ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Τ**ΟΜΕΑΣ **Σ**ΥΣΤΗΜΑΤΩΝ **Ε**ΛΕΓΧΟΥ ΚΑΙ **Ρ**ΟΜΠΟΤΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΕΚΜΑΘΗΣΗ ΣΕ ΕΙΚΟΝΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΟΠΤΙΚΑ ΟΔΗΓΟΥΜΕΝΗΣ ΡΟΜΠΟΤΙΚΗΣ ΛΑΒΗΣ ΜΕ ΠΟΛΥΣΤΡΩΜΑΤΙΚΟ ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ

Σπυρίδων Καραχάλιος

**Επιβλέπων : Κωνσταντίνος Τζαφέστας**

Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Τ**ΟΜΕΑΣ **Σ**ΥΣΤΗΜΑΤΩΝ **Ε**ΛΕΓΧΟΥ ΚΑΙ **Ρ**ΟΜΠΟΤΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΕΚΜΑΘΗΣΗ ΣΕ ΕΙΚΟΝΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΟΠΤΙΚΑ ΟΔΗΓΟΥΜΕΝΗΣ ΡΟΜΠΟΤΙΚΗΣ ΛΑΒΗΣ ΜΕ ΠΟΛΥΣΤΡΩΜΑΤΙΚΟ ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ

Σπυρίδων Καραχάλιος

**Επιβλέπων : Κωνσταντίνος Τζαφέστας**
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε το Νοέμβριο του 2010 από την τριμελή εξεταστική επιτροπή :

| …….. | ……. | …….…… |
|---|---|---|
| **Κωνσταντίνος Τζαφέστας** | **Νικόλαος Μαράτος** | **Γεώργιος Στάμου** |
| Επίκουρος Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Λέκτορας |

Αθήνα, Νοέμβριος 2010

…….
Σπυρίδων Καραχάλιος

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

# Περίληψη

Η έρευνα για την δημιουργία ρομποτικών χειριστών με εφαρμογές στην καθημερινότητα του ανθρώπου αναπτύσσεται, ραγδαίως, τα τελευταία χρόνια. Στην έρευνα αυτή συμβάλλουν πολλοί επιστημονικοί τομείς, όπως είναι η τεχνητή νοημοσύνη, η αλληλεπίδραση ανθρώπου – μηχανής, η όραση υπολογιστών, η βιολογία και η ψυχολογία. Σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι η δημιουργία ενός αυτοματοποιημένου προγράμματος για την εκ των προτέρων εκπαίδευση ενός ρομποτικού χειριστή στο να προσεγγίζει και να πιάνει ορισμένα αντικείμενα χρησιμοποιώντας αποκλειστικά οπτική (μονοσκοπική) πληροφορία. Στόχο της εργασίας αποτελεί η διερεύνηση της δυνατότητας εκπαίδευσης του ρομποτικού χειριστή με τρόπο αυτοματοποιημένο επί ενός εικονικού περιβάλλοντος.

Στο πρώτο στάδιο κατασκευής του συνόλου εκπαίδευσης της οπτικά οδηγούμενης ρομποτικής λαβής, κατασκευάζονται με τρόπο συστηματικό και αυτοματοποιημένο, διαδοχικά στιγμιότυπα επιτυχούς ρομποτικής λαβής αντικειμένων γνωστής γεωμετρίας επί ενός εικονικού περιβάλλοντος ρομποτικής εργασίας. Το εικονικό αυτό περιβάλλον υλοποιήθηκε χρησιμοποιώντας τη βιβλιοθήκη τριδιάστατων γραφικών OpenGL, ώστε να είναι άμεσα μεταφέρσιμο σε διαφορετικές προγραμματιστικές πλατφόρμες. Η διαδικασία αυτοματοποιημένης λήψης διαδοχικών στιγμιοτύπων εκπαίδευσης από το κεντρικό περιβάλλον εικονικής προσομοίωσης της ρομποτικής εργασίας, ακολουθείται από την αυτόματη εισαγωγή των οπτικών δεδομένων (raw data) σε ένα άλλο υπολογιστικό περιβάλλον προσομοίωσης (υλοποιημένου σε Matlab) όπου εκτελούνται όλες οι διαδικασίες που αφορούν στην εκμάθηση της ρομποτικής λαβής, δηλαδή τη συσχέτιση οπτικών περιγραφέων των αντικειμένων με τις μεταβλητές ελέγχου του ρομποτικού χειριστή. Η διαδικασία αυτή περιλαμβάνει: (α) κατάτμηση κάθε εικόνας για την εξαγωγή του αντικειμένου, διαδικασία η οποία στην παρούσα εργασία βασίζεται σε χρωματική πληροφορία (μετατροπή του συστήματος απεικόνισης χρωμάτων από το σύστημα RGB στο σύστημα HSL και εν συνεχεία κατάτμηση της εικόνας με βάση την απόχρωση του αντικείμενου), (β) Εξαγωγή χαρακτηριστικών περιγραφέων σχήματος του απεικονιζόμενου αντικειμένου και αποθήκευση των δεδομένων αυτών σε ένα αρχείο εκπαίδευσης. Στο ίδιο αρχείο αποθηκεύονται και τα δεδομένα του ρομποτικού χειριστή, δηλαδή, η θέση και ο προσανατολισμός της ρομποτικής αρπάγης.

Ακολούθως, για την εκπαίδευση του ρομποτικού χειριστή, επιλέχθηκε η δημιουργία ενός πολυστρωματικού νευρωνικού δικτύου προσοτροφοδότησης. Το νευρωνικό δίκτυο αυτό, αποτελείται από 7 νευρώνες στο επίπεδο εισόδου, 15 νευρώνες στο πρώτο κρυμμένο επίπεδο, 10 νευρώνες στο δεύτερο κρυμμένο επίπεδο και 4 νευρώνες στο επίπεδο εξόδου. Ύστερα από την αυτοματοποιημένη και συστηματική συλλογή των δεδομένων εκπαίδευσης για ένα πλήθος αντικειμένων και των αντίστοιχων δεδομένων του ρομποτικού χειριστή, εισάγονται τα δεδομένα αυτά ως είσοδοι και έξοδοι αντίστοιχα στο νευρωνικό δίκτυο. Το νευρωνικό δίκτυο εκπαιδεύεται χρησιμοποιώντας ως πρότυπο μάθησης την επιβλεπόμενη μάθηση με βάση τον αλγόριθμο Levenberg – Marquardt και ως μέθοδο αναπροσαρμογής των συναπτικών βαρών του νευρωνικού δικτύου τη μέθοδο διόρθωσης σφάλματος. Συνεπώς, μετά την εκπαίδευση, το νευρωνικό δίκτυο και κατά συνέπεια ο ρομποτικός χειριστής, είναι σε θέση παίρνοντας ως είσοδο οποιοδήποτε αντικείμενο σε οποιαδήποτε θέση να δίνει ως έξοδο την θέση και τον προσανατολισμό της ρομποτικής αρπάγης.

Τέλος, έγινε έλεγχος της γενίκευσης της εκπαίδευσης του ρομποτικού χειριστή σε ένα πλήθος από σύνολα ελέγχου, τα οποία περιλαμβάνουν αντικείμενα με γνωστό ύψος και άγνωστο μήκος, αντικείμενα με άγνωστο ύψος και άγνωστο ύψος, καθώς και αντικείμενα

τόσο με παρόμοια όσο και με διαφορετική γεωμετρία με τα αντικείμενα της εκπαίδευσης. Τα αποτελέσματα από την προσομοίωση έδειξαν ότι ο ρομποτικός χειριστής είναι σε θέση να προσεγγίζει και να πιάνει σωστά όλα τα αντικείμενα εκτός από αυτά που δεν παρουσιάζουν γεωμετρική ομοιότητα με τα αντικείμενα στα οποία έχει εκπαιδευτεί το ρομπότ.

# Λέξεις – κλειδιά

Αλληλεπίδραση ανθρώπου – ρομπότ, εκμάθηση ρομπότ, εικονικό περιβάλλον, ρομποτική όραση, οπτικά καθοδηγούμενη ρομποτική λαβή, οπτικοί περιγραφείς σχήματος, εξαγωγή οπτικών χαρακτηριστικών, κατάτμηση εικόνας, νευρωνικό δίκτυο

# Abstract

The research to design robots with applications in human's everyday routine is developing rapidly in the latest years. A lot of scientific sectors contribute in this research such as artificial intelligence, human – robot interaction, computer vision, biology and psychology. The aim of this particular diploma thesis is the creation of an automated program for prior teaching a robotic manipulator to approach and grasp certain objects using only optical (monoscopic) information. The investigation of the possibility of automatically teaching a robotic manipulator in a virtual environment constitutes the objective of this work.

In the first stage of constructing the training set for teaching a visually guided robotic grasping, consecutive snapshots of successful grasping of objects with known geometry are taken in the virtual environment, in a systematic and automatic manner. The virtual environment was created using OpenGL 3D graphics library, in order to make the program directly portable to different programming platforms. The automatic process of taking the successive training snapshots within the virtual environment is followed by the automatic import of these raw data to another simulation environment (implemented in Matlab). In this environment, all the robot learning processes are executed, which concern teaching of the robotic grasping mechanism, and more specifically associating the optical descriptors of the objects with the control variables of the robotic manipulator. This process consists of: (a) image segmentation to extract the object, a process which in this work is based on colour information (transformation of the colour system of the image from RGB to HSL system and then segmentation of the image using hue values), (b) extraction of specific shape descriptors of the object and storing of these data in a training file. The data of the robotic manipulator (world coordinates and orientation of the robotic gripper) are also saved in the same training file.

At the following stage, the creation of a multilayer feedforward neural network was chosen in order to train the robotic manipulator. This neural network has 4 neurons in the input layer, 15 neurons in the first hidden layer, 10 neurons in the second hidden layer and 4 neurons in the output layer. After the systematic and automatic collection of the training data for a number of objects and the respective data of the robotic manipulator, the data are imported as input and output data respectively in the neural network. The neural network is then trained using the Levenberg – Marquardt algorithm as a base for supervised learning and error correction as the method for changing the weights of the neural network. As a result, after training the robot, the neural network having as input any object in any location is capable of giving as output the coordinates and the orientation of the robotic hand.

Finally, a number of tests have been performed for the generalization of the training of the robotic manipulator with a variety of test sets that consist of objects with known height and unknown length, objects with unknown height and length and objects with either similar or different geometry with the training objects. The results of the simulation in the virtual environment showed that the robotic manipulator is capable of grasping correctly all the objects, apart from those objects that have significantly different geometry from the training objects.

# Keywords

Human – robot interaction, robot learning, virtual environment, robot vision, visual robot grasping, visual shape descriptors, image feature extraction, colour image segmentation, neural network

# Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το επιστέγασμα των σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο. Την ανάθεση του θέματος που από την πρώτη στιγμή βρήκα ιδιαίτερα ενδιαφέρον, οφείλω στον επιβλέποντα καθηγητή μου κ. Κωνσταντίνο Τζαφέστα.

Από τη θέση αυτή θα ήθελα να τον ευχαριστήσω θερμά για την πολύτιμη καθοδήγησή του και τις ουσιαστικές παρατηρήσεις του που συνέβαλαν αποφασιστικά στην ποιότητα αυτής της εργασίας. Τον ευχαριστώ επίσης για την υπομονή του, το χρόνο που μου διέθεσε, την εμπιστοσύνη που μου έδειξε και την γενικότερη υποστήριξή του.

Ακόμα, ευχαριστίες οφείλω στην οικογένειά μου για τις κατάλληλες συνθήκες που επέτρεψαν να επιδοθώ αναπόσπαστα στις σπουδές μου, για τη συμπαράστασή τους όταν τα αποτελέσματα δεν ήταν ικανοποιητικά και για τη βοήθειά τους όταν η καθημερινότητα ήταν δύσκολη.

Τέλος θα ήθελα να ευχαριστήσω θερμά την συνεργάτιδα και φίλη μου Γεωργία Μώτση που όλο αυτόν τον καιρό της συγγραφής αυτής της διπλωματικής έδειξε ιδιαίτερη υπομονή, συμπαράσταση και ψυχική βοήθεια ενώ παράλληλα εκπονούσε και η ίδια την δική της διπλωματική εργασία.

# Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

# ΕΙΣΑΓΩΓΗ

## 1.1 Επικοινωνία Ανθρώπου - Μηχανής

Στη σημερινή εποχή, η διαρκής εξέλιξη των υπολογιστών οδηγεί, σε μια ολοένα και μεγαλύτερη ανάπτυξη νέων εφαρμογών ευφυούς ελέγχου των μηχανών όπως είναι και οι ρομποτικοί χειριστές. Είναι, λοιπόν, αναγκαίο να βρεθούν φυσικοί τρόποι αλληλεπίδρασης μεταξύ των ανθρώπων και των μηχανών, έτσι ώστε να είναι ευκολότερη η επικοινωνία των απλών ανθρώπων, χωρίς ιδιαίτερες γνώσεις, με τα ρομπότ.

Στην κατεύθυνση αυτή συνεργάζονται πολλοί επιστημονικοί τομείς, όπως είναι η τεχνητή νοημοσύνη, η αλληλεπίδραση ανθρώπου-μηχανής, η όραση υπολογιστών, η βιολογία και η ψυχολογία. Οι επιστημονικοί τομείς που ασχολούνται, κυρίως, με την επικοινωνία του ανθρώπου και της μηχανής χρησιμοποιούν τη γνώση που τους παρέχουν οι τομείς που ασχολούνται με τον άνθρωπο και τον τρόπο λειτουργίας και συμπεριφοράς του και προσπαθούν να ανακαλύψουν νέες μεθόδους και νέα προγράμματα διαδραστικής επικοινωνίας των ανθρώπων με τις μηχανές.

Στην παρούσα εργασία, θα ασχοληθούμε με την δυνατότητα εκπαίδευσης του ρομποτικού χειριστή με τρόπο αυτοματοποιημένο επί ενός εικονικού περιβάλλοντος.

## 1.2 Επικοινωνία Ανθρώπου - Υπολογιστή

Ένας υπολογιστής μπορεί να επικοινωνεί με το περιβάλλον και κατά συνέπεια και με τον άνθρωπο με πολλούς τρόπους.

Ένας τρόπος, που χρησιμοποιήθηκε από την αρχή, σχεδόν, της δημιουργίας των υπολογιστών, ήταν με ένα έτοιμο πρόγραμμα καταχωρημένο στη μνήμη του υπολογιστή στο οποίο εισάγει δεδομένα ο άνθρωπος με κάποιες συσκευές εισαγωγής δεδομένων, όπως είναι το ποντίκι, το πληκτρολόγιο και άλλες. Αυτός είναι και ο λεγόμενος εκ των προτέρων προγραμματισμός ενός υπολογιστή, ο οποίος περιμένει συγκεκριμένες εισόδους από τον άνθρωπο για να λειτουργήσει.

Στη συνέχεια, ο άνθρωπος ανέπτυξε διατάξεις που θα μπορούσαν να χρησιμοποιηθούν ως άμεσοι καταγραφείς κινήσεων και ενεργειών του ανθρώπου, για την εισαγωγή δεδομένων σε εφαρμογές εικονικής πραγματικότητας. Μια τέτοια διάταξη, που πρωτοαναπτύχθηκε είναι τα γάντια δεδομένων (data gloves) με τα οποία ο άνθρωπος μπορεί να στέλνει δεδομένα σε έναν υπολογιστή είτε ενσύρματα είτε ασύρματα και εκείνος με τη σειρά του να τα επεξεργάζεται. Έτσι έγινε δυνατή η πολύ γρήγορη και άμεση επικοινωνία του ανθρώπου με τον υπολογιστή μεταφέροντας, έτσι, και πολλά δεδομένα του περιβάλλοντος [1].

Ακόμα αναπτύχθηκε αρκετά και η χρήση των καμερών που παίζουν το ρόλο των ματιών του υπολογιστή. Όπως γίνεται αντιληπτό, ο υπολογιστής, πλέον, αποκτά μια

καλύτερη αίσθηση του περιβάλλοντος και οι δυνατότητές του αυξάνονται σε μεγάλο ποσοστό. Οι υπολογιστές αποκτούν την δυνατότητα με χρήση κατάλληλων προγραμμάτων να αντιλαμβάνονται αντικείμενα που βρίσκονται στο χώρο, χειρονομίες των ανθρώπων ή και ακόμα ολόκληρες κινήσεις με συγκεκριμένο σκοπό από τους ανθρώπους [2].

Τέλος, ένας άλλος τρόπος επικοινωνίας, ο οποίος αναπτύσσεται, πλέον, πολύ τα τελευταία χρόνια είναι με φυσική γλώσσα, δίνοντας, δηλαδή, απευθείας φωνητικές εντολές στους υπολογιστές, καθώς, επίσης, και η σύγχρονη τάση καταγραφής της «πρόθεσης» και της «σκέψης» του ανθρώπου μέσω νέων διατάξεων (Brain – Computer Interfaces – BCI) [3].

## 1.3 Χειρισμός ρομπότ από τον άνθρωπο

Ο άνθρωπος, στη συνέχεια, ήρθε αντιμέτωπος με ένα καινούριο πρόβλημα – τον χειρισμό των ρομπότ. Ο χειρισμός των ρομπότ, από τον άνθρωπο, με την χρήση του υπολογιστή μπορεί να γίνει σε δύο κόσμους, τον πραγματικό και τον εικονικό.

Στον πραγματικό κόσμο, ο άνθρωπος έχει τη δυνατότητα να επιδεικνύει μία κίνηση σε ένα ρομπότ με δύο τρόπους. Μπορεί, αρχικά, είτε να χειρίζεται απευθείας ένα ρομπότ από κοντά καθοδηγώντας το είτε από μακριά με κάποιο είδος τηλεχειρισμού. Επίσης, μπορεί να εκτελεί ο άνθρωπος την κίνηση και το ρομπότ να καταγράφει την κίνηση αυτή με δικούς του αισθητήρες. Στην πρώτη περίπτωση, η αποθήκευση της κίνησης είναι άμεση καθώς όλα τα δεδομένα αποθηκεύονται στο χώρο κίνησης του ρομπότ, ενώ στην δεύτερη περίπτωση, τα δεδομένα είναι έμμεσα, καθώς η καταγραφή της κίνησης γίνεται στο χώρο κίνησης του ανθρώπου [4].

Ωστόσο, λόγω των δυσκολιών που προκύπτουν στον πραγματικό κόσμο, όπως είναι η επιλογή και η κατασκευή του κατάλληλου ρομποτικού χειριστή, ο περιορισμός στα λάθη καθώς αυτά μπορεί να κοστίζουν ή να καταστρέφουν το ρομπότ και άλλα, καθώς και η ραγδαία εξέλιξη των υπολογιστών, έχουν ωθήσει τους ανθρώπους στο να δημιουργούν ένα εικονικό περιβάλλον στο οποίο θα γίνεται ο χειρισμός των ρομπότ. Στο εικονικό αυτό περιβάλλον, οι άνθρωποι έχουν τη δυνατότητα να χειρίζονται απευθείας το ρομπότ και να του επιδεικνύουν κινήσεις, καθώς και να πειραματίζονται πολύ γρήγορα με νέες μεθόδους, χωρίς αυτό να επιφέρει οικονομικά ή άλλα προβλήματα.

Στην παρούσα εργασία δημιουργήθηκε ένα τέτοιο εικονικό περιβάλλον. Ο προγραμματισμός έγινε σε προγραμματιστική γλώσσα C++ με χρήση της βιβλιοθήκης OpenGL έτσι ώστε να δημιουργηθεί ένας τρισδιάστατος χώρος και για να μπορεί να μεταφερθεί άμεσα το πρόγραμμα σε διαφορετικές προγραμματιστικές πλατφόρμες. Στο πρόγραμμα αυτό, του οποίου ο κώδικας δίνεται στο Παράρτημα Α, έγιναν όλες οι απαραίτητες διαδικασίες για την δημιουργία του περιβάλλοντος, του ίδιου του ρομπότ, των αντικειμένων πάνω στα οποία έγινε ο πειραματισμός καθώς και σε συνδυασμό με το μαθηματικό περιβάλλον του προγράμματος Matlab η εξαγωγή και αποθήκευση όλων των δεδομένων και των συμπερασμάτων (κώδικας Παραρτήματος Β).

