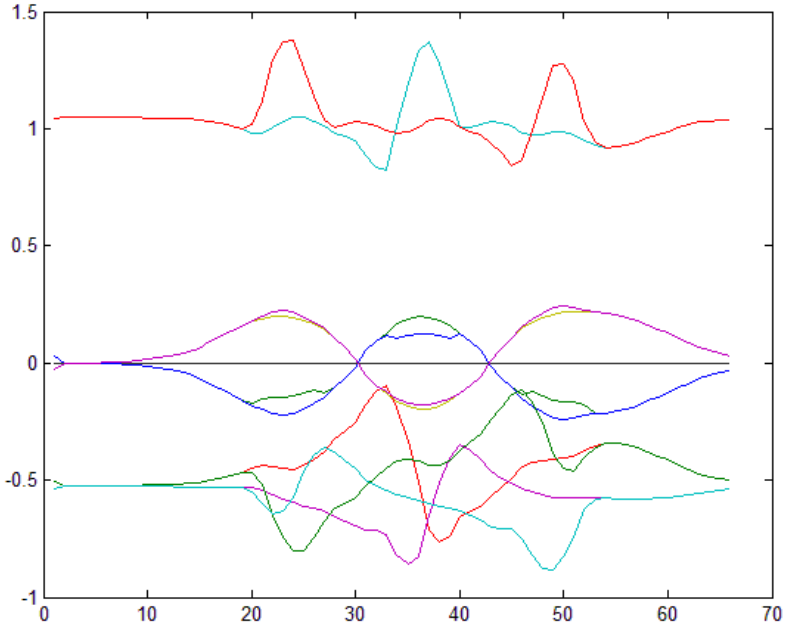
Ο μετασχηματισμό $Fourier$ προσεγγίζει με τον καλύτερο δυνατό τρόπο την τροχιά που εκτέλεσε η κάθε άρθρωση στο χώρο. Για παράδειγμα για τις προηγούμενες αρθρώσεις η τροχιά που ακολουθεί η άρθρωση είναι με το μπλε χρώμα ενώ η προσέγγιση τους είναι με το κόκκινο χρώμα όπως παρατηρούμε στο σχήμα(4.4).

4.4 Reinforcement Learning

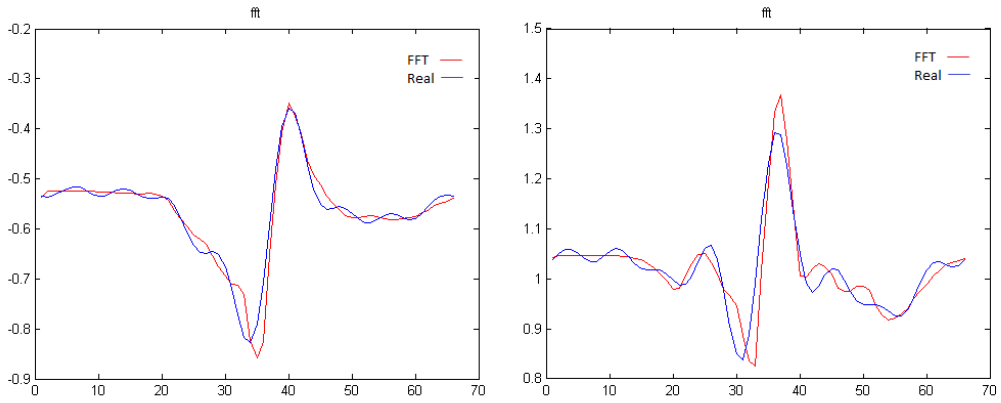
Ο αλγόριθμος υλοποίησης της μεθόδου έχει συνολικά 3 μέρη. Στην αρχή γίνεται η παραμετροποίηση για κάθε μία από τις 22 αρθρώσεις του NAO. Στο δεύτερο μέρος υπολογίζονται οι τροχιές των αρθρώσεων που θα βελτιώσουμε χρησιμοποιώντας κάποιες τυχαίες αρχικές τιμές στις παραμέτρους. Στην συνέχεια εκτελείται η κίνηση. Καθώς οι αρθρώσεις λαμβάνουν τις αντίστοιχες τιμές για την κάθε χρονική στιγμή, επίσης για κάθε χρονική στιγμή υπολογίζεται η ανταμοιβή $r(Rewards)$ της θέσης του ρομπότ. Στο τρίτο μέρος του αλγορίθμου, υπολογίζουμε τις νέες παραμέτρους (α), (b) και με τις νέες αυτές παραμέτρους ξεκινά το νέο επεισόδιο. Τα επεισόδια είναι τόσα όσα χρειάζεται για να έχουμε ικανοποιητικά αποτελέσματα.