## 1.4 Εκμάθηση ρομποτικού χειριστή από τον άνθρωπο

Παρά το γεγονός ότι έχουμε αποθηκεύσει τα σωστά δεδομένα της κίνησης κάποιου ανθρώπου που θέλουμε να εκτελέσει το ρομπότ, στη μνήμη του ρομπότ-υπολογιστή, ουσιαστικά δεν έχουμε καταφέρει τίποτα ακόμα καθώς το ρομπότ είναι σε θέση να αναπαράγει μόνο την κίνηση που του δείξαμε νωρίτερα. Έτσι, λοιπόν, είναι αναγκαίο να

γενικεύσουμε σε ένα πλήθος κινήσεων, να εκπαιδεύσουμε, δηλαδή, το ρομπότ στο να κάνει ορισμένες κινήσεις σε οποιαδήποτε κατάσταση ή οποιοδήποτε περιβάλλον βρεθεί. Αυτό, βέβαια, θα πρέπει να γίνει χωρίς να χρειάζεται να δείξουμε πολλές κινήσεις στο ρομπότ αφού θα πρέπει όσο το δυνατόν να μειωθεί ο χρόνος που ο άνθρωπος θα εκπαιδεύει το ρομπότ. Στην κατεύθυνση αυτή έχουν ασχοληθεί αρκετοί επιστημονικοί τομείς και υπάρχει και αρκετή βιβλιογραφία ως προς τον τρόπο εκμάθησης του ρομπότ από τον άνθρωπο.

Πρέπει, λοιπόν, να βρεθεί ένα είδος συνάρτησης, ή διαδικασία αντιστοίχησης (αναλυτική ή μη), η οποία ως είσοδο θα έχει ορισμένα χαρακτηριστικά του περιβάλλοντος και ως έξοδο κάποια κατάλληλη ενέργεια από το ρομπότ. Το ποιά θα είναι η μέθοδος προσδιορισμού της συνάρτησης αυτής προκύπτει από διάφορους παράγοντες. Εξαρτάται από το αν η είσοδος και η έξοδος είναι συνεχής ή διακριτές τιμές, αν η μέθοδος χρησιμοποιεί δεδομένα για την προσέγγιση της συνάρτησης πριν ή κατά τη διάρκεια της κίνησης του ρομπότ, από το αν είναι αδιάφορο ή θεμιτό να κρατάμε τα δεδομένα της επίδειξης καθ' όλη τη διάρκεια της εκπαίδευσης ή αν συνεχώς τα δεδομένα αυτά ανανεώνονται.

Γενικά οι στατιστικές μέθοδοι προσέγγισης μιας συνάρτησης εμπίπτουν σε δύο μεγάλες κατηγορίες ανάλογα με το αν η έξοδος που πρόκειται να προβλεφθεί είναι διακριτή ή συνεχής. Οι μέθοδοι ταξινόμησης παράγουν διακριτές εξόδους, ενώ οι μέθοδοι παλινδρόμησης παράγουν συνεχής εξόδους. Έχουν αναπτυχθεί πολλές τεχνικές ταξινόμησης και παλινδρόμησης, οι οποίες αναλύονται διεξοδικά σε βιβλίο της βιβλιογραφίας [5].

## 1.5 Μέθοδοι εκμάθησης μηχανής από τον άνθρωπο στη βιβλιογραφία

### 1.5.1 Μέθοδοι ταξινόμησης (Classification)

Οι προσεγγίσεις ταξινόμησης κατηγοριοποιούν τις εισόδους σε διακριτές τάξεις, συγχωνεύοντας, έτσι, όμοιες τιμές εισόδων μαζί. Στα πλαίσια της πολιτικής εκμάθησης, η είσοδος σε έναν ταξινομητή είναι οι διάφορες καταστάσεις του ρομπότ, ενώ ως διακριτή έξοδος είναι κάποια ενέργεια του ρομπότ. Παρακάτω παρουσιάζουμε μια περίληψη των μεθόδων ταξινόμησης που υπάρχουν στη βιβλιογραφία οι οποίες εφαρμόζονται σε τρία επίπεδα ενέργειας (βασικός έλεγχος κίνησης, πρωτογενείς ενέργειες, περίπλοκες συμπεριφορές).

Οι χαμηλού επιπέδου ενέργειες ενός ρομπότ αφορούν βασικές εντολές, όπως κίνηση προς τα μπροστά ή στρίψιμο. Παραδείγματα εφαρμογών που προσεγγίζουν μια συνάρτηση από καταστάσεις σε χαμηλού επιπέδου ενέργειες περιλαμβάνουν τον έλεγχο ενός αυτοκινήτου σε μια προσομοίωση οδήγησης χρησιμοποιώντας Μοντέλα Μίξης Γκαουσσιανών (Gaussian Mixture Models – GMMs) [6], πτήση ενός αεροπλάνου σε προσομοίωση χρησιμοποιώντας δέντρα αποφάσεων [7] και εκμάθηση αποφυγής εμποδίων και πλοήγησης χρησιμοποιώντας ένα Δίκτυο του Bayes (Bayesian Network) [8] και k-Κοντινότερους Γείτονες (k-Nearest Neighbours – kNN) [9] ως ταξινομητές.

Όταν η έξοδος είναι κάποια λίγο πιο σύνθετη κίνηση τότε η κίνηση αυτή διασπάται σε πρωτογενή επιμέρους τμήματα τα οποία συντίθενται αποτελώντας, έτσι, το κάθε κομμάτι μία απλή κίνηση. Για παράδειγμα, οι Pook και Ballard [10] ταξινομούν τις πρωτόγονες κινήσεις χρησιμοποιώντας kNN και μετά αναγνωρίζουν το κάθε κομμάτι μιας πιο σύνθετης κίνησης με χρήση των Κρυφών Μοντέλων του Markov (Hidden Markov Models – HMMs), για να εκπαιδεύσουν ένα ρομποτικό χέρι και βραχίονα στο να γυρίζουν ένα τηγανισμένο

αυγό στο τηγάνι. Άλλα παραδείγματα χρησιμοποιούν HMMs για να εκπαιδεύσουν ένα ρομπότ για μία απλή συναρμολόγηση [11] ή για απλές κινήσεις στο χώρο προσδιορίζοντας και γενικεύοντας τις προθέσεις του χρήστη [12]. Ένα βιολογικά εμπνευσμένο πλαίσιο εργασίας βρίσκει αυτόματα τις πρωτόγονες κινήσεις από τα δεδομένα της επίδειξης, κατατάσσει τα δεδομένα με διανυσματική κβαντοποίηση και στη συνέχεια συνθέτει τις πρωτόγονες αυτές κινήσεις σε ένα ιεραρχικό Νευρωνικό Δίκτυο [13]. Αυτό το είδος εργασίας εφαρμόζεται σε πολλών ειδών πειράματα, που περιλαμβάνουν ένα προσομοιωμένο ανθρωπόμορφο κορμί με 20 βαθμούς ελευθερίας, σε μια ανθρωπόμορφη προσομοίωση που αντιδρά σε εξωτερικούς αισθητήρες με 37 βαθμούς ελευθερίας, σε σκυλιά AIBO της Sony και σε μικρά ρομπότ της Pioneer με διαφορική κίνηση. Με αυτό το πλαίσιο, το προσομοιωμένο ανθρωπόμορφο κορμί εκπαιδεύεται σε πληθώρα κινήσεων χορού, αεροβικής και αθλητικών κινήσεων [14], η ανθρωπόμορφη προσομοίωση σε τρόπους προσέγγισης αντικειμένων [15] και τα Pioneer σε αλληλουχίες κίνησης στο χώρο σε συγκεκριμένα μονοπάτια [16].

Παρόμοιες προσεγγίσεις, έχουν χρησιμοποιηθεί και για την ταξινόμηση υψηλού επιπέδου συμπεριφορών. Μία μέτρηση στην ομοιότητα της αναπαράστασης ενός ιδιοδιανύσματος αναγνωρίζει χειρονομίες ενός ανθρωπομορφικού χεριού [17] και μέσα σε αυτό το πλαίσιο τα HMMs ταξινομούν τις επιδείξεις σε χειρονομίες για μια εργασία ταξινόμησης κουτιών με ένα ρομπότ της Pioneer [18]. Ο κανόνας πιθανοτήτων του Bayes διαλέγει ενέργειες για ένα ανθρωπόμορφο ρομπότ σε μια εργασία πατήματος κουμπιών [19] και οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines – SVMs) ταξινομούν συμπεριφορές για μια ρομποτική εργασία ταξινόμησης μπαλών [20].

### 1.5.2 Μέθοδοι παλινδρόμησης (Regression)

Οι προσεγγίσεις παλινδρόμησης αντιστοιχούν τις καταστάσεις της επίδειξης σε συνεχείς χώρους δράσης. Παρόμοια με την ταξινόμηση, η είσοδος του παλινδρομητή είναι οι καταστάσεις του ρομπότ και η συνεχής έξοδος είναι οι ενέργειες του ρομπότ. Η παλινδρόμηση, επειδή οι συνεχείς τιμές της εξόδου είναι, συχνά, αποτέλεσμα συνδυασμού πολλαπλών ομάδων επιδείξεων, ουσιαστικά, εφαρμόζεται σε χαμηλού επιπέδου κινήσεις και όχι σε υψηλού επιπέδου συμπεριφορές. Μια σημαντική διαφορά μεταξύ των μεθόδων παλινδρόμησης είναι αν η προσέγγιση της συνάρτησης αντιστοίχησης πραγματοποιείται κατά την εκτέλεση της κίνησης από το ρομπότ ή πριν από αυτή. Παρακάτω, παρουσιάζεται μία σύνοψη των μεθόδων παλινδρόμησης για την εκμάθηση του ρομπότ από την αλληλεπίδραση ανθρώπου-μηχανής για τους δύο παραπάνω τρόπους λειτουργίας, καθώς και για έναν ενδιάμεσο τρόπο αυτών.

Στο ένα άκρο υπάρχει η Οκνηρή Μάθηση (Lazy Learning) [21], όπου η προσέγγιση της συνάρτησης δεν πραγματοποιείται μέχρι να είναι αναγκαία η αντιστοίχηση κάποιου τωρινού σημείου παρατήρησης. Η απλούστερη μέθοδος Οκνηρής Μάθησης είναι η kNN, η οποία εφαρμόζεται, για παράδειγμα, στην επιλογή της δράσης ενός ρομπότ σε ένα λαβύρινθο με μπίλιες [22] και στην προσομοίωση της εύρεσης της τροχιάς μιας μπάλας [23]. Περισσότερο πολύπλοκες προσεγγίσεις περιλαμβάνουν την Τοπικά Σταθμισμένη Παλινδρόμηση (Locally Weighted Regression – LWR) [24]. Μια μέθοδος Τοπικής Σταθμισμένης Παλινδρόμησης συνδέει περαιτέρω τοπικές συναρτήσεις με τη φάση μη γραμμικών ταλαντωτών [25] για την αναπαραγωγή ρυθμικών κινήσεων, και συγκεκριμένα κρούση των ντράμς [26] και περπάτημα [27] με ένα ανθρωπόμορφο ρομπότ. Αν και οι προσεγγίσεις Οκνηρής Εκμάθησης είναι γρήγορες και δεν έχουν μεγάλο κόστος προσέγγισης της συνάρτησης κατά τη διάρκεια της εκτέλεσης της κίνησης σε χώρους που

δεν έχει επισκεφτεί ξανά στο παρελθόν το ρομπότ, απαιτούν την αποθήκευση όλων των δεδομένων εκπαίδευσης.

Στο ενδιάμεσο των δύο άκρων, βρίσκονται τεχνικές στις οποίες τα αρχικά δεδομένα μετατρέπονται σε άλλη, πιθανότατα αραιή, απεικόνιση των δεδομένων πριν την εκτέλεση της κίνησης. Αυτά τα τροποποιημένα δεδομένα στη συνέχεια χρησιμοποιούνται από τεχνικές Οκνηρής Εκμάθησης κατά την εκτέλεση τη κίνησης. Για παράδειγμα, η Σταθμισμένη Παλινδρόμηση Δεκτικού Πεδίου (Receptive Field Weighted Regression – RFWR) [28] πρώτα μετατρέπει τα δεδομένα της επίδειξης σε μία απεικόνιση ενός τοπικού και γραμμικού Γκαουσσιανού μοντέλου. Η Τοπικά Σταθμισμένη Προβολική Παλινδρόμηση (Locally Weighted Projection Regression – LWPR) [29] επεκτείνει αυτή την προσέγγιση για να ανταπεξέρχεται στην διάσταση και στον πλεονασμό των δεδομένων εισόδου. Και η RFWR και η LWPR είναι ικανές στο να ανανεώνουν σταδιακά σε απευθείας σύνδεση τον αριθμό των Γκαουσσιανών αντιπροσώπων καθώς και των παραμέτρων της παλινδρόμησης. Επιτυχημένες ρομποτικές εφαρμογές που χρησιμοποιούν την LWPR περιλαμβάνουν ένα ρομπότ AIBO που επιδεικνύει βασικές ποδοσφαιρικές ικανότητες [30] και ένα ανθρωπόμορφο ρομπότ που παίζει επιτραπέζιο χόκεϋ αέρος (air hockey) [22]. Στην τελευταία αυτή εργασία, στην πραγματικότητα, περιλαμβάνονται πολλές μέθοδοι. Αρχικά χρησιμοποιείται η kNN για την επιλογή της κλάσης συμπεριφοράς και στη συνέχεια η LWPR για την πραγματοποίηση χαμηλού επιπέδου ενεργειών χρησιμοποιώντας τα δεδομένα επίδειξης της συγκεκριμένης κλάσης. Οι προσεγγίσεις αυτές επωφελούνται από το γεγονός ότι δεν χρειάζεται να υπολογίζουν όλα τα δεδομένα της εκπαίδευσης κατά τη διάρκεια εκτέλεσης της κίνησης, εις βάρος, όμως, του επιπλέον κόστους υπολογισμού και γενίκευσης πριν από την εκτέλεση της κίνησης.

Στο εντελώς αντίθετο άκρο βρίσκονται οι προσεγγίσεις οι οποίες σχηματίζουν μια ολοκληρωμένη προσέγγιση της συνάρτησης πριν από την εκτέλεση οποιασδήποτε κίνησης. Έτσι, κατά την εκτέλεση κάποιας κίνησης δεν είναι απαραίτητη η ύπαρξη δεδομένων ή τροποποιημένων απεικονίσεων αυτών. Σε μια δημιουργική εργασία χρησιμοποιείται ένα Νευρωνικό Δίκτυο (Neural Network – NN) για την αυτόνομη οδήγηση ενός μικρού φορτηγού με ταχύτητα σε μια ποικιλία από είδη δρόμων. [31] Επίσης, οι τεχνικές των Νευρωνικών Δικτύων επιτρέπουν σε έναν ρομποτικό βραχίονα να εκτελεί μια κίνηση τοποθέτησης αντικειμένων σε τρύπες [32] και μία ομάδα από b-splines wavelets προσεγγίζουν αλληλουχίες χρονικών δεδομένων για μια κίνηση ενός ολόκληρου ανθρωπόμορφου κορμιού. [33] Ακόμα, είναι πιθανό να χρησιμοποιηθούν στατιστικές προσεγγίσεις που αντιπροσωπεύουν τα δεδομένα της επίδειξης σε μία απλή ή σύνθετη κατανομή η οποία δειγματοληπτείται κατά την εκτέλεση της κίνησης. Τα δεδομένα μιας επίδειξης που κωδικοποιούνται σαν μία κοινή κατανομή των αντικειμένων, της θέσης του χεριού και του προσανατολισμού του χεριού, χρησιμοποιούνται από ένα ανθρωπόμορφο ρομπότ για να πιάσει νέα και γνωστά αντικείμενα που απαντώνται σε σπίτια. [34] Η Παλινδρόμηση Μίξης Γκαουσσιανών (Gaussian Mixture Regression – GMR) μαθαίνει σε ένα ρομπότ διαιτητικά σφυρίγματα καλαθοσφαίρισης [35] και οι Αραιές σε Απευθείας Σύνδεση Γκαουσσιανές Διαδικασίες (Sparse On-line Gaussian Processes – SOGP) μαθαίνουν σε ένα ρομπότ AIBO να εκτελεί βασικές ποδοσφαιρικές κινήσεις [36]. Αυτές οι τεχνικές παλινδρόμησης έχουν όλες το πλεονέκτημα ότι δεν χρειάζεται καθόλου να υπολογίζουν δεδομένα εκπαίδευσης κατά τη διάρκεια εκτέλεσης της κίνησης, αλλά έχουν μεγαλύτερο κόστος υπολογισμού πριν από την εκτέλεση. Επίσης, μερικές από τις τεχνικές, όπως για παράδειγμα, το Νευρωνικό Δίκτυο, έχουν το μειονέκτημα ότι "ξεχνούν" καμιά φορά την αντιστοιχία εισόδων-εξόδων η οποία προσδιορίστηκε προηγουμένως κατά τις εκτελέσεις εκμάθησης, πρόβλημα το οποίο είναι αρκετά σημαντικό όταν η πολιτική του ρομπότ ανανεώνεται σε απευθείας σύνδεση.

Στην παρούσα εργασία χρησιμοποιήθηκε, ως μέθοδος εκμάθησης του ρομπότ στην προσέγγιση και αρπαγή των αντικειμένων, ένα Νευρωνικό Δίκτυο. Έτσι το μεγαλύτερο κόστος υπολογισμού της προσέγγισης της συνάρτησης εισόδου-εξόδου ήταν πριν από την εκτέλεση οποιασδήποτε κίνησης, ενώ κατά τη διάρκεια το κόστος υπολογισμού και ολοκλήρωσης της κίνησης ήταν σχεδόν μηδενικό.

## 1.6 Σύνοψη της παρούσας εργασίας

Στην παρούσα εργασία επικεντρωθήκαμε κυρίως στην αυτοματοποιημένη εκ των προτέρων εκμάθηση του ρομπότ στην προσέγγιση και αρπαγή ορισμένων αντικειμένων καθώς και στη γενίκευση της κίνησης αυτής σε άγνωστα αντικείμενα που βρίσκονταν σε άγνωστες θέσεις στο χώρο. Πιο αναλυτικά:

Στο κεφάλαιο 1, υπάρχει μία εισαγωγή στους τρόπους επικοινωνίας του ανθρώπου με την μηχανή και τον υπολογιστή, στον χειρισμό της μηχανής και κατά συνέπεια και των ρομπότ από τον άνθρωπο, όπως, επίσης, και στην εκμάθησή τους στην πραγματοποίηση ορισμένων συμπεριφορών. Στη συνέχεια του κεφαλαίου, υπάρχει μία ανάλυση των μεθόδων εκμάθησης των ρομπότ από τον άνθρωπο καθώς και παραδείγματα από την βιβλιογραφία για κάθε μέθοδο.

Στο κεφάλαιο 2, υπάρχει μία ανάλυση του τρισδιάστατου χώρου που δημιουργήθηκε για την προσομοίωση του προβλήματος που μας απασχόλησε. Γίνεται περιγραφή της δημιουργίας του περιβάλλοντος εργασίας του ρομπότ, η οποία περιλαμβάνει την δημιουργία μιας στατικής κάμερας και του κατάλληλου φωτισμού και της κατάλληλης σκίασης, ενός τραπεζιού εργασίας, τη δημιουργία των προς εξέταση αντικειμένων και μιας ρομποτικής αρπάγης.

Στο κεφάλαιο 3, υπάρχει η ανάλυση του τρόπου εξαγωγής των αναγκαίων χαρακτηριστικών των αντικειμένων, όπως προκύπτουν από την κάμερα που χρησιμοποιήθηκε για την μετέπειτα αναγνώριση των αντικειμένων και των χαρακτηριστικών τους. Επίσης, υπάρχει και ανάλυση του τρόπου αποθήκευσης των αποτελεσμάτων αυτών καθώς και των δεδομένων από το ρομπότ από το πρόγραμμα της προσομοίωσης.

Στο κεφάλαιο 4, γίνεται η ανάλυση του τρόπου εκμάθησης του ρομπότ με βάση τα δεδομένα επίδειξης από την εκπαίδευση. Υπάρχει, αρχικά, μία ανάλυση στον τρόπο λειτουργίας των Νευρωνικών Δικτύων και στη συνέχεια παρουσιάζεται το Νευρωνικό Δίκτυο που χρησιμοποιήθηκε στην παρούσα εργασία για την εκπαίδευση του ρομπότ.

Στο κεφάλαιο 5 παρατίθενται όλα τα αντικείμενα των συνόλων εκπαίδευσης και ελέγχου καθώς και τα αποτελέσματα που προέκυψαν από την προσομοίωση στον εικονικό περιβάλλον της εκμάθησης του ρομπότ.

Στο κεφάλαιο 6, γίνονται ορισμένα γενικά σχόλια για τα αποτελέσματα καθώς και για τα πλεονεκτήματα και τα μειονεκτήματα ή τους περιορισμούς που υπήρξαν στην συγκεκριμένη επίλυση του προβλήματος. Ακόμα, παρουσιάζονται κάποιες μελλοντικές κατευθύνσεις της έρευνας που μπορεί να υπάρξουν για το συγκεκριμένο πρόβλημα που μας απασχόλησε.

Στο τέλος της εργασίας υπάρχει η βιβλιογραφία που χρησιμοποιήθηκε για την σύνταξη αυτής της διπλωματικής εργασίας καθώς και παραρτήματα τα οποία περιέχουν όλο τον κώδικα που γράφτηκε ή χρησιμοποιήθηκε για την παρούσα εργασία.

ΚΕΦΑΛΑΙΟ 2

# ΔΗΜΙΟΥΡΓΙΑ ΤΡΙΔΙΑΣΤΑΤΟΥ ΕΙΚΟΝΙΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

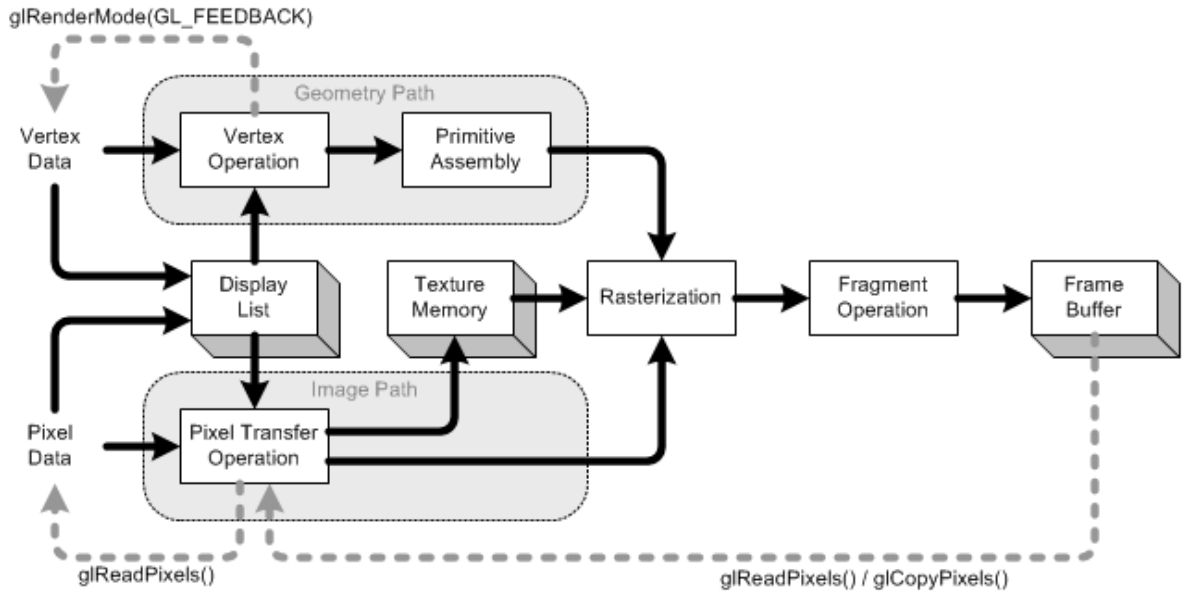
## 2.1 Εισαγωγή

Στην προσπάθεια, αρχικά, να επιλυθεί το πρόβλημα της εκμάθησης του ρομπότ από τον άνθρωπο κρίθηκε, αναγκαία η δημιουργία ενός τριδιάστατου εικονικού περιβάλλοντος, όπου θα γινόντουσαν όλες οι απαραίτητες προσομοιώσεις της πραγματικότητας. Έτσι, λοιπόν, δημιουργήθηκε ένα πρόγραμμα σε αντικειμενοστραφή προγραμματισμό (C++) με χρήση του συστήματος γραφικών της OpenGL.

Το σύστημα γραφικών της OpenGL είναι μια ανοιχτή βιβλιοθήκη γραφικών για την απευθείας εκτέλεση τμημάτων κώδικα από την κάρτα γραφικών και όχι τον επεξεργαστή. Η βιβλιοθήκη αυτή επιτρέπει στο χρήστη να δημιουργήσει προγράμματα για την παραγωγή έγχρωμων εικόνων και κινούμενων διδιάστατων ή τριδιάστατων αντικειμένων. Με την OpenGL, επιτυγχάνεται ο έλεγχος των συστημάτων γραφικής με υπολογιστές, ώστε κανείς να μπορεί να παράγει ρεαλιστικές ή φανταστικές απεικονίσεις της πραγματικότητας που μας περιβάλλει [37].

Το σύστημα απεικόνισης της OpenGL αποτελείται από διακριτές εντολές που χρησιμοποιούνται για να ορίσουν τις λεπτομέρειες των αντικειμένων καθώς και τις απαραίτητες λειτουργίες που παράγουν τα οπτικά εφέ τριδιάστατων ή διδιάστατων εφαρμογών. Το σύστημα αυτό είναι σχεδιασμένο ως μία ανεξάρτητη πλατφόρμα διεπαφής που μπορεί να χρησιμοποιηθεί σε πολλές διαφορετικές πλατφόρμες ανάπτυξης λογισμικών. Για να επιτευχθούν αυτά τα χαρακτηριστικά, δεν περιλαμβάνεται καμία εντολή που να επιτρέπει λειτουργίες διαχείρισης παραθύρων ή επεξεργασία εισόδων από το χρήστη. Στη θέση αυτών, κανείς, είναι υποχρεωμένος να δουλέψει μέσω του λειτουργικού συστήματος που ελέγχει τις λειτουργίες διαχείρισης παραθύρων στη συγκεκριμένη πλατφόρμα που χρησιμοποιεί. Με την ίδια λογική, η OpenGL δε διαθέτει εντολές υψηλού επιπέδου για την περιγραφή μοντέλων τριδιάστατων αντικειμένων που θα επέτρεπαν την εύκολη αναπαραγωγή συγκεκριμένων σχετικά πολύπλοκων σχημάτων. Αντιθέτως, με την OpenGL όλα τα αντικείμενα πρέπει να σχεδιαστούν με βάση ένα μικρό σύνολο γεωμετρικών προτύπων: σημεία, γραμμές, πολύγωνα. Επομένως ο μόνος τρόπος να εισαχθούν πιο πολύπλοκα μοντέλα σχημάτων, είναι με μια βιβλιοθήκη που θα δημιουργηθεί από την OpenGL.

Οι περισσότερες εφαρμογές της OpenGL ακολουθούν την ίδια ροή λειτουργιών, μια σειρά από στάδια επεξεργασίας, τα διαδοχικά στάδια απεικόνισης (OpenGL Rendering Pipeline) τα οποία φαίνονται στο Σχήμα 1.




Σχήμα 1: OpenGL Rendering Pipeline

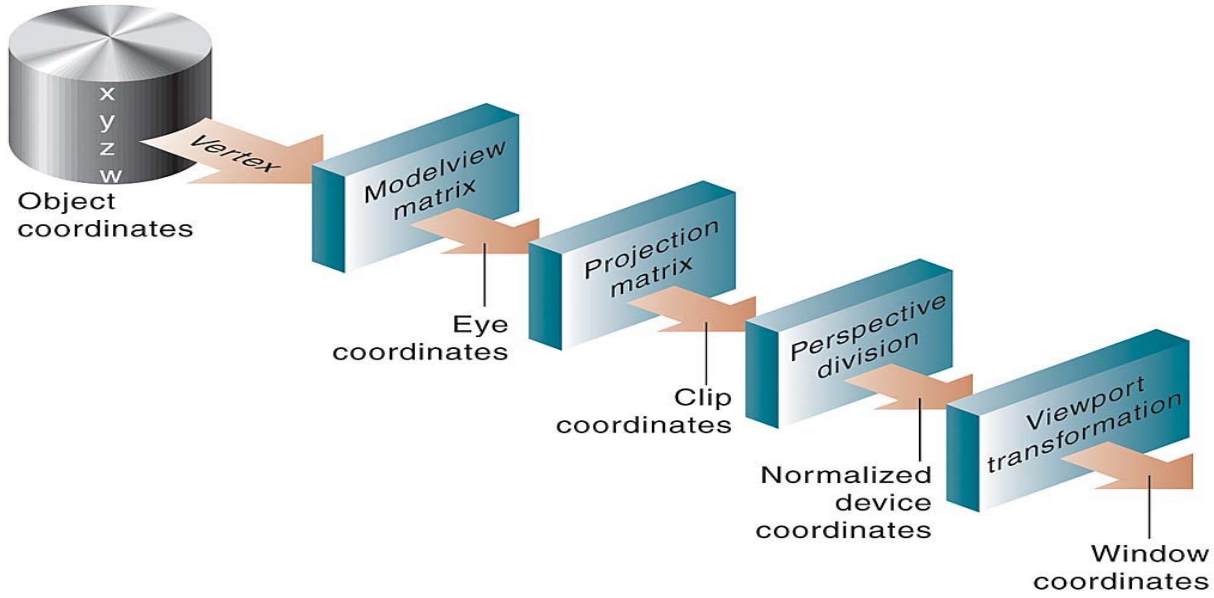
Τα δεδομένα κορυφών (Vertex Data) δηλαδή τα διανύσματα, οι γραμμές και τα πολύγωνα ακολουθούν το γεωμετρικό μονοπάτι (Geometry Path) μέσω των λειτουργιών των κορυφών (Vertex Operation) και των πρωτόγονων συναρμολογήσεων (Primitive Assembly), ενώ τα δεδομένα για τα εικονοστοιχεία (Pixel Data) δηλαδή τα σημεία, οι εικόνες και τα κείμενα ακολουθούν το μονοπάτι εικόνας (Image Path) μέσω της λειτουργίας μεταφοράς εικονοστοιχείων (Pixel Transfer Operation) και της μνήμης υφής (Texture Memory). Ωστόσο, και τα δύο είδη δεδομένων υποβάλλονται σε μια διαδικασία απεικόνισης (Rasterization), όπου τα δεδομένα μετατρέπονται μέσω της λειτουργίας κατάτμησης (Fragment Operation) σε κβάντα εικόνας τα οποία αντιστοιχούν σε εικονοσημεία του πλαισίου μνήμης (Frame Buffer) της κάρτας οθόνης.

Η OpenGL είναι μία μηχανή που έχει εισόδους διάφορες καταστάσεις ή τρόπους. Μόλις ένας τέτοιος τρόπος ή κατάσταση οριστεί, παραμένει εωσότου οριστεί εκ νέου. Τέτοιες καταστάσεις ελέγχουν χαρακτηριστικά όπως το τρέχον χρώμα, τους μετασχηματισμούς απεικόνισης, προβολής, χάραξης γραμμών και πολυγώνων, σχεδίασης, συμπίεσης εικονοστοιχείων της οθόνης, τη θέση και τις ιδιότητες του φωτός καθώς και τα υλικά χαρακτηριστικά των υπό σχεδίαση αντικειμένων. Για όλες τις καταστάσεις αντιστοιχούν δύο εντολές, μία ενεργοποίησης και μία απενεργοποίησης.

Ένα επίπεδο πίσω από τα θέματα διαχείρισης κατάστασης, βρίσκονται οι μετασχηματισμοί που αφορούν το οπτικό πεδίο της απεικόνισης και τον προσανατολισμό του αντικειμένου. Μία σειρά από τρεις υπολογιστικές διαδικασίες μετατρέπουν τις τριδιάστατες συντεταγμένες ενός αντικειμένου σε εικονοστοιχείο στην οθόνη του υπολογιστή. Οι μετασχηματισμοί αυτοί υλοποιούνται με πολλαπλασιασμό πινάκων και περιλαμβάνουν λειτουργίες μοντελοποίησης, εστίασης και προβολής (modeling, viewing, projection transformations). Οι λειτουργίες αυτές περιλαμβάνουν περιστροφή, μεταφορά, διαβάθμιση κλίμακας αξόνων, αντικατοπτρισμό αντικειμένου, ορθογραφική ή προοπτική προβολή. Ένας συνδυασμός μετασχηματισμών συνήθως χρησιμοποιείται κατά τη σχεδίαση της σκηνής.

Εφόσον η σχεδίαση αναπαρίσταται σε ένα ορθογώνιο πλαίσιο-παράθυρο, τα αντικείμενα που εκτείνονται έξω από το παράθυρο πρέπει να αποκοπούν. Το ίδιο συμβαίνει και για τριδιάστατα αντικείμενα για κάθε επίπεδη επιφάνεια ξεχωριστά. Τέλος, πρέπει να οριστεί μια αντιστοιχία μεταξύ των μετασχηματισμένων συντεταγμένων και τα σημεία της οθόνης. Αυτός είναι γνωστός ως μετασχηματισμός παραθύρου (viewport transformation).

Παρατίθεται στο Σχήμα 2 η λογική αλλά όχι η αναγκαία σειρά των μετασχηματισμών:




Σχήμα 2: Μετασχηματισμοί διανυσμάτων

Τους μετασχηματισμούς που χρησιμοποιούμε για να παράγουμε την επιθυμητή σκηνή παρατήρησης μπορούμε να τους φανταστούμε ως τους ανάλογους χειρισμούς ενός φωτογράφου πριν τραβήξει μια φωτογραφία:

- Τοποθέτηση και στόχευση της κάμερας (viewing transformation)
- Σύνθεση της σκηνής: τοποθέτηση και προσανατολισμός αντικειμένων-μοντέλων (modelview matrix transformation)
- Επιλογή φακού (εύρος) και προσαρμογή των διαστάσεων προβολής (zoom) (projection matrix transformation)
- Καθορισμός διαστάσεων εμφάνισης και διαστάσεων παραθύρου (viewport transformation)

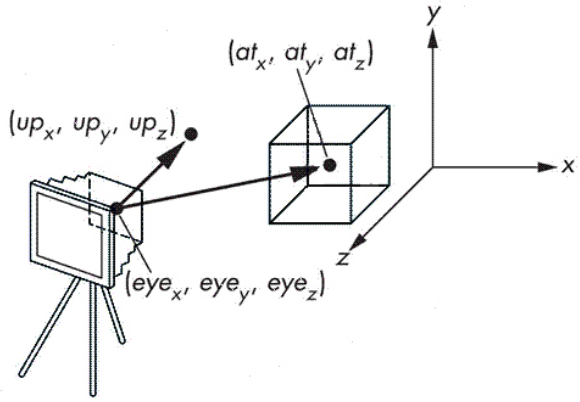
Για να καθοριστούν όλοι οι παραπάνω μετασχηματισμοί, κατασκευάζεται ένας 4 × 4 πίνακας M, ο οποίος πολλαπλασιάζει το κάθε διάνυσμα κατάστασης $v = [x\ y\ z\ w]^T$ που απαρτίζει τη σκηνή για να επιφέρει το ζητούμενο μετασχηματισμό $v' = M \cdot v$.

## 2.2 Δημιουργία στατικής κάμερας

Για την οπτικοποίηση των γραφικών που δημιουργήθηκαν στο πρόγραμμα, αρχικά, ήταν αναγκαία η δημιουργία μιας κάμερας. Στο σημείο αυτό, πρέπει να επισημανθεί ότι θα μπορούσαν να δημιουργηθούν παραπάνω από μία κάμερες έτσι ώστε να υπάρχει στερεοσκοπική αίσθηση του χώρου ή να δημιουργηθεί μία κινούμενη κάμερα η οποία θα βρίσκεται πάνω στο ρομποτικό χειριστή. Ωστόσο, επειδή κύριο μέλημα της εργασίας αυτής δεν είναι η όραση των υπολογιστών και όλα τα προβλήματα που πηγάζουν από αυτόν τον τομέα, δημιουργήθηκε για απλούστευση μία στατική κάμερα που βρίσκεται και παρακολουθεί τον χώρο και δίνει τις πληροφορίες στο ρομποτικό χειριστή όπως αυτή που φαίνεται στο Σχήμα 3. Η υλοποίηση της κάμερας αυτής προκύπτει στην αρχικοποίηση του προγράμματος με την παρακάτω έτοιμη συνάρτηση της βιβλιοθήκης OpenGL.

void gluLookAt (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
GLdouble atX, GLdouble atY, GLdouble atZ,
GLdouble upX, GLdouble upY, GLdouble upZ)

όπου οι παράμετροι eyeX, eyeY, eyeZ ορίζουν την θέση της κάμερας στο χώρο, οι atX, atY, atZ το σημείο αναφοράς όπου θα εστιάζει η κάμερα και οι upX, upY, upZ ορίζουν τη διεύθυνση του διανύσματος της κάμερας που θα δείχνει προς τα πάνω.

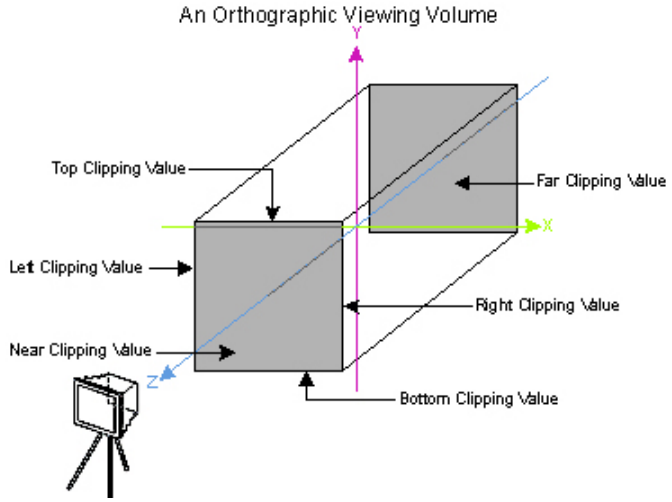



Σχήμα 3: Στατική κάμερα

Στη συνέχεια για τη σχεδίαση της σκηνής χρησιμοποιήθηκε ένα ορθογραφικό πλαίσιο έτσι ώστε να εμφανίζονται τα αντικείμενα που βρίσκονται μόνο μέσα σε αυτό το πλαίσιο όπως φαίνεται και στο Σχήμα 4. Αυτό έγινε χρησιμοποιώντας την συνάρτηση

void glOrtho( GLdouble left clipping value, GLdouble right clipping value,
GLdouble bottom clipping value, GLdouble top clipping value,
GLdouble near clipping value, GLdouble far clipping value );

όπου οι παράμετροι left clipping value και right clipping value ορίζουν τις συντεταγμένες του αριστερού και δεξιού κάθετου επίπεδου αποκοπής, οι bottom clipping value και top clipping value τις συντεταγμένες του κάτω και πάνω οριζόντιου επιπέδου αποκοπής και οι near clipping value και far clipping value τις συντεταγμένες του κοντινότερου και μακρύτερου σε βάθος επιπέδου αποκοπής.




Σχήμα 4: Ορθογραφική προβολή σχεδίασης

## 2.3 Δημιουργία κατάλληλου φωτισμού και σκίασης

Για τη σωστή αναπαράσταση των αντικειμένων στον εικονικό χώρο ήταν αναγκαία και η δημιουργία κατάλληλου φωτισμού και σκίασης. Για τη δημιουργία του φωτισμού είναι απαραίτητος ο ορισμός των ιδιοτήτων των φωτεινών πηγών (Light Properties), του γενικού μοντέλου φωτισμού (Lighting Model) και των επιφανειών των αντικειμένων (Material Properties) ενώ για τη δημιουργία της σκίασης είναι απαραίτητος ο υπολογισμός του ανακλώμενου φωτός σε κάθε κορυφή, ξεχωριστά για κάθε όψη.