4.5 Η συνάρτηση Rewards

Η παραμετροποίηση της συνάρτησης ανταμοιβής λαμβάνει κατά 80 τοις εκατό την απόσταση που περπατάει το ρομπότ προς τα μπροστά και κατά 20 τοις εκατό την μετακίνηση που κάνει προς τα αριστερά και τα δεξιά. Η συνάρτηση ανταμοιβής πρέπει να παίρνει τιμές μεταξύ του 0 και του 1 (λόγο του αλγορίθμου Power που χρησιμοποιούμε). Η παρακάτω εξίσωση θα μετρήσει την διάφορα απόστασης μεταξύ του σημείου στο οποίο έχει φτάσει το ρομπότ και του τελικού σημείου εκεί που θέλουμε να επιτύχουμε, έπειτα το πολλαπλασιάζουμε με (-1) και το υψώνουμε στον έκθετη της εκθετκής συνάρτησης $e^{-1(\dots)}$, άρα το αποτέλεσμα που θα έχουμε θα είνα ένα νούμερο μεταξύ 0 και



Σχήμα 4.3: Η τροχιά που διαγράφει το αριστερό πόδι



Σχήμα 4.4: Ο μετασχηματισμός Fourier της τροχιάς που διαγράφει το αριστερό πόδι

1. Έτσι ικανοποιούμε την περιορισμό που έχουμε στο Power και δεν ξεφεύγουμε εκτός των ορίων μας. Όμως πολλές φόρες θα έχουμε νούμερα πολύ μικρά τα οποία η γλώσσα προγραμματισμού και η $Matlab$ δεν μπορεί να τα υπολογίσει, και για να βρούμε την τιμή του πρέπει να μετατραπεί σε μεγάλο νούμερο έτσι ώστε να μπορεί η γλώσσα προγραμματισμού να το υπολογίσει. Η λύση είναι να διαιρέσουμε τον εκθέτη δια ένα μεγάλο σταθερό νούμερο και με αυτό τον τρόπο θα επιτύχουμε ένα πολύ καλό αποτέλεσμα.

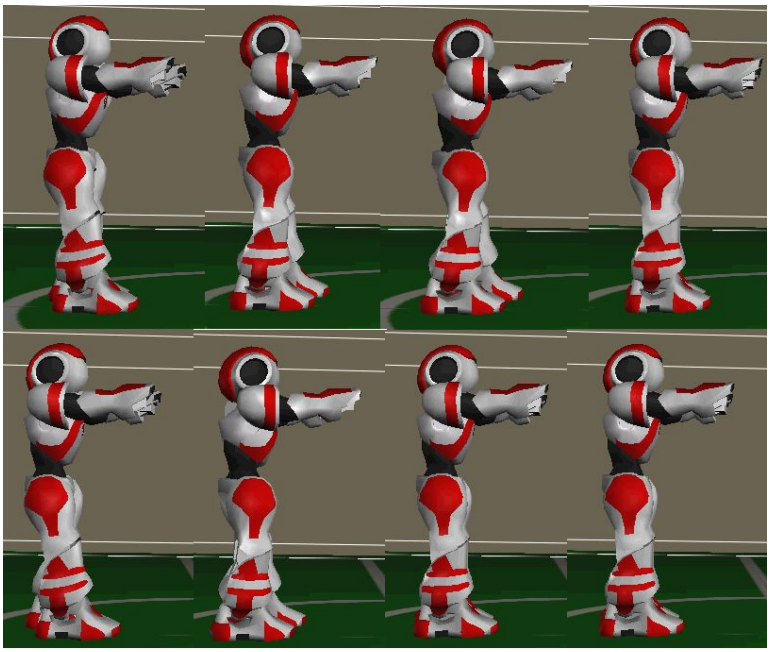
Κεφάλαιο 5

Αποτελέσματα

Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα των πειραμάτων που υλοποιήθηκαν στα ρομπότ NAO.

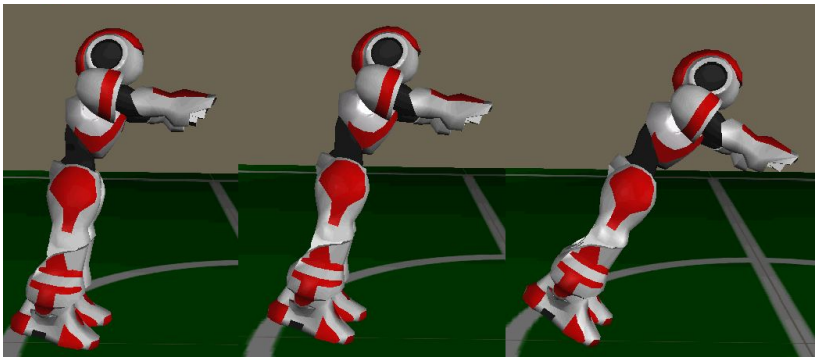
5.1 Εφαρμογή πάνω στο NAO

Οι κινήσεις που επιτύχαμε παρατίθενται στις παρακάτω εικόνες σχήμα(5.1). Η παραμετροποίηση των γωνιών έγινε με την συνάρτηση της μορφής που προκύπτει από την εξίσωση(4.2).



Σχήμα 5.1:

Αρκετές αποτυχίες είχαμε στην φάση που εφαρμόσαμε το περπάτημα χωρίς την χρήση του μετασχηματισμού $Fourier$. Όπως φαίνεται στο σχήμα(5.2).



Σχήμα 5.2:

Οι γωνίες που παραμετροποιήθηκαν είναι 3 γωνίες από κάθε πόδι (Αστράγαλος, Γοφς, Γόνατος). Χρησιμοποιούμε συγκεκριμένα τις γωνίες $Pitch$ και όχι των Yaw και $Roll$ για να επιτύχουμε σίγουρη μετατόπιση προς τα μπροστά. $Pitch$: Είναι η γωνία που θα μας δείχνει πόσο γέρνει το ρομπότ σε σχέση με το δάπεδο κίνησης.

Ο χρόνος που είχαμε ήταν μη πραγματικός, στο πρόγραμμα Webots επιταχύνουμε την κίνηση του ρομπότ για να επιτύχουμε καλά αποτελέσματα σε λιγότερο χρόνο.