Χρησιμοποιώντας μία εντολή μπορούν να οριστούν όλες οι ιδιότητες που αφορούν τον φωτισμό εφαρμόζοντας διαφορετικές παραμέτρους στην εντολή αυτή. Η εντολή της βιβλιοθήκης OpenGL που χρησιμοποιήθηκε στο πρόγραμμα είναι η

void glLightfv(GLenum light, GLenum pname, const GLfloat *params)

όπου η παράμετρος light αναφέρεται στη φωτεινή πηγή (αύξων αριθμός φωτεινής πηγής) έχοντας τη δυνατότητα χρησιμοποίησης μέχρι και εννέα φωτεινών πηγών (GL_LIGHT0 μέχρι GL_LIGHT8), η παράμετρος pname αναφέρεται σε κάποια ιδιότητα της φωτεινής πηγής όπως είναι η θέση, ο τύπος της φωτεινής πηγής καθώς και το χρώμα της ενώ η παράμετρος *params ορίζει έναν δείκτη σε μία τιμή της αντίστοιχης παραμέτρου pname.

Στον Πίνακα 1 παρατίθενται όλες οι παράμετροι, οι προεπιλεγμένες τιμές τους και η αντίστοιχη λειτουργία για κάθε ιδιότητα του γενικού μοντέλου φωτισμού.

| **Όνομα παραμέτρου** | **Προεπιλεγμένες τιμές** | **Λειτουργία** |
|---|---|---|
| GL_AMBIENT | (0.0, 0.0, 0.0, 1.0) | Ένταση φωτισμού RGBA περιβάλλοντος |
| GL_DIFFUSE | (1.0, 1.0, 1.0, 1.0) | Ένταση φωτισμού RGBA διάχυσης |
| GL_SPECULAR | (1.0, 1.0, 1.0, 1.0) | Ένταση φωτισμού RGBA ανάκλασης |
| GL_POSITION | (0.0, 0.0, 1.0, 0.0) | Συντεταγμένες πηγής φωτισμού (x, y, z, w) |
| GL_SPOT_DIRECTION | (0.0, 0.0, -1.0) | Διεύθυνση εστίασης πηγής φωτισμού (x, y, z) |
| GL_SPOT_EXPONENT | 0.0 | Συντελεστής εστίασης φωτεινής πηγής |
| GL_SPOT_CUTOFF | 180.0 | Γωνία αποκοπής φωτεινής πηγής |
| GL_CONSTANT_ATTENUATION | 1.0 | Σταθερός παράγοντας μείωσης |
| GL_LINEAR_ATTENUATION | 0.0 | Γραμμικός παράγοντας μείωσης |
| GL_QUADRATIC_ATTENUATION | 0.0 | Τετραγωνικός παράγοντας μείωσης |

Πίνακας 1: Πίνακας ιδιοτήτων φωτεινών πηγών

Στην OpenGL η έννοια του γενικού μοντέλου φωτισμού απαρτίζεται από τρείς παραμέτρους:

- Την γενική ένταση της φωτεινότητας του περιβάλλοντος.
- Τον τρόπο υπολογισμού των γωνιών ανάκλασης ο οποίος εξαρτάται από τη διεύθυνση παρατήρησης που μπορεί να είναι τοπική ή σε άπειρη απόσταση από την φωτεινή πηγή.
- Τον τρόπο υπολογισμού της φωτεινότητας των αντικειμένων ο οποίος ενδέχεται να είναι διαφορετικός για την μπροστινή όψη και διαφορετικός για την πίσω όψη.

Η συνάρτηση της βιβλιοθήκης OpenGL με την οποία ορίζονται οι ιδιότητες του γενκού μοντέλου φωτισμού είναι η

void glLightModelfv(GLenum pname, TYPE *param);

όπου η παράμετρος pname αναφέρεται σε κάποια ιδιότητα του μοντέλου φωτισμού ενώ η παράμετρος *param ορίζει έναν δείκτη στην τιμή της αντίστοιχης ιδιότητας.

Στον Πίνακα 2 παρατίθενται όλες οι παράμετροι, οι προεπιλεγμένες τιμές τους και η αντίστοιχη λειτουργία για κάθε ιδιότητα του γενικού μοντέλου φωτισμού.

| **Όνομα παραμέτρου** | **Προεπιλεγμένες τιμές** | **Λειτουργία** |
|---|---|---|
| GL_LIGHT_MODEL_AMBIENT | (0.2, 0.2, 0.2, 1.0) | Γενική ένταση RGBA της φωτεινότητας ολόκληρης της σκηνής |
| GL_LIGHT_MODEL_LOCAL_VIEWER | 0.0 or GL_FALSE | Τρόπος υπολογισμού των γωνιών ανάκλασης |
| GL_LIGHT_MODEL_TWO_SIDE | 0.0 or GL_FALSE | Τρόπος υπολογισμού φωτεινότητας με χρήση μονής ή διπλής όψης αντικειμένων |

Πίνακας 2: Πίνακας ιδιοτήτων γενικού μοντέλου φωτισμού

Ακόμα, για τη δημιουργία του φωτισμού είναι αναγκαίος και ο ορισμός των ιδιοτήτων των επιφανειών των αντικειμένων. Η ήδη υλοποιημένη συνάρτηση της βιβλιοθήκης OpenGL που χρησιμοποιήθηκε είναι η

void glMaterialfv(GLenum face, GLenum pname, TYPE *param);

όπου η παράμετρος face μπορεί να είναι GL_FRONT, GL_BACK ή GL_FRONT_AND_BACK και προσδιορίζει ποια ή ποιες όψεις θα έχουν τα χαρακτηριστικά που θα οριστούν στη συνέχεια, η παράμετρος pname αναφέρεται σε κάποια ιδιότητα του μοντέλου φωτισμού ενώ η παράμετρος *param ορίζει έναν δείκτη στην τιμή της αντίστοιχης ιδιότητας.

Στον Πίνακα 3 παρατίθενται όλες οι παράμετροι, οι προεπιλεγμένες τιμές τους και η αντίστοιχη λειτουργία για κάθε ιδιότητα των επιφανειών των αντικειμένων.

| **Όνομα παραμέτρου** | **Προεπιλεγμένες τιμές** | **Λειτουργία** |
|---|---|---|
| GL_AMBIENT | (0.2, 0.2, 0.2, 1.0) | Χρώμα περιβάλλοντος υλικού |
| GL_DIFFUSE | (0.8, 0.8, 0.8, 1.0) | Χρώμα διάχυσης υλικού |
| GL_AMBIENT_AND_DIFFUSE | | Χρώμα περιβάλλοντος και διάχυσης υλικού |
| GL_SPECULAR | (0.0, 0.0, 0.0, 1.0) | Χρώμα ανάκλασης υλικού |
| GL_SHININESS | 0.0 | Εκθετικός συντελεστής ανάκλασης |
| GL_EMISSION | (0.0, 0.0, 0.0, 1.0) | Χρώμα εκπομπής υλικού |
| GL_COLOR_INDEXES | (0,1,1) | Δείκτες χρώματος περιβάλλοντος, διάχυσης και ανάκλασης |

Πίνακας 3: Πίνακας ιδιοτήτων των επιφανειών των αντικειμένων

Για τη σωστή αναπαράσταση των αντικειμένων, παρά τη δημιουργία κατάλληλου φωτισμού, είναι αναγκαία και η δημιουργία κατάλληλης σκίασης. Η σκίαση δημιουργείται με την παρακάτω συνάρτηση της OpenGL.

void glShadeModel(GLenum mode);

όπου η παράμετρος mode μπορεί να είναι είτε GL_FLAT είτε GL_SMOOTH.

Κατά την επίπεδη σκίαση (Flat Shading) επιλέγεται το υπολογισθέν χρώμα μιας κορυφής του αντικειμένου και το χρώμα αυτό εφαρμόζεται σε όλα τα εικονοστοιχεία που έχουν δημιουργηθεί κατά την διαδικασία της απεικόνισης. Κατά την ομαλή σκίαση (Smooth Shading), η οποία είναι και η προεπιλεγμένη σκίαση, τα υπολογισθέντα χρώματα των κορυφών ενός αντικειμένου παρεμβάλλονται μεταξύ τους κατά την διαδικασία της απεικόνισης εφαρμόζοντας τελικά διαφορετικά χρώματα σε κάθε τελικό εικονοστοιχείο.

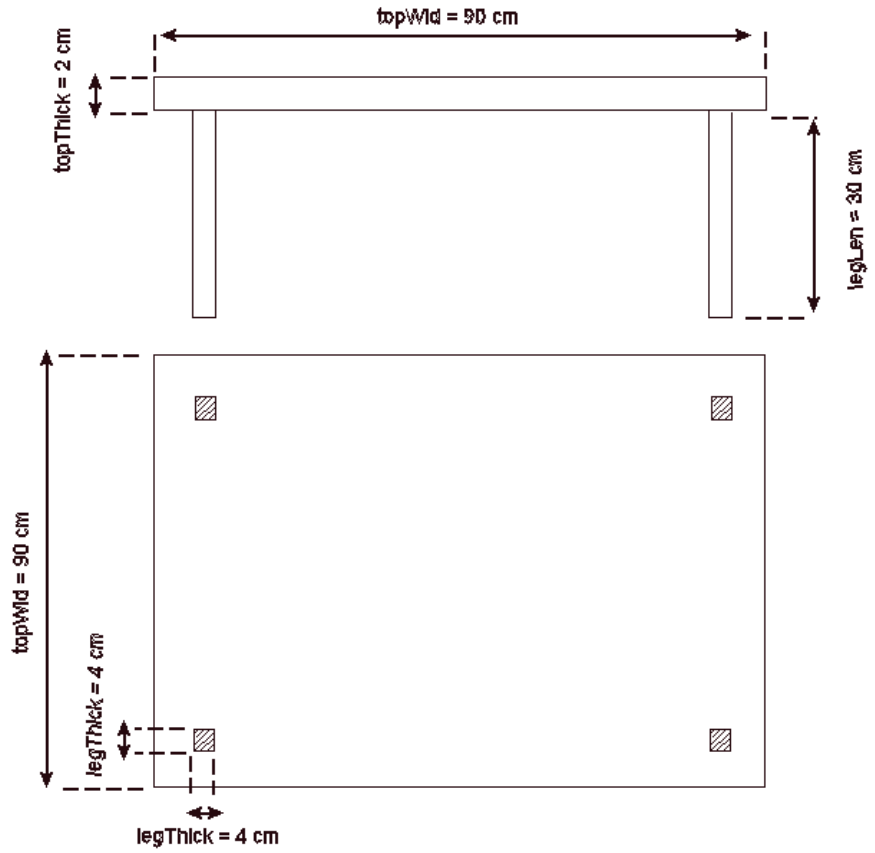
## 2.4 Δημιουργία τραπεζιού ρομποτικής εργασίας

Στο εικονικό περιβάλλον που δημιουργήθηκε, για την τοποθέτηση των αντικειμένων, δημιουργήθηκε και ένα περιβάλλον εργασίας. Το περιβάλλον αυτό αποτελείται από ένα τραπέζι εργασίας το οποίο υλοποιήθηκε με δύο συναρτήσεις με χρήση του αντικειμενοστραφή προγραμματισμού και της βιβλιοθήκης OpenGL. Οι συναρτήσεις αυτές αφορούν τη δημιουργία του πάνω μέρους ενός τραπεζιού και των τεσσάρων ποδιών του τραπεζιού. Οι συναρτήσεις που υλοποιήθηκαν καθώς και τα ορίσματά τους περιγράφονται παρακάτω.

```
void table(double topWid, double topThick, double legThick, double legLen)
void tableLeg(double legThick, double legLen)
```

όπου η παράμετρος topWid είναι το μήκος και το πλάτος και η topThick το πάχος του πάνω μέρος του τραπεζιού και η legThick το πάχος και το πλάτος και legLen το μήκος των τεσσάρων ποδιών του τραπεζιού.

Οι τιμές των μηκών είναι αυθαίρετες και αυτές που επιλέχθηκαν στη συγκεκριμένη εργασία φαίνονται στο
Σχήμα 5.




Σχήμα 5: Τραπέζι εργασίας εικονικού περιβάλλοντος

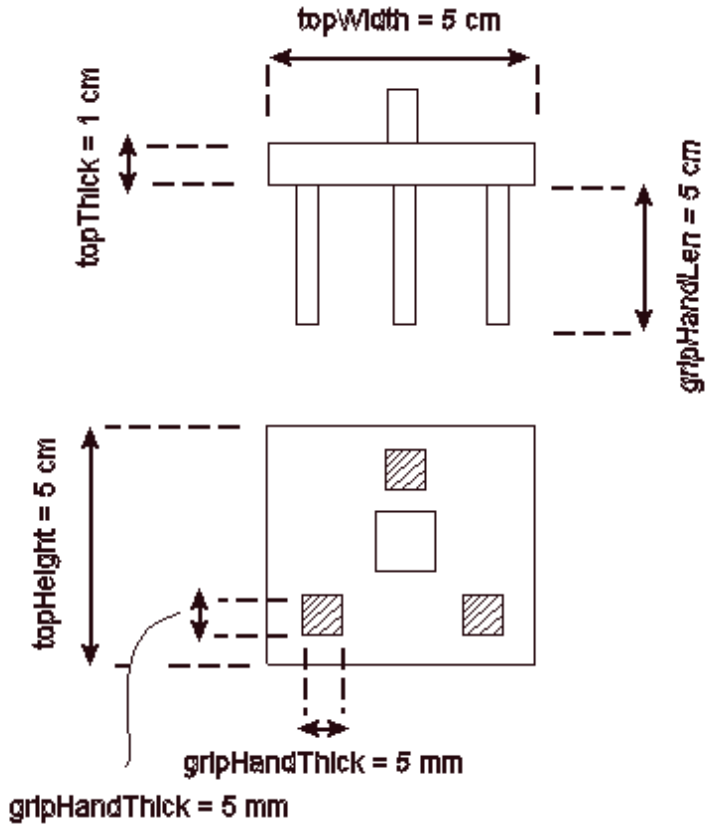
## 2.5 Δημιουργία ρομποτικής αρπάγης

Στο εικονικό περιβάλλον που δημιουργήθηκε έπρεπε να δημιουργηθεί και το ρομπότ στο οποίο θα γίνει η εκμάθηση. Ωστόσο, χωρίς βλάβη της γενικότητας και για να αποφευχθούν τυχόν προβλήματα και για καλύτερη οπτικοποίηση του εικονικού περιβάλλοντος, δημιουργήθηκε μόνο η ρομποτική αρπάγη και όχι το ολόκληρο το ρομπότ με τους βραχίονές του. Υλοποιήθηκαν δύο συναρτήσεις στον αντικειμενοστραφή προγραμματισμό με χρήση της βιβλιοθήκης OpenGL, μία για την δημιουργία του ρομποτικού χεριού και μία για τη δημιουργία τριών ρομποτικών δακτύλων. Οι συναρτήσεις αυτές καθώς και τα ορίσματά τους περιγράφονται παρακάτω.

```
void gripper (double topWidth, double topThick, double topHeight,
                    double gripHandThick, double gripHandLen)
void gripperHand(double gripHandThick, double gripHandLen)
```

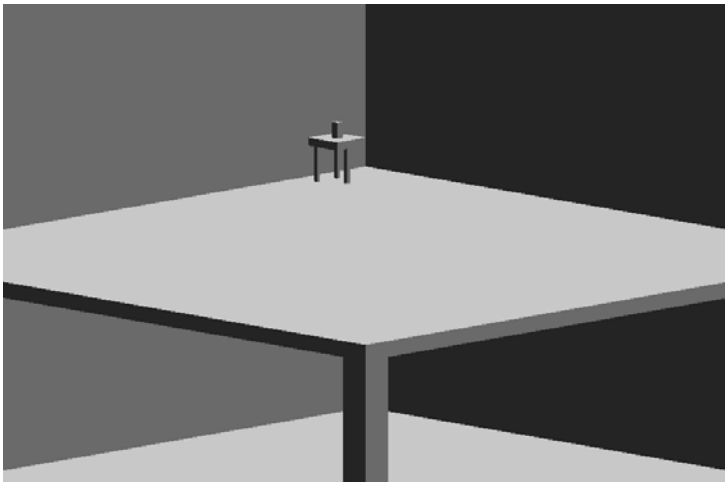
όπου οι παράμετροι topWidth, topThick, topHeight προσδιορίζουν το μήκος, το πάχος και το πλάτος αντίστοιχα του ρομποτικού χεριού και η παράμετρος gripHandThick το πλάτος και πάχος και η gripHandLen το μήκος των τριών ρομποτικών δακτύλων.

Οι τιμές των μηκών είναι αυθαίρετες και αυτές που επιλέχθηκαν στη συγκεκριμένη εργασία φαίνονται στο Σχήμα 6.




Σχήμα 6: Ρομποτική αρπαγή εικονικού περιβάλλοντος

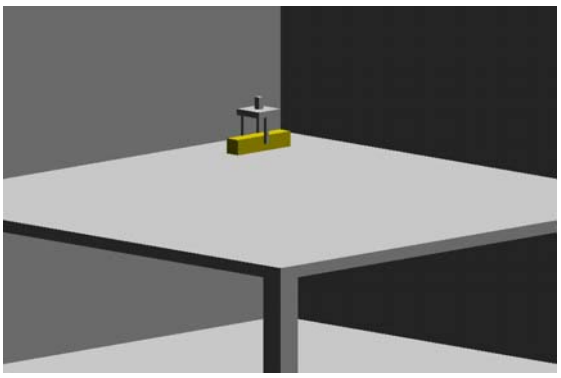
Στο Σχήμα 7 φαίνεται το εικονικό περιβάλλον που δημιουργήθηκε, η ρομποτική αρπάγη καθώς και η οπτική γωνία από την οποία ελήφθησαν οι εικόνες των αντικειμένων για την μετέπειτα επεξεργασία τους.



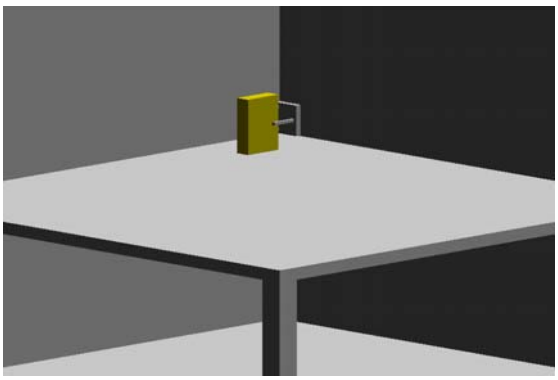
**Σχήμα 7: Εικονικό περιβάλλον με τη ρομποτική αρπάγη**

## 2.6 Δημιουργία χειριζόμενων αντικειμένων

Τέλος, δημιουργήθηκαν και τα αντικείμενα στα οποία έγινε, μετέπειτα, η εκμάθηση του ρομποτικού χειριστή. Τα αντικείμενα αυτά είναι ορθογώνια παραλληλεπίπεδα με διάφορα ύψη και μήκη στον έναν από τους άξονές τους επειδή αν αυξανόταν και ο τρίτος άξονας του παραλληλεπιπέδου τότε θα ήταν αδύνατη η αρπαγή του από τον ρομποτικό χειριστή. Πιο συγκεκριμένα, το μήκος του αντικειμένου στον x άξονα θεωρήθηκε σταθερό και ίσο με $x = 0.025m$. Επίσης, θεωρήθηκε ότι τα αντικείμενα με ύψος μικρότερο του 0.09m και τα αντικείμενα των οποίων το μήκος του άξονα z είναι μεγαλύτερο από το ύψος τους, η ρομποτική αρπάγη θα τα προσεγγίζει χωρίς γωνία κύλισης, ενώ τα αντικείμενα με ύψος μεγαλύτερο ή ίσο με 0.09 και με μήκος στον άξονα z μικρότερο από το ύψος τους θα τα προσεγγίζει με γωνία κύλισης $90^o$. Στο Σχήμα 8 και στο Σχήμα 9 παρουσιάζονται ενδεικτικά δύο από τα αντικείμενα που δημιουργήθηκαν. Ολόκληρο το σύνολο των αντικειμένων που δημιουργήθηκαν παρουσιάζεται σε επόμενο κεφάλαιο ως το σύνολο εκπαίδευσης του ρομποτικού χειριστή.



**Σχήμα 8: Χειριζόμενο αντικείμενο**



**Σχήμα 9: Χειριζόμενο αντικείμενο**

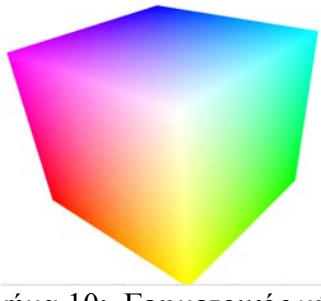
# ΕΞΑΓΩΓΗ ΟΠΤΙΚΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ

## 3.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται αρχικά μια παρουσίαση των δύο συστημάτων απεικόνισης χρωμάτων που χρησιμοποιήθηκαν στη συγκεκριμένη εργασία, του συστήματος RGB (Red Green Blue – Κόκκινο Πράσινο Μπλέ) και του συστήματος HSL (Hue Saturation Lightning – Απόχρωση Κορεσμός Φωτεινότητα) καθώς και των σχέσεων μετατροπής του συστήματος RGB στο σύστημα HSL. Στη συνέχεια επεξηγείται ο τρόπος κατάτμησης της εικόνας που λαμβάνει το ρομπότ με βάση την απόχρωση του αντικειμένου καθώς και όλα τα χαρακτηριστικά για την περιοχή του αντικειμένου που μπορούν να ληφθούν με υλοποιημένες συναρτήσεις του περιβάλλοντος Matlab. Τέλος παρουσιάζονται τα συγκεκριμένα οπτικά χαρακτηριστικά (οπτικοί περιγραφείς σχήματος – shape descriptors) που χρησιμοποιήθηκαν στην εργασία και η αποθήκευση τόσο των χαρακτηριστικών αυτών όσο και των δεδομένων που αφορούν την ρομποτική αρπάγη και προέρχονται από το ίδιο το ρομπότ.

## 3.2 Μετατροπή εικόνων από το σύστημα RGB στο σύστημα HSL

Ο πιο διαδεδομένος χρωματικός χώρος που χρησιμοποιείται, σχεδόν, στο σύνολο των συσκευών που έχουν οθόνη είναι ο χώρος RGB (Red Green Blue). Ο χώρος αυτός αποτελείται από τρείς διαστάσεις που αντιπροσωπεύουν την περιεκτικότητα σε κόκκινο (Red), σε πράσινο (Green) και σε μπλε (Blue). Η γεωμετρική αναπαράσταση του RGB, όπως φαίνεται και στο διπλανό Σχήμα 10 αντιστοιχεί σε έναν κύβο, όπου οι οκτώ ακμές του είναι μόνο κόκκινο, μόνο μπλε, μόνο πράσινο, όλα τα χρώματα μαζί δίνοντας λευκό, κανένα χρώμα δίνοντας μαύρο, κόκκινο και πράσινο μαζί δίνοντας κίτρινο, κόκκινο και μπλε μαζί δίνοντας μωβ και μπλε και πράσινο μαζί δίνοντας το χρώμα κυανό.



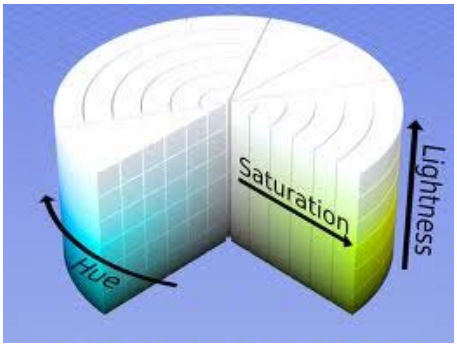
Σχήμα 10: Γεωμετρικός χώρος συστήματος RGB

Γενικά το σύστημα RGB αποτελεί το πιο διαδεδομένο μοντέλο στην αναπαράσταση των εικόνων. Τα βίντεο παρουσιάζουν τις έγχρωμες εικόνες διαμορφώνοντας την ένταση των τριών βασικών χρωμάτων σε κάθε εικονοστοιχείο. Το σύστημα RGB είναι βολικό για την χρωματική αναπαράσταση αλλά όχι και για την κατάτμηση και ανάλυση των έγχρωμων

αναπαραστάσεων εικόνων, εξαιτίας της υψηλής συσχέτισης ανάμεσα στα τρία χρώματα του RGB. Υπό υψηλή συσχέτιση, δηλαδή όταν η ένταση αλλάξει, τότε και τα τρία χρώματα θα αλλάξουν. Επίσης, η μέτρηση ενός χρώματος στον χώρο RGB δεν αναπαριστά διαφορές χρωμάτων σε μια ομοιόμορφη κλίμακα και ως εκ τούτου είναι δύσκολο να εκτιμηθεί η ομοιότητα δυο χρωμάτων από την απόσταση τους στον RGB χώρο.

Η RGB εικόνα μπορεί να θεωρηθεί ως μια στοίβα τριών γκρίζων εικόνων που όταν τοποθετηθούν στις εισόδους του κόκκινου, πράσινου και μπλε μιας έγχρωμης οθόνης θα αναπαράγουν την αρχική έγχρωμη εικόνα. Η κλάση κάθε συνιστώσας καθορίζει το εύρος των τιμών των στοιχείων. Δηλαδή, εάν η RGB εικόνα είναι κλάσης διπλής ακρίβειας (double) τότε το εύρος τιμών είναι [0,1]. Ομοίως, εάν το εύρος είναι [0,255] ή [0,65535] τότε οι RGB εικόνες είναι κλάσης uint8 ή uint16 αντίστοιχα. Όπως είναι γνωστό το πλήθος των bits που χρησιμοποιούνται για την τιμή του κάθε μικροστοιχείου καθορίζει το «βάθος» της εικόνας. Για παράδειγμα εάν κάθε συνιστώσα εικόνας είναι 8bit τότε η συνολική εικόνα RGB είναι 24bit. Ο δυνατός αριθμός των χρωμάτων σε μια RGB εικόνα δίνεται από τον τύπο $3 \cdot 2^b$ οπού b ο αριθμός των bits που χρησιμοποιούνται.

Το σύστημα RGB είναι εύκολο στην αναπαράσταση σε οθόνη, η επεξεργασία του όμως για μηχανική όραση είναι δύσκολη αφού η αλλαγή στην απόχρωση, αλλάζει τιμή και στους 3 άξονες. Για το λόγο αυτό είναι αναγκαία η μετατροπή της εικόνας σε ένα άλλο σύστημα χρωματισμού. Το νέο σύστημα που επιλέχθηκε είναι το HSL Ο χώρος του συστήματος HSL όπως φαίνεται και στο διπλανό Σχήμα 11 είναι κυλινδρικός. Σε κάθε κύλινδρο, η γωνία γύρω από τον κεντρικό άξονα αντιπροσωπεύει το χρώμα (Hue), η απόσταση από τον κεντρικό άξονα αντιπροσωπεύει τον κορεσμό (Saturation) και η απόσταση πάνω στον κεντρικό άξονα αντιπροσωπεύει την ένταση της φωτεινότητας (Lightning).



Σχήμα 11: Γεωμετρικός χώρος συστήματος HSL

Το σύστημα αυτό διαχωρίζει την πληροφορία του χρώματος μιας εικόνας από την ένταση του. Η πληροφορία του χρώματος αναπαρίσταται από τις τιμές της απόχρωσης και του κορεσμού, ενώ η ένταση η οποία περιγράφει τη φωτεινότητα μιας εικόνας καθορίζεται από τη συνολική ποσότητα του φωτός. Η απόχρωση αναπαριστά βασικά χρώματα και καθορίζεται από το επικρατέστερο μήκος κύματος στη φασματική διάδοση των φωτεινών μηκών κύματος. Ο κορεσμός αποτελεί ένα μέτρο της καθαρότητας του χρώματος και δείχνει το σύνολο του λευκού φωτός που έχει προστεθεί στην απόχρωση.

Γενικά, η απόχρωση θεωρείται ως μια γωνία μεταξύ μιας γραμμής αναφοράς και του σημείου-χρώματος πάνω στον RGB χώρο. Οι τιμές της απόχρωσης κυμαίνονται από 0 έως 360 μοίρες. Για παράδειγμα στο μπλε αντιστοιχούν 240 μοίρες, στο βαθύ κόκκινο 300, στο κίτρινο 60 και στο πράσινο 120. Η συνιστώσα του κορεσμού αναπαριστά την ακτινική απόσταση από το κέντρο του κυλίνδρου. Όσο το σημείο βρίσκεται πιο κοντά στο κέντρο τόσο πιο λαμπερό είναι το χρώμα. Η ένταση εκφράζεται από το ύψος του άξονα που περνάει από το κέντρο του κυλίνδρου .Ο άξονας του κυλίνδρου περιγράφει τα επίπεδα του γκρι. Για παράδειγμα η μηδενική ένταση αντιστοιχεί στο μαύρο, ενώ η μέγιστη ένταση αντιστοιχεί στο λευκό. Κάθε επιφάνεια του κυλίνδρου κάθετη στον άξονα της έντασης, αποτελεί ένα επίπεδο με την ίδια ένταση. Το συγκεκριμένο σύστημα έχει μια αναπτυγμένη ικανότητα αναπαράστασης των χρωμάτων, που μπορεί να αντιληφθεί ένας άνθρωπος διότι το οπτικό του σύστημα είναι σε θέση να διακρίνει διαφορετικές αποχρώσεις. Έτσι, η

αντίληψη διαφορετικής έντασης ή κορεσμού δεν υποδηλώνει την αναγνώριση διαφορετικών χρωμάτων.

Οι συντεταγμένες του HSL συστήματος μπορούν να μεταφερθούν από τον RGB χώρο. Οι τύποι για την μετατροπή της απόχρωσης, του κορεσμού και της έντασης δίνονται παρακάτω [39].

Ορίζουμε ως μέγιστη και ως ελάχιστη τιμή από τις τιμές του χώρου RGB τις μεταβλητές M και m καθώς και τη διαφορά τους ως τη μεταβλητή C.

$$M = \max(R, G, B)$$
$$m = \min(R, G, B)$$
$$C = M - m$$

Σύμφωνα με τα παραπάνω η τιμή για την απόχρωση H θα δίνεται από τον τύπο:

$$H_{HSL} = 60^o \cdot H' \text{ όπου } H' = \begin{cases} απροσδιόριστο, & εάν\ C = 0 \\ \dfrac{G-B}{C} \bmod 6 & , εάν\ M = R \\ \dfrac{B-R}{C} + 2 & , εάν\ M = G \\ \dfrac{R-G}{C} + 4 & , εάν\ M = B \end{cases}$$

Η τιμή της έντασης της φωτεινότητας δίνεται από τον τύπο:

$$L_{HSL} = \frac{1}{2}(M + m)$$

Η τιμή του κορεσμού δίνεται από τον τύπο:

$$S_{HSL} = \begin{cases} 0 & , εάν\ C = 0 \\ \dfrac{C}{2L} & , εάν\ L \le \dfrac{1}{2} \\ \dfrac{C}{2-2L} & , εάν\ L > \dfrac{1}{2} \end{cases}$$

## 3.3 Κατάτμηση εικόνας με βάση την απόχρωση του αντικειμένου

Η μέθοδος κατάτμησης χρησιμοποιεί την απόχρωση του αντικειμένου. Πιο συγκεκριμένα, θεωρήθηκε κατά τη διάρκεια δημιουργίας του εικονικού χώρου και των αντικειμένων ότι αυτά θα έχουν συγκεκριμένη απόχρωση. Για το λόγο αυτό χρησιμοποιήθηκε η μετατροπή της εικόνας από το χώρο RGB στο χώρο HSL όπως περιγράφτηκε στο προηγούμενο υποκεφάλαιο. Στη συνέχεια στο πρόγραμμα του Matlab

που δημιουργήθηκε, έγινε η απομόνωση του αντικειμένου χρησιμοποιώντας τη γνωστή απόχρωση του αντικειμένου από το εικονικό περιβάλλον και έπειτα μετατράπηκε η εικόνα σε ασπρόμαυρη. Αυτό έγινε θέτοντας σε κάθε εικονοστοιχείο της εικόνας την τιμή μηδέν (μαύρο χρώμα) στην περίπτωση που το εικονοστοιχείο αυτό αποτελούσε εικονοστοιχείο του αντικειμένου και ένα (λευκό χρώμα) σε όλα τα υπόλοιπα που δεν αποτελούσαν το αντικείμενο.

Η τμηματοποιημένη εικόνα μπορεί να περιέχει έναν αριθμό από περιοχές που εσφαλμένα έχουν ανιχνευθεί ως περιοχές του αντικειμένου, λόγω διακυμάνσεων στο φωτισμό ή λόγω εμφάνισης άλλων περιοχών στην αρχική εικόνα με την ίδια απόχρωση του αντικειμένου. Για το λόγο αυτό κρατήθηκε μόνο η περιοχή με το μεγαλύτερο εμβαδόν που θεωρείται ότι είναι το προς εξέταση αντικείμενο.

## 3.4 Υπολογισμός χαρακτηριστικών αντικειμένου

Στη συνέχεια του προγράμματος στο Matlab έγινε η εξαγωγή των απαραίτητων πληροφοριών από την περιοχή της εικόνας που ανιχνεύθηκε ότι είναι το αντικείμενο. Αυτό έγινε με χρήση μίας ήδη υλοποιημένης συνάρτησης (regionprops) του Matlab.

STATS = regionprops (region , property)

όπου η παράμετρος region αναφέρεται στην περιοχή για την οποία θέλουμε να εξάγουμε χαρακτηριστικά και property μια ιδιότητα της περιοχής που θέλουμε να εξάγουμε.

Στη θέση του property μπορούν να τοποθετηθούν οι παρακάτω παράμετροι σύμφωνα με το εγχειρίδιο χρήσης του Matlab.

| **Όνομα παραμέτρου** | **Λειτουργία ιδιότητας** |
|---|---|
| 'Area' | Εμβαδόν της περιοχής |
| 'BoundingBox' | Το μικρότερο δυνατόν τετράγωνο που περιέχει την περιοχή |
| 'Centroid' | Κέντρο μάζας της περιοχής |
| 'ConvexArea' | Εμβαδόν του μικρότερου δυνατού πολυγώνου περιγράμματος της περιοχής |
| 'ConvexHull' | Συντεταγμένες του μικρότερου δυνατού πολυγώνου περιγράμματος της περιοχής |
| 'ConvexImage' | Δυαδική εικόνα του περιγράμματος του μικρότερου δυνατού πολυγώνου της περιοχής με όλα τα εσωτερικά εικονοστοιχεία να έχουν χρώμα λευκό ενώ τα υπόλοιπα μαύρο |

| | |
|---|---|
| 'Eccentricity' | Εκκεντρότητα της περιοχής |
| 'EquivDiameter' | Αντίστοιχη διάμετρος ενός κύκλου με το ίδιο εμβαδόν με την περιοχή |
| 'EulerNumber' | Αριθμός των αντικειμένων της περιοχής μείον τις τρύπες του κάθε αντικειμένου της περιοχής αυτής |
| 'Extent' | Αναλογία των εικονοστοιχείων της περιοχής σε σχέση με τα συνολικά εικονοστοιχεία από το μικρότερο δυνατόν τετράγωνο που περιέχει την περιοχή αυτή |
| 'Extrema' | Ακραία σημεία της περιοχής |
| 'FilledArea' | Αριθμός των εικονοστοιχείων της περιοχής με όλες τις τρύπες της καλυμμένες |
| 'FilledImage' | Την ίδια περιοχή αλλά με όλες τις τρύπες της καλυμμένες |
| 'Image' | Δυαδική εικόνα της περιοχής με όλα τα εσωτερικά εικονοστοιχεία να έχουν χρώμα λευκό ενώ τα υπόλοιπα μαύρο |
| 'MazorAxisLength' | Μήκος του μεγαλύτερου άξονα της περιοχής σε εικονοστοιχεία |
| 'MinorAxisLength' | Μήκος του μικρότερου άξονα της περιοχής σε εικονοστοιχεία |
| 'Orientation' | Τη γωνία σε μοίρες του μεγαλύτερου άξονα της περιοχής ως προς τον οριζόντιο άξονα |
| 'Perimeter' | Μήκος περιμέτρου της περιοχής σε εικονοστοιχεία |
| 'PixelIdxList' | Λίστα που περιέχει τους γραμμικούς δείκτες των εικονοστοιχείων της περιοχής |
| 'PixelList' | Λίστα που περιέχει τις συντεταγμένες των εικονοστοιχείων της περιοχής |
| 'Solidity' | Αναλογία των εικονοστοιχείων του περιγράμματος της περιοχής που ανήκουν και στην ίδια την περιοχή. Προκύπτει από τον λόγο 'Area'/'ConvexArea' |

Πίνακας 4: Πίνακας παραμέτρων και αντίστοιχων ιδιοτήτων της συνάρτησης regionprops

### 3.4.1 Κέντρο μάζας αντικειμένου

Χρησιμοποιώντας την ιδιότητα 'Centroid' υπολογίζεται το κέντρο μάζας του αντικειμένου. Το αποτέλεσμα είναι δύο αριθμοί που υποδεικνύουν τις συντεταγμένες του κέντρου μάζας, την τετμημένη και την τεταγμένη, του αντικειμένου σε εικονοστοιχεία.

### 3.4.2 Μήκος πρωτεύοντος και δευτερεύοντος άξονα του αντικειμένου

Χρησιμοποιώντας τις ιδιότητες 'MazorAxisLength' και 'MinorAxisLength' υπολογίζεται το μήκος του πρωτεύοντος και δευτερεύοντος άξονα του αντικειμένου. Το αποτέλεσμα από την κλήση της συνάρτησης του Matlab είναι ένας αριθμός για κάθε ιδιότητα που υποδεικνύει το μήκος του αντίστοιχου άξονα σε εικονοστοιχεία. Στην συνέχεια υπολογίζεται και ο λόγος των μηκών των δύο αξόνων του αντικειμένου ο οποίος προκύπτει από τη διαίρεση του μήκους του πρωτεύοντος προς το μήκος του δευτερεύοντος άξονα.

### 3.4.3 Προσανατολισμός αντικειμένου

Χρησιμοποιώντας την ιδιότητα 'Orientation' υπολογίζεται ο προσανατολισμός της περιοχής που έχει αναγνωριστεί ως αντικείμενο. Το αποτέλεσμα της συνάρτησης του Matlab είναι ένας αριθμός που υποδεικνύει την γωνία του κεντρικού άξονα σε μοίρες ως προς τον οριζόντιο άξονα.

### 3.4.4 Εκκεντρότητα αντικειμένου

Χρησιμοποιώντας την ιδιότητα 'Eccentricity' υπολογίζεται η εκκεντρότητα της περιοχής του αντικειμένου. Η εκκεντρότητα είναι ο λόγος της απόστασης των δύο εστιών της αντίστοιχης έλλειψης προς το μήκος του μεγαλύτερου άξονα της έλλειψης. Μαθηματικώς υπολογίζεται ως εξής:

$$e = \frac{\sqrt{a^2 - b^2}}{a}$$

όπου α ο μεγαλύτερος άξονας και b ο μικρότερος άξονας.

Το αποτέλεσμα της συνάρτησης του Matlab είναι ένας αριθμός μεταξύ 0 και 1. Όσο ο αριθμός πλησιάζει την τιμή 0 τόσο η περιοχή που έχει αναγνωριστεί ως αντικείμενο πλησιάζει τον κύκλο και όσο πλησιάζει την τιμή 1 τόσο η περιοχή τείνει να προσεγγίσει μια γραμμή.

## 3.5 Αποθήκευση δεδομένων

Για την μετέπειτα επεξεργασία των δεδομένων ήταν αναγκαία η αποθήκευσή τους σε ένα αρχείο. Τα δεδομένα αυτά χωρίζονται σε δύο κατηγορίες. Τα δεδομένα τα οποία προέρχονται από την επεξεργασία της εικόνας και αφορούν τα χαρακτηριστικά του

αντικειμένου και τα δεδομένα που προέρχονται από το ίδιο το ρομπότ και αφορούν τη θέση και τον προσανατολισμό της ρομποτικής αρπάγης.

Στο πρόγραμμα του Matlab που υλοποιήθηκε έγινε χρήση της εντολής fpintf με την οποία γίνεται η εξαγωγή των δεδομένων σε ένα αρχείο. Το αρχείο αυτό αποθηκεύεται στη διεύθυνση C:\traindata.txt αν πρόκειται για δεδομένα εκπαίδευσης ή στη διεύθυνση C:\testdata.txt αν πρόκειται για δεδομένα δοκιμής. Τα αρχεία αυτά έχουν τη μορφή πινάκων όπου κάθε γραμμή αντιστοιχεί σε ένα σύνολο δεδομένων εκπαίδευσης ή δοκιμής και κάθε στήλη στα χαρακτηριστικά ή δεδομένα που προέρχονται από την επεξεργασία της εικόνας ή του ρομποτικού χειριστή αντίστοιχα.