Κάθε ένα λεπτό πραγματικό το Webots προσομοιώνει 7,5 λεπτά, δηλαδή μια πραγματική ώρα αντιστοιχεί σε 450 λεπτά δηλαδή σε 7.5 ώρες! Επίσης σε αυτούς τους αλγορίθμους εφαρμόστηκε αναζήτηση επί γραμμής και ο αλγόριθμος της φυσαλίδας (κυρίως στο MCEM) και ο λόγος ήταν για να συλλέγονται κάθε φορά οι πιο σωστές τροχιές και να πραγματοποιείται μάθηση με αυτές.

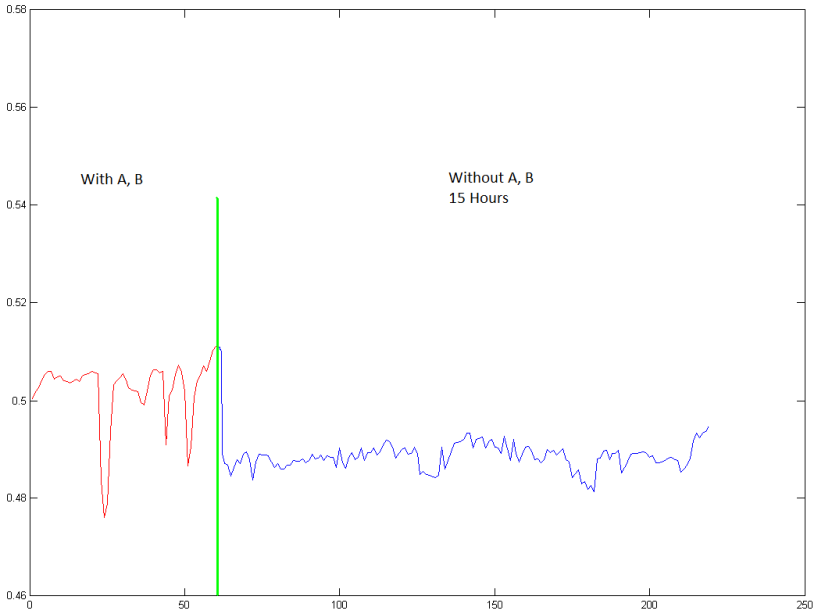
5.1.1 PoWER πάνω στο NAO

Ο αλγόριθμος εφαρμόστηκε με δύο παραλαγές:

- Εφαρμογή του αλγορίθμου με την χρήση των παραμέτρων α_i και b_i που συλλέγω από το Fourier.

- Εφαρμογή του αλγορίθμου χωρίς την χρήση των παραμέτρων α_i και b_i τις παραμέτρους αυτές.

Στο διάγραμμα που παρουσιάζεται παρακάτω (σχήμα 5.3) αποτυπώνονται δυο καμπύλες, η κόκκινη δείχνει τα reward όταν χρησιμοποιούμε τον αλγόριθμο Power με τις παραμέτρους α_i και b_i τις οποίες τις πήραμε από το μετασχηματισμό Fourier για τις τιμές που πήραμε από τα Motion File. Παρατηρούμε ότι οι τιμές ξεκινάνε από 0.5 και σιγά σιγά μαθαίνει ο αλγόριθμος και τις βελτιώνει. Παράλληλα στην μπλε καμπύλη βλέπουμε όταν ξεκινάει η μάθηση χωρίς αρχικές τιμές των παραμέτρων α_i και b_i παρατηρούμε μικρότερη τιμή των rewards στην αρχή της μάθησης και μετά βελτιώνεται αυτή η τιμή με τον χρόνο. Όμως η βελτίωση που έχουμε είναι πολύ μικρή σε σχέση με τον χρόνο που δαπανά ο αλγόριθμος στην μάθηση.



Σχήμα 5.3:

Με άλλα λόγια, εάν εφαρμόσουμε τον αλγόριθμο χωρίς αρχικοποίηση των τιμών των παραμέτρων θα χρειαστούμε 40 ώρες μάθησης περισσότερο από ότι αν εφαρμόσουμε αρχικοποίηση των μεταβλητών.

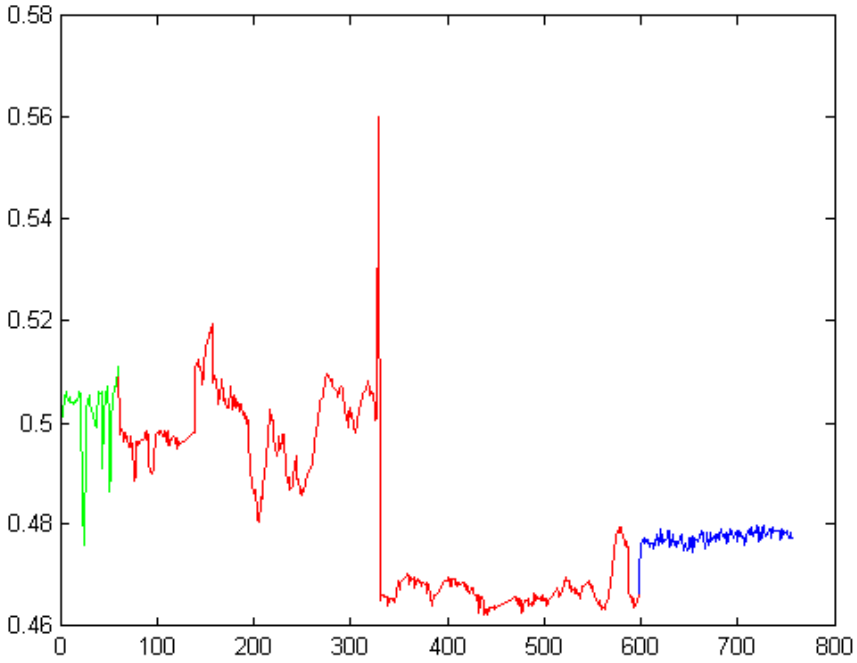
Το διάγραμμα του σχήματος (5.3) το αποκτήσαμε μετά από 99,000 episodes. Αν συνεχίζουμε την μάθηση με το PoWER θα καταλήξουμε στο σχήμα (5.4). Όπου εδώ βλέπουμε ότι ο αλγόριθμος φράσσεται στο διάστημα μεταξύ 0.47 και 0.48 (μπλε κομμάτι) και στην συνέχεια θα αποκτήσουμε το σχήμα (5.4).