Πιο συγκεκριμένα, η πρώτη στήλη αφορά την τετμημένη του κέντρου του αντικειμένου, ενώ η δεύτερη στήλη την τεταγμένη του αντικειμένου όπως αυτή προκύπτει από την επεξεργασία της εικόνας. Στην τρίτη στήλη αποθηκεύεται ο προσανατολισμός του αντικειμένου, όπου θεωρήθηκε ότι οι γωνίες θα είναι μόνο θετικές στο διάστημα [0,180] και όχι αρνητικές. Στην τέταρτη και πέμπτη στήλη αποθηκεύονται το μήκος του πρωτεύοντος και το μήκος του δευτερεύοντος άξονα αντίστοιχα, ενώ στην έκτη στήλη ο λόγος των δύο αυτών μηκών. Τέλος, όσον αφορά τα δεδομένα του αντικειμένου που προκύπτουν από την επεξεργασία της εικόνας που λαμβάνει το ρομπότ, στην έβδομη στήλη αποθηκεύεται η εκκεντρότητα του αντικειμένου.

Στην συνέχεια έπονται οι στήλες που αφορούν τα δεδομένα που προέρχονται κατευθείαν από το ρομπότ. Έτσι, στην όγδοη, ένατη και δέκατη στήλη αποθηκεύονται η τετμημένη, η τεταγμένη και το βάθος της ρομποτικής αρπάγης αντίστοιχα με σημείο αναφοράς το κέντρο των παγκόσμιων συντεταγμένων. Στις τρείς επόμενες στήλες, δηλαδή στην ενδέκατη, στη δωδέκατη και στην δέκατη τρίτη στήλη αποθηκεύεται ο προσανατολισμός της ρομποτικής αρπάγης ο οποίος προκύπτει αντίστοιχα από τις γωνίες κύλισης (roll), εκτροπής (yaw) και ανύψωσης (pitch).

Τα δύο αυτά αρχεία που δημιουργήθηκαν, ένα για την εκπαίδευση και ένα για την δοκιμή της εκμάθησης του ρομπότ θα εισαχθούν ως είσοδοι σε νέο πρόγραμμα, όπως περιγράφεται σε επόμενο κεφάλαιο, που θα δημιουργηθεί στο περιβάλλον του προγράμματος Matlab.

ΚΕΦΑΛΑΙΟ 4

# ΕΚΜΑΘΗΣΗ ΡΟΜΠΟΤΙΚΟΥ ΧΕΙΡΙΣΤΗ ΜΕ ΝΕΥΡΩΝΙΚΟ ΔΙΚΤΥΟ

## 4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται, αρχικά, τα διαφορετικά είδη των νευρώνων [38]. Τα μοντέλα των νευρώνων που χρησιμοποιούνται είναι τα εξής:

- Απλός Νευρώνας βαθμωτής εισόδου χωρίς πόλωση
- Απλός Νευρώνας βαθμωτής εισόδου με πόλωση
- Νευρώνας με είσοδο διάνυσμα

Στη συνέχεια αναλύονται οι συναρτήσεις ενεργοποίησης των νευρώνων [38]. Οι συναρτήσεις που χρησιμοποιούνται πιο συχνά είναι οι εξής:

- Βηματική συνάρτηση ενεργοποίησης (Hard-limit Transfer Function)
- Γραμμική συνάρτηση ενεργοποίησης (Linear Transfer Function)
- Εφαπτομενική – σιγμοειδής συνάρτηση ενεργοποίησης (Tan-sigmoid Transfer Function)
- Λογαριθμική – σιγμοειδής συνάρτηση ενεργοποίησης (Log-sigmoid Transfer Function)

Έπειτα, αναφέρονται οι δύο αρχιτεκτονικές νευρωνικών δικτύων καθώς και οι δύο γενικοί τύποι νευρωνικών δικτύων [38]. Οι αρχιτεκτονικές των νευρωνικών δικτύων είναι οι εξής:

- Νευρωνικά δίκτυα ενός στρώματος (Single Layer Neural Networks)
- Πολυστρωματικά νευρωνικά δίκτυα (Multilayer Neural Networks)

Οι τύποι των νευρωνικών δικτύων είναι οι εξής:

- Νευρωνικά δίκτυα Προσοτροφοδότησης (Feed Forward Neural Networks)
- Νευρωνικά δίκτυα Ανατροφοδότησης (Recurrent Neural Networks)

Στη συνέχεια του κεφαλαίου περιγράφονται τα πρότυπα μάθησης νευρωνικών δικτύων καθώς και οι κυριότερες μέθοδοι μάθησης [38]. Τα γενικά πρότυπα μάθησης των νευρωνικών δικτύων είναι:

- Επιβλεπόμενη (ενεργή) μάθηση (Supervised Learning)
- Μη επιβλεπόμενη μάθηση (Unsupervised Learning)
- Ενισχυτική μάθηση (Reinforcement Learning)

Όσον αφορά τις μεθόδους μάθησης των νευρωνικών δικτύων, οι κυριότερες για τις οποίες γίνεται αναφορά στη συγκεκριμένη εργασία, είναι:

- Μάθηση διόρθωσης σφάλματος (Error Correction Learning)

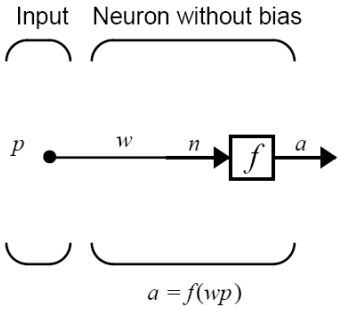
- Μάθηση Boltzmann (Boltzmann Learning)
- Ανταγωνιστική μάθηση (Competitive Learning)
- Μάθηση Hebb (Hebbian Learning)
- Μάθηση αυτο-οργανούμενων χαρτών
  (Self-organizing Maps)

Στο τέλος του κεφαλαίου αυτού αναλύεται το Νευρωνικό δίκτυο Perceptron, ο Αλγόριθμος οπισθοδρόμησης, ο Αλγόριθμος Levenberg – Marquardt καθώς και το νευρωνικό δίκτυο που χρησιμοποιήθηκε τελικά στη συγκεκριμένη εργασία.

## 4.2 Μοντέλα Νευρώνων

### 4.2.1 Απλός Νευρώνας βαθμωτής εισόδου χωρίς πόλωση

Στο Σχήμα 12 παρουσιάζεται σχηματικά ένας νευρώνας με βαθμωτή είσοδο χωρίς πόλωση.

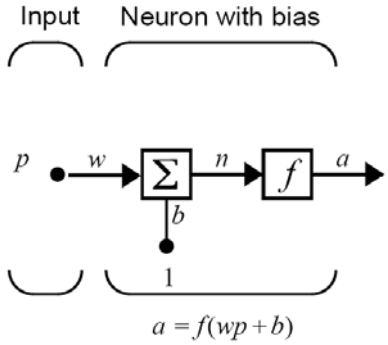



Σχήμα 12: Μοντέλο νευρώνα με βαθμωτή είσοδο χωρίς πόλωση.

Η βαθμωτή είσοδος *p* μεταδίδεται μέσω μιας σύνδεσης, η οποία πολλαπλασιάζεται επί το συντελεστή βαρύτητας *w* ώστε να σχηματιστεί το *wp*, το όποιο είναι και αυτό βαθμωτό μέγεθος. Η σταθμισμένη είσοδος *wp* αποτελεί το μοναδικό όρισμα της συνάρτησης ενεργοποίησης *f*, η οποία τελικά παράγει τη βαθμωτή έξοδο *a*.

### 4.2.2 Απλός Νευρώνας βαθμωτής εισόδου με πόλωση

Ένας νευρώνας με βαθμωτή είσοδο και πόλωση έχει την μορφή που φαίνεται στο Σχήμα 13.




Σχήμα 13: Μοντέλο νευρώνα με βαθμωτή είσοδο και πόλωση.

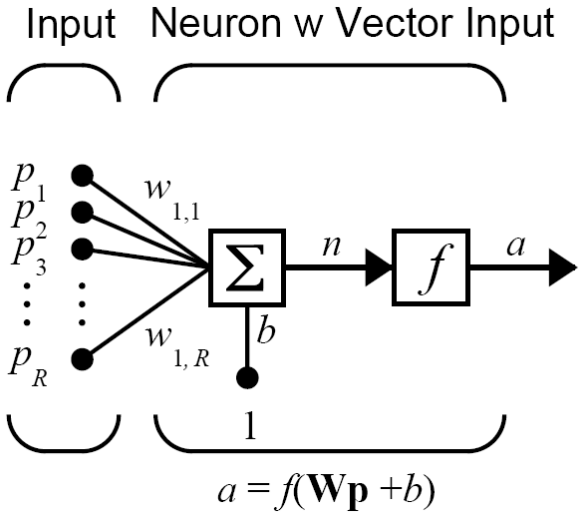
Ο νευρώνας, σε αυτή την περίπτωση, έχει μια βαθμωτή πόλωση *b*. Η πόλωση αυτή θεωρείται ως μια απλή πρόσθεση στη σταθμισμένη είσοδο *wp*, όπως προκύπτει από την πρόσθεση στη διασταύρωση ή ως μια μετατόπιση της συνάρτησης *f* στα αριστερά κατά μία ποσότητα *b*. Η πόλωση είναι ένας συντελεστής, όπως είναι και ο συντελεστής του βάρους, αλλά με τη διαφορά ότι η πόλωση έχει σταθερή τιμή ίση με τη μονάδα. Έτσι, το όρισμα της συνάρτησης μεταφοράς *f* θα είναι το βαθμωτό μέγεθος *n*, όπου $n = wp + b$.

Στο σημείο αυτό, πρέπει να επισημανθεί το γεγονός ότι τα μεγέθη *w* και *b* αποτελούν προσαρμόσιμες παραμέτρους του νευρώνα. Η κύρια ιδέα των νευρωνικών δικτύων είναι ότι αυτές οι παράμετροι μπορούν να μεταβληθούν ώστε τελικά το νευρωνικό δίκτυο να έχει την επιθυμητή συμπεριφορά. Έτσι, το νευρωνικό δίκτυο μπορεί να εκπαιδευθεί για να αποδίδει συγκεκριμένα αποτελέσματα μεταβάλλοντας τους συντελεστές του βάρους ή και της πόλωσης, ή μπορεί και από μόνο του το νευρωνικό δίκτυο να προσαρμόζει τις παραμέτρους αυτές για να επιτύχει τα επιθυμητά αποτελέσματα.

Όπως αναφέρθηκε και προηγουμένως, η πόλωση *b* είναι μια προσαρμόσιμη παράμετρος του νευρώνα και δεν αποτελεί κάποια είσοδο. Παρόλα αυτά η σταθερά 1 η οποία κατευθύνει την πόλωση είναι είσοδος και πρέπει να αντιμετωπίζεται ως είσοδος όταν αναφερόμαστε στη γραμμική εξάρτηση των διανυσμάτων εισόδου.

### 4.2.3 Νευρώνας με είσοδο διάνυσμα

Στο Σχήμα 14 παρουσιάζεται ένας νευρώνας με είσοδο ένα διάνυσμα p με R στοιχεία.




Σχήμα 14: Μοντέλο νευρώνα με είσοδο διάνυσμα.

Σε αυτήν την περίπτωση τα μεμονωμένα μεγέθη $p_1, p_2, \ldots p_R$ πολλαπλασιάζονται με τους συντελεστές βαρύτητας $w_{1,1}, w_{1,2}, \ldots, w_{1,R}$ και το σταθμισμένο αποτέλεσμα οδηγείται στην αθροιστική διασταύρωση. Το άθροισμα είναι το Wp, που είναι το εσωτερικό γινόμενο μιας γραμμής του πίνακα W και του διανύσματος p.

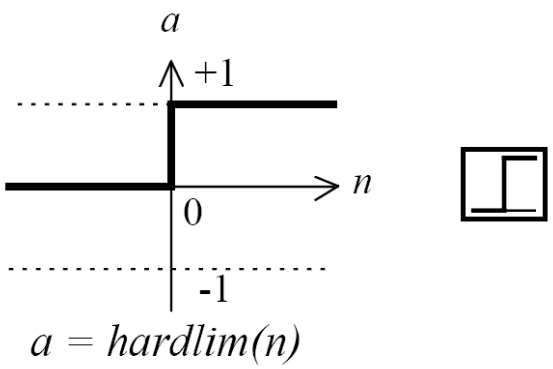
Ο νευρώνας έχει μια πόλωση *b*, η οποία αθροίζεται με τις σταθμισμένες εισόδους για να σχηματίσει την είσοδο *n*. Το άθροισμα *n* αποτελεί το όρισμα της συνάρτησης μεταφοράς *f* όπου:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b$$

## 4.3 Συναρτήσεις ενεργοποίησης νευρώνων

Παρακάτω αναλύονται οι τέσσερις πιο συχνά χρησιμοποιούμενες συναρτήσεις ενεργοποίησης. Υπάρχουν βέβαια και άλλες, οι οποίες χρησιμοποιούνται σε ιδιάζουσες περιπτώσεις.

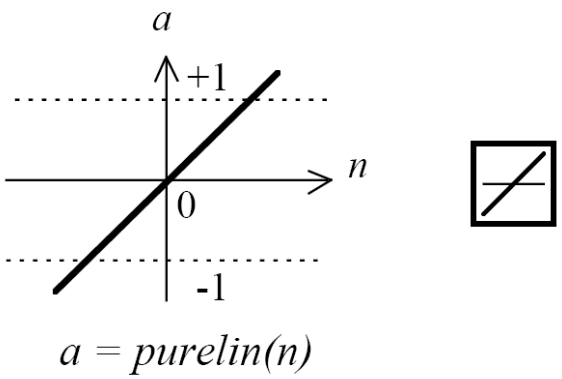
### 4.3.1 Βηματική συνάρτηση ενεργοποίησης (Hard-limit Transfer Function)




Σχήμα 15: Βηματική συνάρτηση ενεργοποίησης.

Η βηματική συνάρτηση ενεργοποίησης, που φαίνεται στο Σχήμα 15, θέτει ως έξοδο του νευρώνα είτε το 0, εάν το όρισμα εισόδου είναι μικρότερο από 0, είτε το 1, σε οποιαδήποτε άλλη περίπτωση. Συχνά, χρησιμοποιείται στα perceptrons ώστε να δημιουργηθούν νευρώνες κατάλληλοι για αποφάσεις κατηγοριοποίησης.

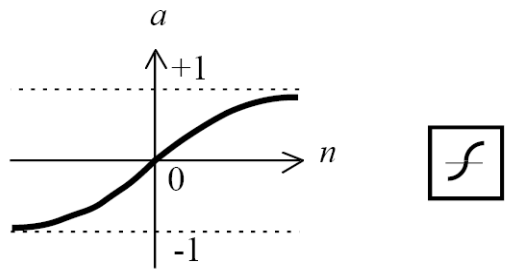
### 4.3.2 Γραμμική συνάρτηση ενεργοποίησης (Linear Transfer Function)




Σχήμα 16: Γραμμική συνάρτηση ενεργοποίησης.

Νευρώνες με γραμμική συνάρτηση ενεργοποίησης, όπως φαίνεται στο Σχήμα 16, χρησιμοποιούνται κυρίως σε προβλήματα γραμμικής προσέγγισης.

### 4.3.3 Εφαπτομενική – σιγμοειδής συνάρτηση ενεργοποίησης (Tan-sigmoid Transfer Function)

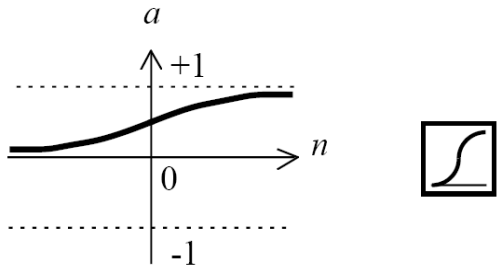



$a = tansig(n)$

Σχήμα 17: Εφαπτομενική – σιγμοειδής συνάρτηση ενεργοποίησης.

Η είσοδος της εφαπτομενικής – σιγμοειδούς συνάρτησης ενεργοποίησης, η οποία φαίνεται στο Σχήμα 17, μπορεί να είναι οποιαδήποτε τιμή στο διάστημα (-∞, +∞). Το πεδίο τιμών της όμως, περιορίζεται στο διάστημα (-1,1). Η συνάρτηση αυτή αποτελεί την κύρια επιλογή σε πολυστρωματικά δίκτυα τύπου οπισθοδρόμησης (multilayer back propagation networks) λόγω της διαφορισιμότητάς της.

### 4.3.4 Λογαριθμική – σιγμοειδής συνάρτηση ενεργοποίησης (Log-sigmoid Transfer Function)




$a = logsig(n)$

Σχήμα 18: Λογαριθμική – σιγμοειδής συνάρτηση ενεργοποίησης.

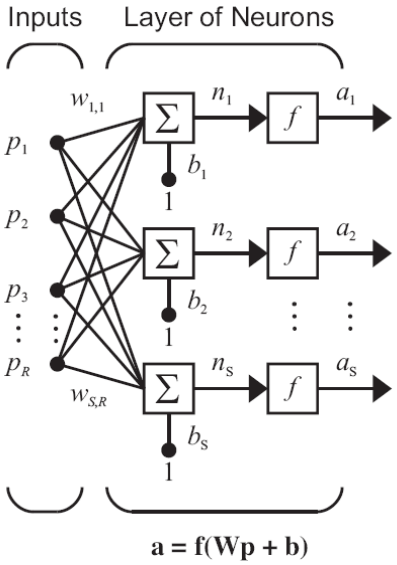
Η λογαριθμική – σιγμοειδής συνάρτηση ενεργοποίησης (Σχήμα 18) έχει μια είσοδο, η οποία μπορεί να έχει τιμές από -∞ έως +∞ και περιορίζει την έξοδο στο πεδίο τιμών (0,1). Η συνάρτηση αυτή είναι κατάλληλη για δίκτυα τύπου οπισθοδρόμησης (back propagation) αφού η συνάρτηση αυτή είναι διαφορίσιμη.

## 4.4 Αρχιτεκτονικές νευρωνικών δικτύων

Η αρχιτεκτονική των δικτύων είναι το κύριο χαρακτηριστικό των νευρωνικών δικτύων και αναφέρεται στους τρόπους με του οποίους τοποθετούνται και διασυνδέονται πολλαπλοί νευρώνες. Οι δύο βασικές ιδιότητες που καθορίζουν την αρχιτεκτονική ενός νευρωνικού δικτύου είναι το πλήθος των επιπέδων (layers) και οι συνδέσεις μεταξύ των νευρώνων. Το τρίτο χαρακτηριστικό, το οποίο σχετίζεται και με τον τρόπο κατά τον οποίο είναι δομημένοι οι νευρώνες, είναι ο αλγόριθμος μάθησης που χρησιμοποιείται για την εκπαίδευση του δικτύου. Η μάθηση θα εξεταστεί σε επόμενη παράγραφο.

### 4.4.1 Νευρωνικά δίκτυα ενός στρώματος (Single Layer Neural Networks)

Όταν δύο ή περισσότεροι νευρώνες συνδυαστούν τότε σχηματίζουν ένα στρώμα ή επίπεδο νευρώνων. Στο Σχήμα 19 ακολουθεί η γραφική αναπαράσταση ενός δικτύου με ένα επίπεδο νευρώνων με *R* στοιχεία εισόδου και *S* νευρώνες.




Σχήμα 19: Νευρωνικό δίκτυο με ένα επίπεδο νευρώνων.

Σε αυτό το δίκτυο κάθε στοιχείο του διανύσματος εισόδου συνδέεται με την είσοδο κάθε νευρώνα μέσω του πίνακα των συντελεστών βαρύτητας W. Ο i-οστός νευρώνας έχει έναν αθροιστή ο οποίος συλλέγει τις σταθμισμένες εισόδους και τις πολώσεις για να σχηματίσει τη δική του βαθμωτή είσοδο *n(i)*. Εάν συνδυαστούν οι διάφοροι είσοδοι *n(i)*, δημιουργούν ένα διάνυσμα εισόδου n με *S* πλήθος στοιχείων. Τελικά, το νευρωνικό δίκτυο έχει ως έξοδο ένα διάνυσμα a.

Να σημειωθεί ότι είναι σύνηθες ο αριθμός των εισόδων σε ένα επίπεδο να είναι διαφορετικός από τον αριθμό των νευρώνων που το αποτελούν, δηλαδή η τιμή του *R* δεν είναι απαραίτητα ίδια με την τιμή του *S*. Έτσι, ένα επίπεδο δεν περιορίζεται στο να έχει τον ίδιο αριθμό εισόδων με τον αριθμό των νευρώνων. Ένα νευρωνικό δίκτυο με ένα επίπεδο νευρώνων και με διαφορετικές συναρτήσεις μεταφοράς για κάθε νευρώνα μπορεί να δημιουργηθεί αν απλά τοποθετηθούν τα δίκτυα παράλληλα. Έτσι κάθε δίκτυο θα έχει τις ίδιες εισόδους και θα παράγει μερικές από τις εξόδους.

Ο πίνακας των συντελεστών βαρύτητας έχει την παρακάτω μορφή.

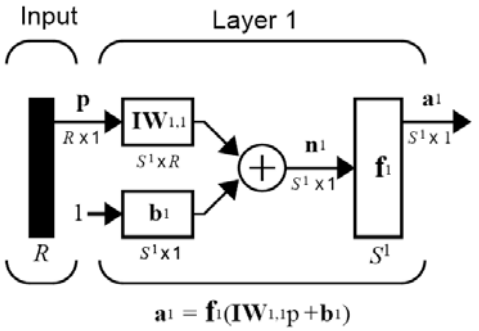
$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & . & . & . & w_{1,R} \\ w_{2,1} & w_{2,2} & . & . & . & w_{2,R} \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ w_{S,1} & w_{s,2} & . & . & . & w_{S,R} \end{bmatrix}$$

Η γραμμή του πίνακα υποδεικνύει τον νευρώνα στον οποίο θα εφαρμοστεί αυτό το βάρος, ενώ η στήλη υποδηλώνει την πηγή της εισόδου για το βάρος αυτό.

### 4.4.2 Πολυστρωματικά νευρωνικά δίκτυα (Multilayer Neural Networks)

Για να περιγραφούν τα δίκτυα πολλαπλών επιπέδων, πρέπει να γίνουν κάποιες διευκρινήσεις. Ειδικότερα, πρέπει να γίνει διαχωρισμός μεταξύ των πινάκων των συντελεστών βαρύτητας που συνδέονται με τις εισόδους και των πινάκων των συντελεστών βαρύτητας, οι οποίοι συνδέονται με τα διάφορα επίπεδα. Θα καλούνται, λοιπόν, οι πίνακες που συνδέονται με τις εισόδους, βάρη εισόδων, ενώ οι πίνακες που συνδέονται με τις εξόδους των νευρωνικών επιπέδων, βάρη επιπέδων. Για να γίνεται αντιληπτό σε ποιο επίπεδο αναφέρεται η κάθε παράμετρος γίνεται χρήση ενός δείκτη. Από εδώ και στο εξής θα υιοθετηθεί η παραπάνω σύμβαση για την περαιτέρω περιγραφή των νευρωνικών δικτύων.

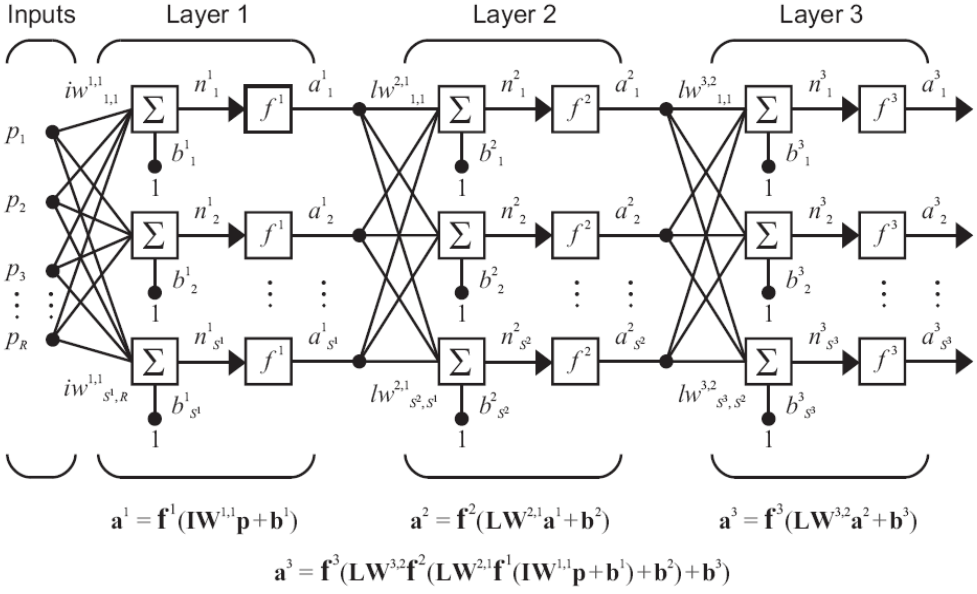
Στο Σχήμα 20 αναπαρίσταται γραφικώς το επίπεδο εισόδου ενός πολυεπίπεδου νευρωνικού δικτύου.




Σχήμα 20: Μοντέλο εισόδου ενός πολυεπίπεδου νευρωνικού δικτύου.

Όπως αναφέρθηκε και παραπάνω ένα νευρωνικό δίκτυο μπορεί να αποτελείται από αρκετά επίπεδα. Το επίπεδο εισόδου έχει ένα πίνακα βαρών IW (Input Weight Matrix), κάθε επόμενο επίπεδο έχει ένα πίνακα LW (Layer Weight Matrix), μία πόλωση b και ένα διάνυσμα εξόδου a. Ο συντελεστής που βρίσκεται στο πάνω μέρος των διάφορων μεγεθών σκοπό έχει να καθορίσει σε ποιο επίπεδο αναφέρονται οι πίνακες βαρών, τα διανύσματα εξόδου κτλ..

Το Σχήμα 21 αναπαριστά γραφικά ένα πολυστρωματικό νευρωνικό δίκτυο.




Σχήμα 21: Πολυστρωματικό νευρωνικό δίκτυο.

Το παραπάνω δίκτυο, στο Σχήμα 21, έχει $R^1$ εισόδους και $S^1$ νευρώνες στο πρώτο επίπεδο, $S^2$ νευρώνες στο δεύτερο επίπεδο κοκ. Είναι σύνηθες τα διαφορετικά επίπεδα να έχουν διαφορετικό αριθμό νευρώνων. Η είσοδος της πόλωσης είναι σταθερή για κάθε νευρώνα και ίση με τη μονάδα.

Από το Σχήμα 21 φαίνεται, επίσης, ότι η έξοδος ενός ενδιάμεσου επιπέδου αποτελεί την είσοδο για το επόμενο επίπεδο. Επομένως, το δεύτερο επίπεδο μπορεί να αναλυθεί ως ένα επίπεδο με $S^1$ εισόδους, $S^2$ νευρώνες και έναν $S^2 \cdot S^1$ πίνακα βαρών $LW^2$. Η είσοδος του επιπέδου 2 είναι $a^1$ και η έξοδος του $a^2$. Ύστερα, από τον καθορισμό των παραπάνω παραμέτρων μπορεί το συγκεκριμένο επίπεδο να αντιμετωπιστεί ως ένα νευρωνικό δίκτυο ενός επιπέδου νευρώνων. Αυτή η προσέγγιση μπορεί να γίνει για κάθε επίπεδο του δικτύου.

Τα διάφορα επίπεδα ενός πολυεπίπεδου νευρωνικού δικτύου διαδραματίζουν διαφορετικούς ρόλους το καθένα. Το επίπεδο που παράγει την έξοδο του δικτύου καλείται επίπεδο εξόδου (output layer). Όλα τα υπόλοιπα επίπεδα καλούνται κρυφά επίπεδα (hidden layers).

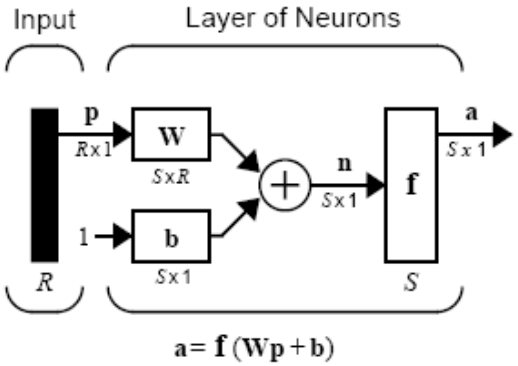
Τα πολυστρωματικά δίκτυα είναι πολύ ισχυρά εργαλεία. Για παράδειγμα ένα νευρωνικό δίκτυο με δύο επίπεδα, όπου το πρώτο χρησιμοποιεί σιγμοειδή συνάρτηση μεταφοράς και το δεύτερο γραμμική συνάρτηση μεταφοράς μπορεί να εκπαιδευτεί στο να προσεγγίζει αρκετά καλά οποιαδήποτε συνάρτηση με πεπερασμένο πλήθος ασυνεχειών.

## 4.5 Τύποι νευρωνικών δικτύων

Παρακάτω αναλύονται οι δύο συνηθέστεροι τύποι νευρωνικών δικτύων που χρησιμοποιούνται σήμερα.

### 4.5.1 Νευρωνικά δίκτυα Προσοτροφοδότησης (Feed Forward Neural Networks)

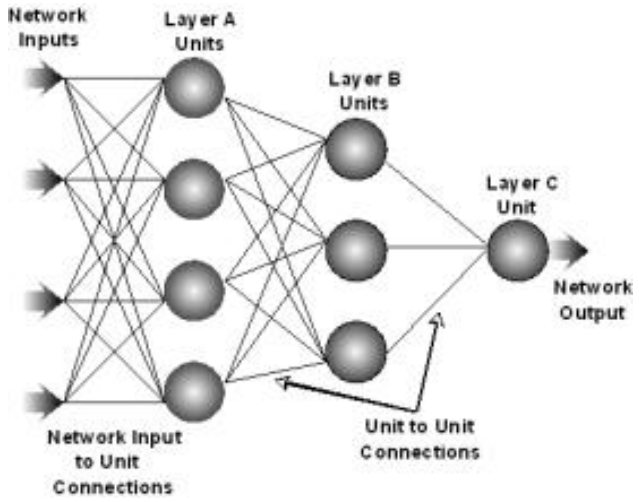
Στην απλούστερη περίπτωση, ένα πολυεπίπεδο νευρωνικό δίκτυο έχει ένα επίπεδο εισόδου από κόμβους πηγής (source nodes) το οποίο προβάλλεται πάνω σε ένα επίπεδο νευρώνων εξόδου, αλλά όχι αντίστροφα. Ένα τέτοιο νευρωνικό δίκτυο είναι αυστηρά του τύπου «προσοτροφοδότησης» (feed forward neural network) (Σχήμα 22) και καλείται «νευρωνικό δίκτυο προσοτροφοδότησης ενός επιπέδου». Το μοναδικό επίπεδο του νευρωνικού δικτύου αποτελεί το επίπεδο των νευρώνων εξόδου (υπολογιστικών κόμβων). Αυτό σημαίνει ότι το επίπεδο των κόμβων πηγής (εισόδου) δεν προσμετράται γιατί δεν λαμβάνει χώρα κανένας υπολογισμός σε αυτό.




Σχήμα 22: Νευρωνικό δίκτυο προσοτροφοδότησης ενός επιπέδου με $S$ νευρώνες εξόδου.

Στη γενική περίπτωση, ένα νευρωνικό δίκτυο προσοτροφοδότησης περιέχει ένα ή περισσότερα κρυφά επίπεδα (hidden layers), των οποίων οι υπολογιστικοί κόμβοι είναι γνωστοί ως «κρυφοί νευρώνες» ή «κρυφές μονάδες» και παρεμβαίνουν μεταξύ των εξωτερικών εισόδων και των εξόδων του νευρωνικού δικτύου. Στα δίκτυα αυτά, που ονομάζονται «πολυεπίπεδα νευρωνικά δίκτυα προσοτροφοδότησης», οι κόμβοι πηγής στο επίπεδο εισόδου παρέχουν τα στοιχεία του διανύσματος εισόδου (πρότυπο δράσης) τα οποία εισέρχονται ως είσοδοι στο πρώτο κρυφό επίπεδο υπολογιστικών κόμβων. Ομοίως, οι έξοδοι των κόμβων του πρώτου κρυφού επιπέδου εισέρχονται ως είσοδοι στους κόμβους του δεύτερου κρυφού επιπέδου, κοκ. Το τελικό επίπεδο κόμβων (επίπεδο εξόδου) δίνει τη συνολική ή τις συνολικές αποκρίσεις στα διανύσματα εισόδου (δηλαδή στα πρότυπα εξωτερικής δράσης).

Ένα παράδειγμα νευρωνικού δικτύου προσοτροφοδότησης με δύο κρυφά επίπεδα τεσσάρων και τριών κόμβων αντίστοιχα, τέσσερις κόμβους πηγής στο επίπεδο εισόδου και ένα κόμβο στο επίπεδο εξόδου εικονίζεται στο Σχήμα 23. Το δίκτυο αυτό αναφέρεται συμβολικά ως νευρωνικό δίκτυο προσοτροφοδότησης 4-4-3-1 (4 κόμβοι εισόδου, 4 κρυμμένοι κόμβοι στο πρώτο κρυφό επίπεδο, 3 κρυμμένοι κόμβοι στο δεύτερο κρυμμένο επίπεδο και 1 κόμβος εξόδου).




Σχήμα 23: Πολυεπίπεδο νευρωνικό δίκτυο προσοτροφοδότησης 4-4-3-1

Το νευρωνικό δίκτυο του παραπάνω σχήματος ονομάζεται «νευρωνικό δίκτυο πλήρως διασυνδεδεμένο» γιατί κάθε κόμβος οποιουδήποτε επιπέδου συνδέεται με όλους τους κόμβους του γειτονικού του προς τα εμπρός (δηλαδή του επομένου) επιπέδου. Εάν αυτό δεν ισχύει, δηλαδή εάν λείπουν μερικοί σύνδεσμοι επικοινωνίας (συναπτικές συνδέσεις), τότε το νευρωνικό δίκτυο ονομάζεται «νευρωνικό δίκτυο μερικώς διασυνδεδεμένο».

### 4.5.2 Νευρωνικά δίκτυα Ανατροφοδότησης (Recurrent Neural Networks)

«Αναδρομικό νευρωνικό δίκτυο» (Recurrent Neural Network) ή νευρωνικό δίκτυο ανατροφοδότησης ονομάζεται το νευρωνικό δίκτυο που περιέχει τουλάχιστον ένα βρόχο ανατροφοδότησης, ο οποίος ανακυκλώνει τις πληροφορίες μέσω του ίδιου ή προηγούμενων επιπέδων. Το αποτέλεσμα της ανατροφοδότησης προκύπτει από ένα διάνυσμα (πρότυπο) εισόδου που εισέρχεται στο αναδρομικό νευρωνικό δίκτυο, το οποίο, ωστόσο, δεν παράγει ένα πρότυπο εξόδου σε πεπερασμένο αριθμό χρονικών βημάτων, αλλά ενεργεί με κυκλικό

τρόπο, ενεργοποιώντας τα ίδια επίπεδα επαναληπτικά. Εάν το νευρωνικό δίκτυο είναι ευσταθές είναι πιθανό να ταλαντωθεί για κάποιο χρονικό διάστημα προτού φθάσει σε μια σταθερή κατάσταση, στην οποία οι νευρωνικές ενεργοποιήσεις θα σταματήσουν να αλλάζουν, με αποτέλεσμα να παραχθεί μια σταθερή έξοδος. Διαφορετικά, εάν το νευρωνικό δίκτυο δεν είναι ευσταθές, οι ταλαντώσεις θα συνεχίσουν αδιάκοπα. Συνεπώς όταν εκπαιδεύουμε ένα αναδρομικό νευρωνικό δίκτυο είναι σημαντικό να βρούμε το σύνολο των συναπτικών βαρών που του επιτρέπουν να σταθεροποιηθεί στις επιθυμητές τιμές εξόδου.

## 4.6 Πρότυπα μάθησης νευρωνικών δικτύων

Αυτό που έχει προσελκύσει περισσότερο το ενδιαφέρον για τα νευρωνικά δίκτυα είναι η δυνατότητα της μάθησης. Μάθηση σημαίνει χρησιμοποιώντας ένα σύνολο από παρατηρήσεις, δεδομένου ενός συγκεκριμένου προβλήματος που πρέπει να επιλυθεί και μιας κλάσης από συναρτήσεις F, να βρεθεί μία συνάρτηση $f^* \in F$, η οποία να επιλύει το πρόβλημα με βέλτιστο τρόπο. Αυτό περιλαμβάνει τον ορισμό μιας συνάρτησης κόστους $C : F \to \Re$ τέτοια ώστε για την βέλτιστη λύση $f^*$ να ισχύει ότι $C(f^*) \le C(f) \forall f \in F$, δηλαδή, να μην υπάρχει άλλη λύση με μικρότερο κόστος από τη βέλτιστη λύση.

Η συνάρτηση κόστους C είναι ένας σημαντικός παράγοντας στη μάθηση, καθώς είναι ένα είδος μέτρησης του πόσο μακριά απέχει μία λύση από τη βέλτιστη λύση του προβλήματος. Οι αλγόριθμοι μάθησης αναζητούν στο χώρο των λύσεων μία συνάρτηση, η οποία να έχει το μικρότερο δυνατό κόστος. Στις εφαρμογές τις οποίες η λύση εξαρτάται από τα δεδομένα, το κόστος πρέπει να είναι μια συνάρτηση των παρατηρήσεων, αλλιώς δεν θα μοντελοποιούταν τίποτα σχετικό με τα δεδομένα.

Η αναζήτηση της βέλτιστης λύσης επιτυγχάνεται με συνεχόμενες αλλαγές των συναπτικών βαρών και κάθε μέθοδος μάθησης προσφέρει έναν διαφορετικό τρόπο προσαρμογής (επιλογής ή ανανέωσης) των συναπτικών αυτών βαρών. Τα πρότυπα μάθησης χωρίζονται σε τρεις μεγάλες κατηγορίες:

- Επιβλεπόμενη (ενεργή) μάθηση (Supervised Learning)
- Μη επιβλεπόμενη μάθηση (Unsupervised Learning)
- Ενισχυτική μάθηση (Reinforcement Learning)

### 4.6.1 Επιβλεπόμενη (ενεργή) μάθηση (Supervised Learning)

Στην επιβλεπόμενη μάθηση [40] έχουμε ένα σύνολο από ζευγάρια παραδειγμάτων $(x, y)$, $x \in X$, $y \in Y$ και σκοπός είναι να βρεθεί μία συνάρτηση $f : X \to Y$ από την κλάση των συναρτήσεων F, η οποία να αντιστοιχίζει τα δοθέντα παραδείγματα. Με άλλα λόγια επιθυμούμε να προσδιορίσουμε την αντιστοιχία που προκύπτει από τα δεδομένα. Η συνάρτηση κόστους σχετίζεται με τον κακό συνδυασμό ανάμεσα στην αντιστοιχία που βρέθηκε και στα δεδομένα.

Μία συχνά χρησιμοποιούμενη συνάρτηση κόστους είναι το μέσο τετραγωνικό σφάλμα (mean squared error) το οποίο στόχο έχει να μειώσει το σφάλμα μεταξύ της εξόδου του νευρωνικού δικτύου και της τιμής στόχου σε όλα τα ζευγάρια παραδειγμάτων. Όταν η ελαχιστοποίηση του κόστους προκύπτει από την κλίση της κλάσης (μερική παράγωγος) των νευρωνικών δικτύων τότε η μέθοδος αυτή καλείται μέθοδος οπισθοδρόμησης και είναι μία από τις πιο διαδεδομένες μεθόδους μάθησης των νευρωνικών δικτύων.

### 4.6.2 Μη επιβλεπόμενη μάθηση (Unsupervised Learning)

Στην μη επιβλεπόμενη μάθηση ([41],[42]) έχουμε ως δεδομένα μόνο ένα σύνολο εισόδων $x \in X$ και όχι ζευγάρια παραδειγμάτων $(x, y),\ x \in X,\ y \in Y$ όπως στην επιβλεπόμενη μάθηση. Ακόμα δίνεται και μια συνάρτηση κόστους προς ελαχιστοποίηση, η οποία μπορεί να είναι οποιουδήποτε είδους συνάρτησης μεταξύ των δεδομένων x και της εξόδου του νευρωνικού δικτύου. Η συνάρτηση αυτή εξαρτάται από το πρόβλημα προς μοντελοποίηση και τις αρχικές υποθέσεις για το μοντέλο αυτό, όπως είναι οι ιδιότητές του, οι παράμετροι και οι παρατηρήσημες μεταβλητές.