Στον αλγόριθμο PoWER με αρχικοποίηση των παραμέτρων $Fourier$ συνεχίζουμε την μάθηση και παρατηρούμε ότι τα rewards αυξάνονται και η καμπύλη είναι πάντα ανοδική όπως φαίνεται στο σχήμα (5.5), όμως αυτές οι τροχιές που βρίσκω δεν είναι οι πραγματικές τροχιές του αλγορίθμου. Επειδή η διαδικασία μάθησης βελτίωνει με πολύ αργό ρυθμό την τροχιά, χρησιμοποιούμε ένα πολλαπλασιαστή για να επιταχύνουμε την διαδικασία.

Στην ουσία έχει γίνει μια αλλαγή στην τιμή των τροχιών, δηλαδή αυτά που θα μπαίνουν κάθε φορά στον υπολογισμό θα πρέπει να περάσουν σε μια από τις παρακάτω εξισώσεις:

$(q = q * 20)$ ή $(q = q * 40)$ ή $(q = q * 45)$
βέβαια ανάλογα με την μάθηση που κάνουμε κάθε φορά.

Ας κάνουμε μια ανάλυση του τι γίνεται με την αλλαγή των βαρών των q.



Σχήμα 5.4:

και ας μελετήσουμε το κομμάτι της καμπύλης πάνω στο σχήμα (5.6) με τα $45*q$. Παρατηρούμε ότι η καμπύλη με αρχικοποίηση των παραμέτρων α_i και b_i έχει μεγαλύτερο reward απο την αντίστοιχη καμπύλη που προέκυψε χωρίς την χρήση των παραμέτρων α_i και b_i. Άν είχε γίνει αρχικοποίηση των παραμέτρων, ο αλγόριθμος θα συνέχιζε να βγάζει καλύτερα αποτελέσματα εφόσον συνέχιζε να εκτελείται, στην αντίθετη περίπτωση όμως δεν θα συνέβαινε κάτι τέτοιο.

Παρακάτω παρατίθενται μερικά διαγράμματα όπως αυτά προέκυψαν από την εφαρμογή του αλγορίθμου PoWER, για διάφορες τιμές επεισοδίων και σε διάφορα μήκη χρόνων.

όπου:

$Batch$: Τροχιές.

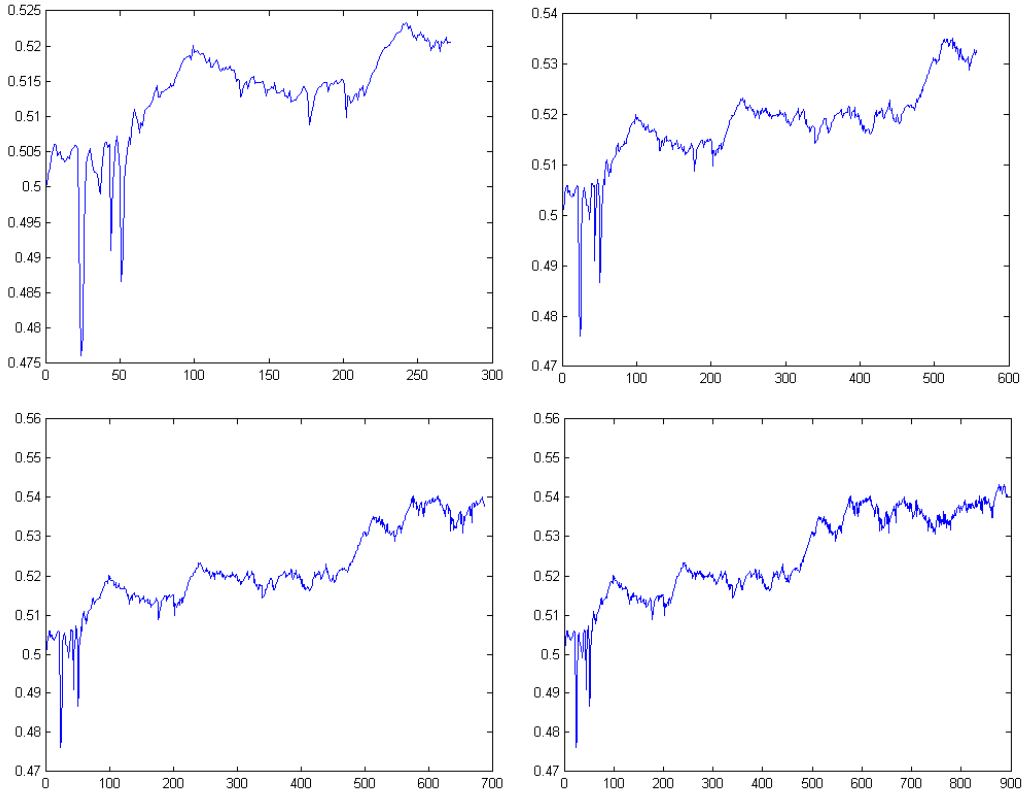
$NewBatch$: Καλύτερες Τροχιές.

$GlobalTime$: Χρόνο εκπαίδευσης.

BATCH=800 NEW-BATCH=200 global time=200 σχήμα(5.8)

BATCH=200 NEW-BATCH=200 global time=500 σχήμα(5.9)

BATCH=200 NEW-BATCH=200 global time=100 σχήμα(5.10)



Σχήμα 5.5:

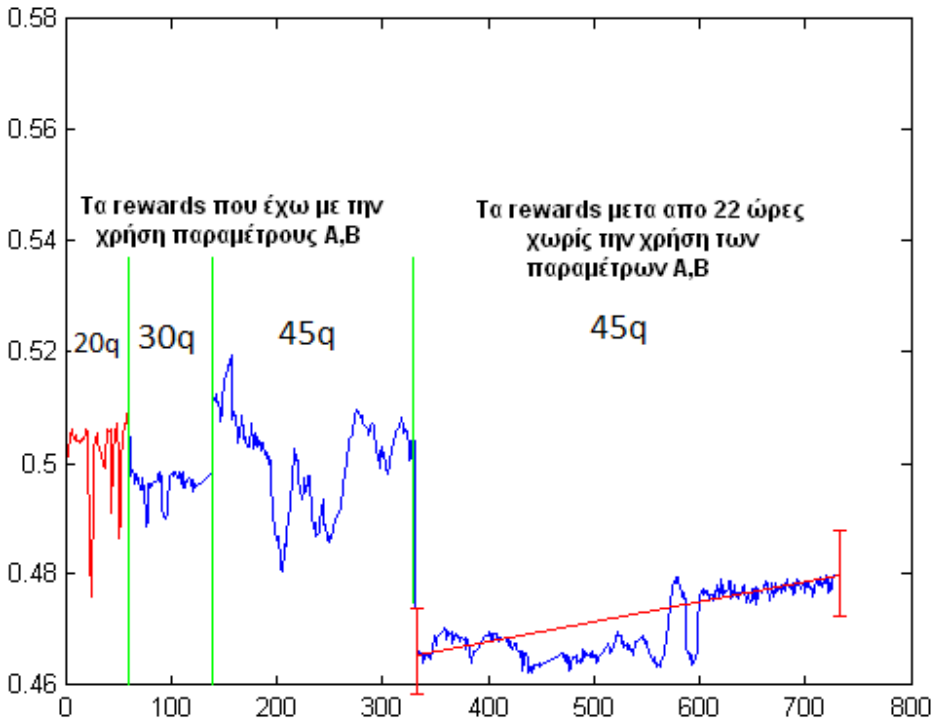
BATCH=200 NEW-BATCH=200 global time=1000 σχήμα(5.11)

BATCH=500 NEW-BATCH=200 global time=1000 σχήμα(5.12)

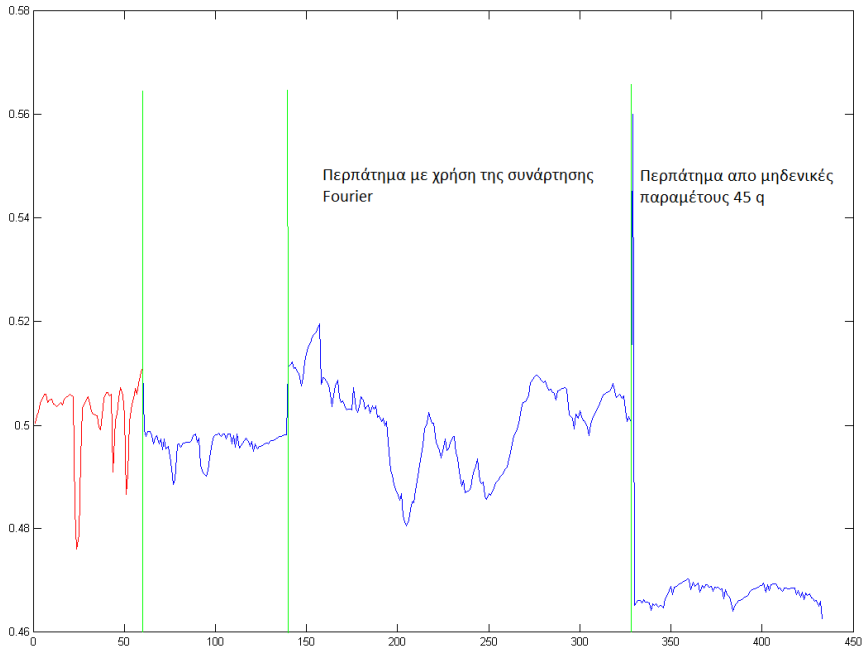
5.1.2 MCEM πάνω στο NAO

Σε αυτό τον αλγόριθμο εφαρμόσαμε μάθηση μόνο στην περίπτωση που έχουμε αρχικοποίηση των τιμών των παραμέτρων α_i και b_i. Επίσης σε αυτό τον αλγόριθμο εφαρμόσαμε αναζήτηση επί γραμμής και τον αλγόριθμο της φυσαλίδας και ο λόγος ήταν κάθε φορά να επιλέγονται οι καλύτερες τροχιές και να εκτελείται η μάθηση με αυτές. Εφαρμόσαμε περίπου 6,000,000 episode σε 5 πραγματικές μέρες.

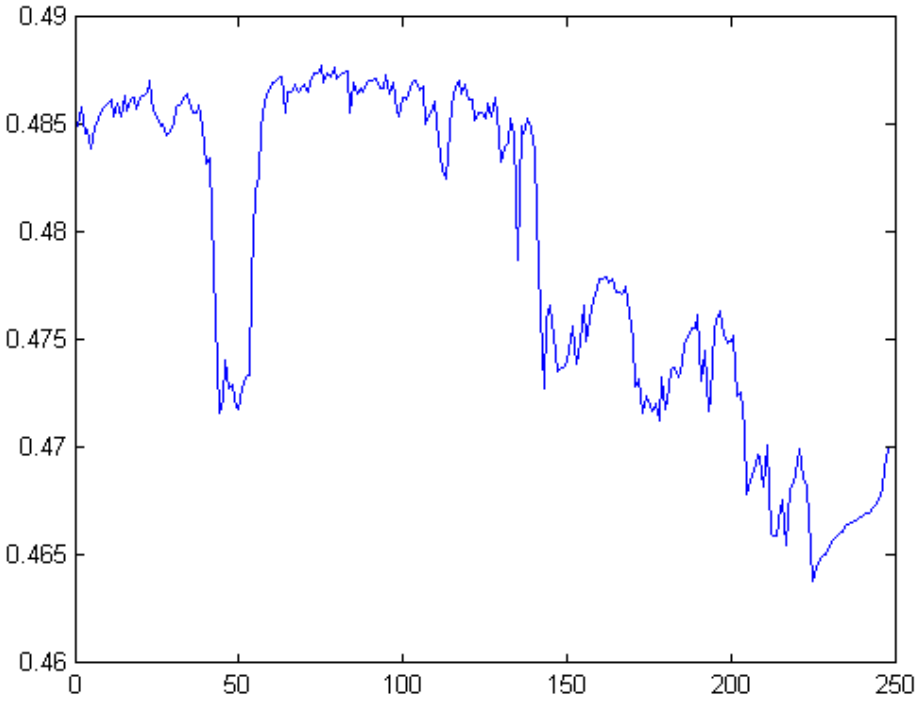
Παρατηρούμε τα rewards που έχουν κατά μέσο όρο ανοδική τιμή, και το περπάτημα του ρομπότ ήταν πολύ καλό,σχήμα (5.13), σχήμα (5.14)


Τα rewards που έχω με την χρήση παραμέτρους A,B
Τα rewards μετα απο 22 ώρες χωρίς την χρήση των παραμέτρων A,B
20q
30q
45q
45q
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0
100
200
300
400
500
600
700
800

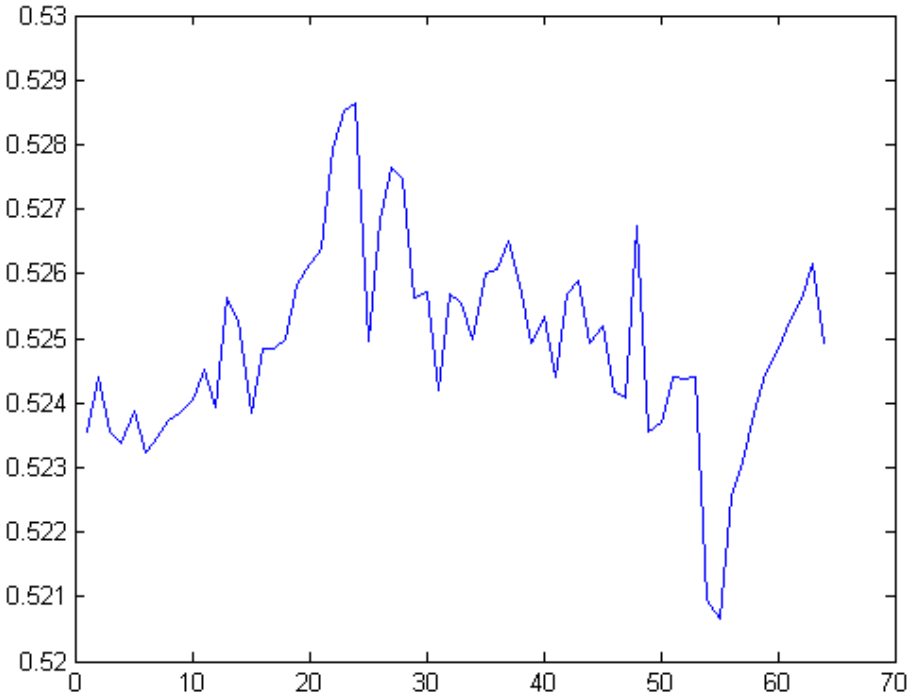
Σχήμα 5.6:


Περπάτημα με χρήση της συνάρτησης Fourier
Περπάτημα απο μηδενικές παραμέτους 45 q
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0
50
100
150
200
250
300
350
400
450

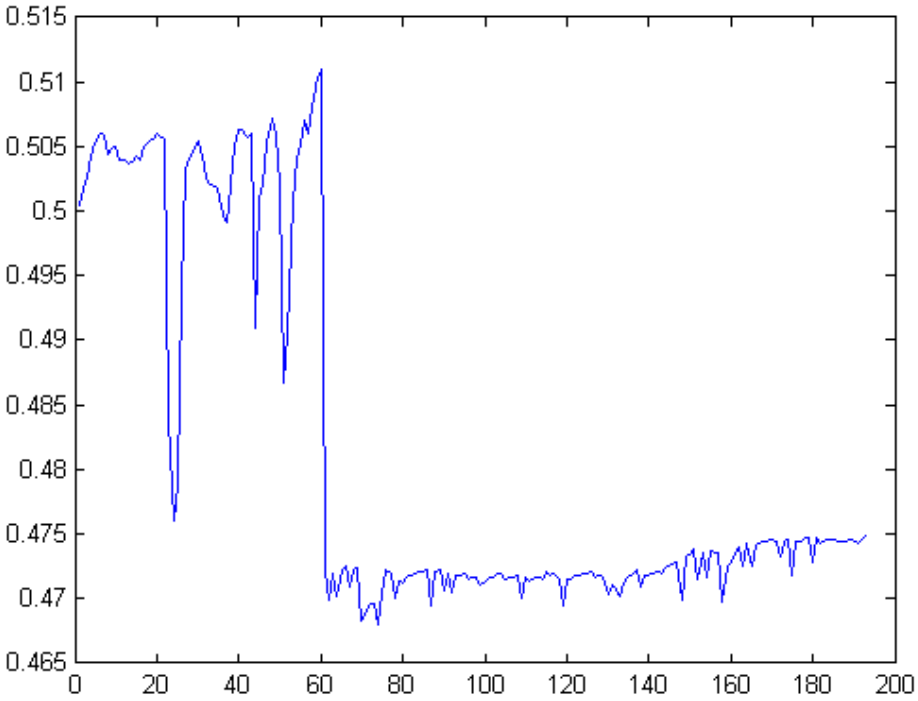
Σχήμα 5.7:



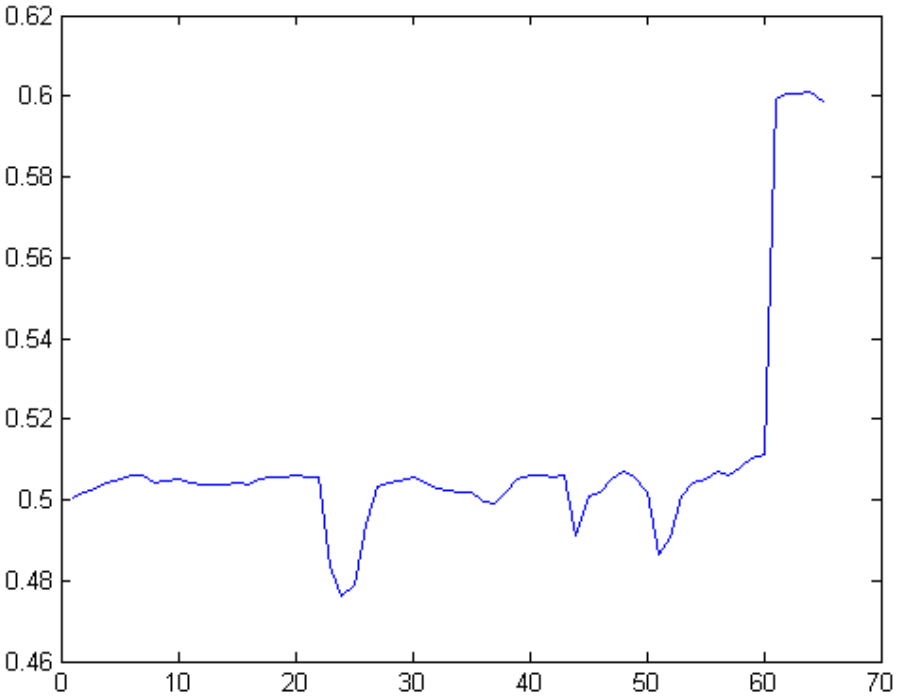
Σχήμα 5.8:


0.53
0.529
0.528
0.527
0.526
0.525
0.524
0.523
0.522
0.521
0.52
0
10
20
30
40
50
60
70

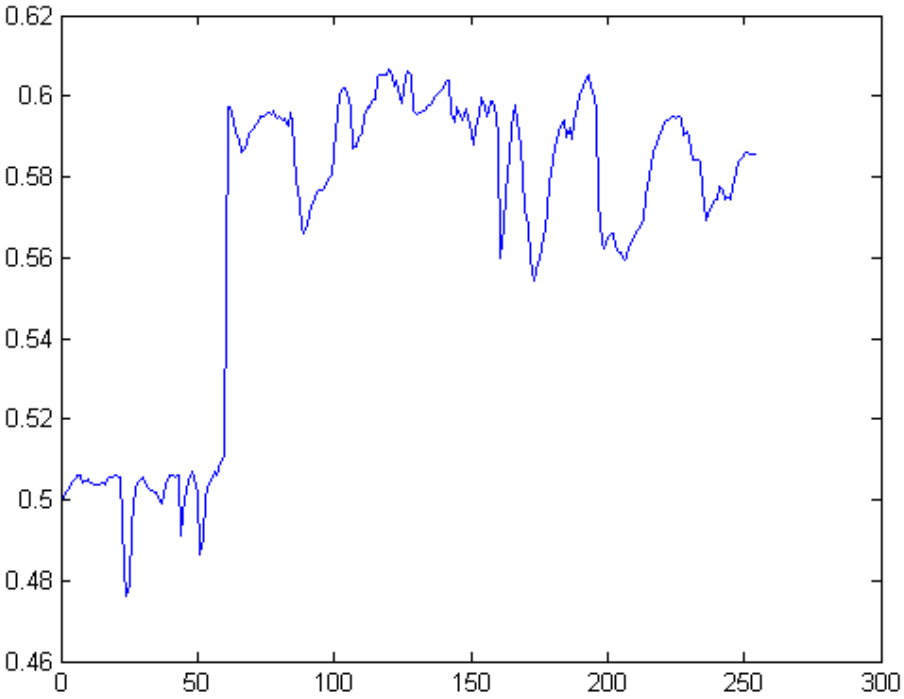
Σχήμα 5.9:


0.515
0.51
0.505
0.5
0.495
0.49
0.485
0.48
0.475
0.47
0.465
0
20
40
60
80
100
120
140
160
180
200

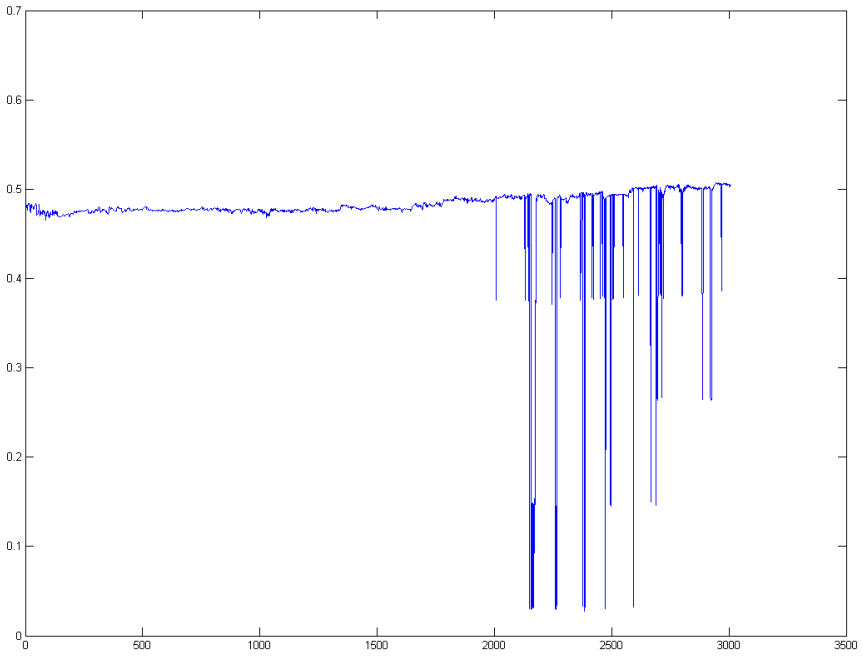
Σχήμα 5.10:



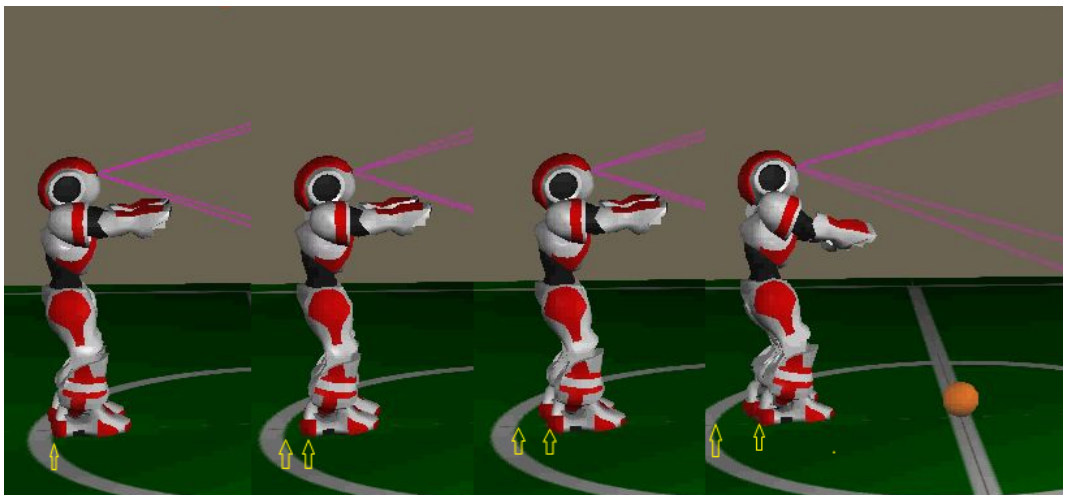
Σχήμα 5.11:



Σχήμα 5.12:



Σχήμα 5.13:



Σχήμα 5.14:

Κεφάλαιο 6

Βιβλιογραφία

-Richard S. Sutton and Andrew G. Barto. Reinforcement Learning : An Introduction, MIT press, Cambridge, Massachusetts, 1998

-N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis. Learning Model-free Robot Control by a Monte Carlo EM Algorithm. Autonomous Robots, 27(2):123-130, 2009.

-Lagoudakis, M.G. and Parr, R. Least-squares policy iteration. The Journal of Machine Learning Research, 2003

-Aldebaran Robotics-www.aldebaran-robotics.com

-Robotstadium-www.robotstadium.org