Πρακτικά, το μόνο που χρειάζεται ένα νευρωνικό δίκτυο μη επιβλεπόμενης μάθησης είναι να συντονισθεί στις στατιστικές ομαλότητες των δεδομένων εισόδου και μετά να μπορέσει να δημιουργήσει εσωτερικές παραστάσεις για την κωδικοποίηση των ιδιοτήτων (χαρακτηριστικών) της εισόδου και να παραγάγει αυτόματα νέες κατηγορίες (κλάσεις). Η μη επιβλεπόμενη μάθηση μπορεί να συνδυασθεί με την επιβλεπόμενη μάθηση σε ένα πολυεπίπεδο δίκτυο προσοτροφοδότησης εκπαιδευόμενο με τον αλγόριθμο οπισθοδρόμησης για να επιταχύνει τη διαδικασία μάθησης.

### 4.6.3 Ενισχυτική μάθηση (Reinforcement Learning)

Κατά την ενισχυτική μάθηση ([43],[44]), το σύνολο εισόδων $x \in X$, συνήθως, δεν δίνεται απευθείας αλλά από ένα διαμεσολαβητή με το περιβάλλον. Σε κάθε χρονική στιγμή t, ο διαμεσολαβητής εκτελεί μία ενέργεια $y_t$ και το περιβάλλον παράγει μία παρατήρηση $x_t$ και ένα στιγμιαίο κόστος ανάλογα με κάποια, συνήθως, γνωστή δυναμική. Ο στόχος είναι να βρεθεί μία πολιτική επιλογής κατάλληλης ενέργειας, η οποία ελαχιστοποιεί μια τιμή ενός μακροχρόνιου κόστους (το συνολικό συνήθως εκτιμώμενο κόστος). Η δυναμική του περιβάλλοντος αλλά και το μακροχρόνιο κόστος είναι, συνήθως, γνωστά αλλά μπορούν και να εκτιμηθούν με στατιστικές μεθόδους.

Πιο συγκεκριμένα το περιβάλλον μοντελοποιείται ως μια διαδικασία επιλογής του Markov (Markov Decision Process – MDP) και οι αλγόριθμοι ενισχυτικής μάθησης σχετίζονται αρκετά με τις μεθόδους δυναμικού προγραμματισμού. Η διαδικασία αυτή περιλαμβάνει:

- Ένα σύνολο από καταστάσεις (states) S
- Ένα σύνολο από ενέργειες (actions) A
- Ένα σύνολο από επιβραβεύσεις (rewards) στο $\Re$

Σε κάθε χρονική στιγμή t, ο διαμεσολαβητής αντιλαμβάνεται την κατάστασή του $s_t \in S$ και ένα σύνολο από πιθανές ενέργειες $A(s_t)$. Επιλέγει στη συνέχεια μια ενέργεια $a \in A(s_t)$ και λαμβάνει από το περιβάλλον τη νέα κατάσταση $s_{t+1}$ και μια επιβράβευση $r_t$. Βασισμένος σε αυτές τις αλληλεπιδράσεις, ο διαμεσολαβητής ενισχυτικής μάθησης, πρέπει να αναπτύξει μια πολιτική $\pi : S \times T \to A$ (όπου T το σύνολο των πιθανών χρονικών στιγμών), η οποία να μεγιστοποιεί την ποσότητα $R = r_o + r_1 + ... + r_n$ για τις διαδικασίες επιλογής του Markov που έχουν ένα τερματικό κόστος ή την ποσότητα $R = \sum_{t=0}^{\infty} \gamma^t t_t$ για τις διαδικασίες επιλογής του Markov που δεν έχουν τερματικό κόστος (όπου $0 \le \gamma \le 1$ είναι ένας παράγοντας εξόφλησης της μελλοντικής επιβράβευσης).

## 4.7 Μέθοδοι μάθησης νευρωνικών δικτύων

Παρακάτω παρουσιάζονται οι κυριότερες μέθοδοι μάθησης των νευρωνικών δικτύων.

### 4.7.1 Μάθηση διόρθωσης σφάλματος (Error Correction Learning)

Στη μέθοδο διόρθωσης σφάλματος [45], η οποία χρησιμοποιείται στην επιβλεπόμενη μάθηση, συγκρίνεται η έξοδος του νευρωνικού δικτύου με την επιθυμητή έξοδο και χρησιμοποιείται αυτό το σφάλμα για την εκπαίδευση του δικτύου. Στην πιο άμεση περίπτωση, οι τιμές του σφάλματος χρησιμοποιούνται απευθείας στην προσαρμογή των συναπτικών βαρών χρησιμοποιώντας έναν αλγόριθμο. Ο πιο ευρέως διαδεδομένος αλγόριθμος, που χρησιμοποιεί τη μέθοδο διόρθωσης σφάλματος, είναι η οπισθοδρόμηση. Αν η έξοδος του συστήματος συμβολίζεται με y και η επιθυμητή έξοδος με d τότε το σήμα σφάλματος ορίζεται ως η διαφορά $e = d - y$. Οι αλγόριθμοι που χρησιμοποιούν την μέθοδο διόρθωσης σφάλματος τείνουν να ελαχιστοποιούν το σήμα σφάλματος σε κάθε επανάληψη της εκπαίδευσης.

### 4.7.2 Μάθηση Boltzmann (Boltzmann Learning)

Η μέθοδος μάθησης του Boltzmann [45] είναι μια μορφή στατιστικής μάθησης και προήλθε από τον επιστημονικό χώρο της θερμοδυναμικής. Η μέθοδος αυτή χρησιμοποιείται κατά την επιβλεπόμενη μάθηση και μοιάζει με την μέθοδο διόρθωσης σφάλματος. Η ομοιότητα έγκειται στο γεγονός ότι χρησιμοποιείται ένα σήμα σφάλματος για την εκπαίδευση του δικτύου σε κάθε επανάληψη, το οποίο όμως δεν προκύπτει από τη διαφορά ανάμεσα στην πραγματική έξοδο και στην επιθυμητή, αλλά από τη διαφορά στις κατανομές πιθανότητας του συστήματος. Στη μέθοδο αυτή, πέρα από την έξοδο του νευρωνικού δικτύου, λαμβάνεται υπόψη και η κατάσταση των μεμονωμένων νευρώνων. Η κατάσταση των νευρώνων μπορεί να έχει δύο επιτρεπτές τιμές, την κατάσταση “on” (+1) και την κατάσταση “off” (-1). Για το λόγο αυτό, η μέθοδος μάθησης του Boltzmann είναι σαφώς πιο αργή από την μέθοδο διόρθωσης σφάλματος.

### 4.7.3 Ανταγωνιστική μάθηση (Competitive Learning)

Η μέθοδος της ανταγωνιστικής μάθησης [45] βασίζεται στην ιδέα ότι μόνο ένας νευρώνας, για μια δεδομένη επανάληψη και σε ένα συγκεκριμένο επίπεδο, θα ενεργοποιείται. Έτσι τα συναπτικά βάρη προσαρμόζονται έτσι ώστε μόνο ένας νευρώνας σε κάθε επίπεδο να ενεργοποιείται. Η ανταγωνιστική μάθηση είναι πολύ χρήσιμη σε προβλήματα ταξινόμησης των δεδομένων εισόδου σε ένα διακριτό σύνολο από κλάσεις εξόδου. Ο νευρώνας – νικητής που ερεθίζεται σε κάθε επανάληψη είναι αυτός που έχει τη μεγαλύτερη σταθμισμένη είσοδο.

Μαθηματικώς, αυτή η λογική, μπορεί να οριστεί ως η εξίσωση:

$$w_{ij}[n+1] = w_{ij}[n] + \Delta w_{ij}[n]$$

όπου $\Delta w_{ij}[n] = \begin{cases} \eta\left(x_i - w_{ij}\right) & , \varepsilon\acute{\alpha}\nu\ i = j \\ 0 & , \varepsilon\acute{\alpha}\nu\ i \neq j \end{cases}$, $\eta$ ένας συντελεστής ρυθμού μάθησης, x η έξοδος του νευρώνα και w το συναπτικό βάρος των δύο νευρώνων.

### 4.7.4 Μάθηση Hebb (Hebbian Learning)

Η μέθοδος μάθησης του Hebb [45] είναι μία από τις πιο παλιές μεθόδους μάθησης και βασίζεται κατά ένα μεγάλο μέρος στη δυναμική των βιολογικών συστημάτων. Στη μέθοδο αυτή δημιουργείται μία σύναψη μεταξύ δύο νευρώνων, η οποία ενδυναμώνεται όταν οι νευρώνες από οποιαδήποτε πλευρά της σύναψης (είτε από την είσοδο είτε από την έξοδο) έχουν υψηλά συσχετισμένες εξόδους. Έτσι, όταν ένας νευρώνας εισόδου ερεθιστεί, συνήθως οδηγεί και στον ερεθισμό ενός νευρώνα εξόδου και συνεπώς η σύναψη αυτή ενδυναμώνεται. Αντίστοιχα, σε ένα τεχνητό νευρωνικό δίκτυο, όταν υπάρχει υψηλή συσχέτιση μεταξύ δύο νευρώνων τότε το αντίστοιχο συναπτικό βάρος αυξάνεται. Σε οποιαδήποτε άλλη περίπτωση η σύναψη σε ένα βιολογικό νευρωνικό δίκτυο αποδυναμώνεται και αντίστοιχα σε ένα τεχνητό νευρωνικό δίκτυο το συναπτικό βάρος μειώνεται.

Μαθηματικώς μπορούμε να περιγράψουμε την μέθοδο του Hebb με την εξίσωση: $w_{ij}[n+1] = w_{ij}[n] + \eta x_i[n] x_j[n]$

όπου $\eta$ είναι ένας συντελεστής ρυθμού μάθησης, x οι έξοδοι των αντίστοιχων νευρώνων και w το συναπτικό βάρος τους.

Η μέθοδος του Hebb εφαρμόζεται τοπικά και για αυτό το λόγο δεν έχει ως παραμέτρους τα συνολικά χαρακτηριστικά της εισόδου-εξόδου. Η μέθοδος αυτή χρησιμοποιείται για την εκμάθηση βιολογικών νευρωνικών δικτύων καθώς και σε VLSI εφαρμογές επειδή τα τοπικά σήματα είναι πολύ πιο εύκολο να προσδιοριστούν.

### 4.7.5 Μάθηση αυτο-οργανούμενων χαρτών (Self-organizing Maps)

Η μέθοδος των αυτο-οργανούμενων αντιστοιχιών (Self-organizing Maps – SOM) [46], που συχνά καλείται και μέθοδος των αυτο-οργανούμενων αντιστοιχιών του Kohonen από το όνομα του δημιουργού τους (Teuvo Kohonen), χρησιμοποιείται στη μη επιβλεπόμενη μάθηση. Συχνά, οι αντιστοιχίες αυτές μοντελοποιούν βιολογικά νευρωνικά δίκτυα, όπου ομάδες νευρώνων συνήθως αυτο-οργανώνονται σε συγκεκριμένες περιοχές για να εκτελέσουν μια κοινή λειτουργία.

Στα δίκτυα αυτο-οργανούμενων αντιστοιχιών διαφορετικές περιοχές εκπαιδεύονται για την αναγνώριση ευδιάκριτων χαρακτηριστικών ενός συνόλου εισόδου. Τα αρχικά συναπτικά βάρη επιλέγονται είτε τυχαία είτε με βάση τα ιδιοδιανύσματα του χώρου της εισόδου. Στη συνέχεια υπολογίζεται η Ευκλείδεια απόσταση του κάθε δείγματος εισόδου από το διάνυσμα βάρους του κάθε νευρώνα και ο νευρώνας, του οποίου το διάνυσμα βάρους είναι περισσότερο όμοιο με την είσοδο, ορίζεται ως η καλύτερη μονάδα ταύτισης (Best Match Unit – BMU).

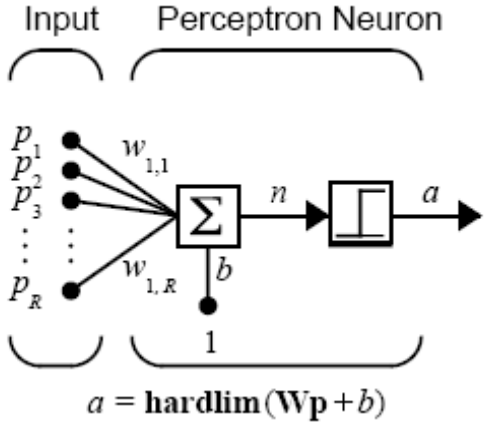
Η εξίσωση $w_j[n+1] = w_j[n] + \Theta[j,n] a[n] \left( x[n] - w_j[n] \right)$ προσδιορίζει τα νέα προσαρμοσμένα συναπτικά βάρη, όπου w είναι το διάνυσμα βάρους σε μια δεδομένη χρονική στιγμή n, α είναι μια φθίνουσα συνάρτηση, η οποία εξασφαλίζει ότι ο ρυθμός μάθησης θα μειώνεται με την πάροδο του χρόνου, x είναι το διάνυσμα εισόδου και $\Theta[j,n]$ είναι η μέτρηση της απόστασης μεταξύ της καλύτερης μονάδας ταύτισης και του αντίστοιχου νευρώνα j τη χρονική στιγμή n.

## 4.8 Νευρωνικό δίκτυο Perceptron

Το νευρωνικό δίκτυο Perceptron [45] αναπτύχθηκε από τον Rosenblatt (1958) και αποτελεί μια από τις πρώτες προσπάθειες σχεδίασης και κατασκευής ευφυών συστημάτων με δυνατότητες αυτό-μάθησης. Το perceptron είναι το απλούστερο νευρωνικό δίκτυο που χρησιμοποιείται για την ταξινόμηση γραμμικά διαχωρίσιμων προτύπων, δηλαδή προτύπων τα οποία διαχωρίζονται από ένα υπερεπίπεδο.

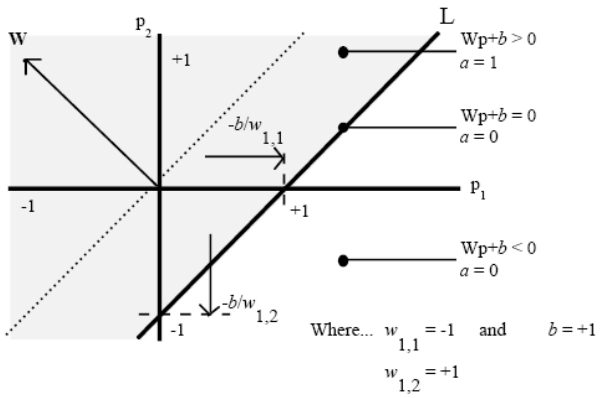
### 4.8.1 Νευρωνικό δίκτυο Perceptron ενός επιπέδου

Το νευρωνικό δίκτυο perceptron ενός επιπέδου αποτελείται από έναν απλό νευρώνα με προσαρμόσιμα βάρη, ο οποίος ακολουθείται από μια διπολική συνάρτηση ενεργοποίησης. Η δομή του perceptron ενός επιπέδου φαίνεται στο Σχήμα 24.




Σχήμα 24: Νευρωνικό δίκτυο perceptron ενός επιπέδου και ενός νευρώνα

Αυτό το δίκτυο μπορεί να πραγματοποιήσει ταξινόμηση στην περίπτωση που έχουμε μόνο δύο κατηγορίες εξόδου. Το δίκτυο παράγει ως έξοδο τη μονάδα, αν η είσοδος της συνάρτησης μεταφοράς είναι ίση ή μεγαλύτερη από το μηδέν. Σε κάθε άλλη περίπτωση, παράγει στην έξοδο το μηδέν. Για να γίνει κατανοητός ο τρόπος λειτουργίας ενός ταξινομητή προτύπων, θεωρείται η περίπτωση δύο μεταβλητών $p_1$ και $p_2$ όπου το σύνορο απόφασης (διαχωρισμός ανάμεσα στις κατηγορίες $L_1$ και $L_2$) είναι μια ευθεία γραμμή L όπως δείχνει το Σχήμα 25, όπου το κατώφλι b απλά μεταθέτει το σύνορο διαχωρισμού .



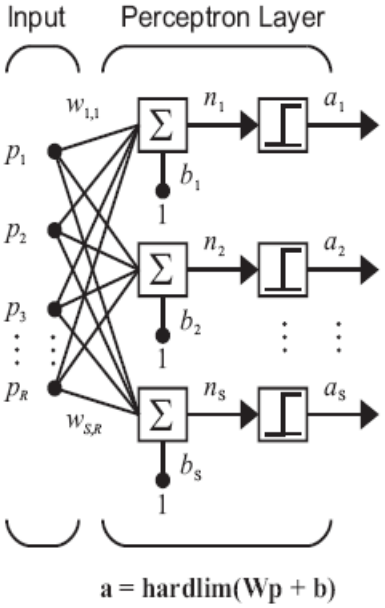

Σχήμα 25: Γραμμικός διαχωρισμός δύο κλάσεων L1 και L2 στην περίπτωση δύο διαστάσεων.

Οποιοδήποτε σημείο βρίσκεται πάνω από τη διαχωριστική γραμμή L ταξινομείται στην κατηγορία $L_1$, διαφορετικά αποδίδεται στην κατηγορία $L_2$. Ο αλγόριθμος επιλογής (ανανέωσης) των βαρών (αλγόριθμος perceptron) στηρίζεται στο γεγονός ότι εάν οι κλάσεις $L_1$ και $L_2$ είναι γραμμικά διαχωρίσιμες, τότε υπάρχει ένα διάνυσμα βαρών w τέτοιο ώστε

- $w^T p \geq 0$ , όταν το p ανήκει στην $L_1$
- $w^T p < 0$ , όταν το p ανήκει στην $L_2$

Συνεπώς όταν το perceptron λαβαίνει ένα ζευγάρι εκπαίδευσης ($p_1$, $p_2$) πρέπει να προσδιορίσει ένα διάνυσμα βάρους w τέτοιο ώστε να ικανοποιούνται οι δύο παραπάνω ανισότητες.

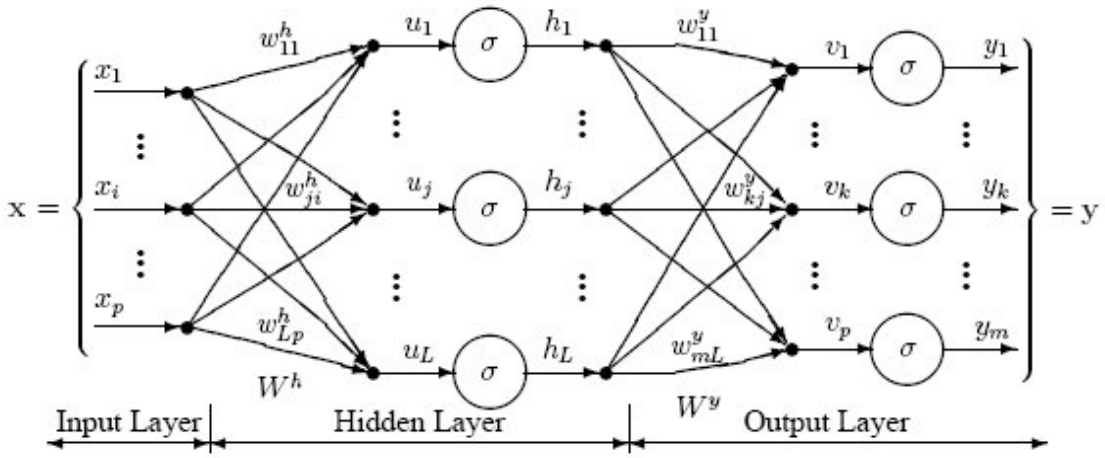
Για να είναι δυνατή η ταξινόμηση περισσοτέρων γραμμικά διαχωρίσιμων κατηγοριών (κλάσεων) το επίπεδο εξόδου χρειάζεται να έχει περισσότερους από έναν νευρώνες όπως φαίνεται στο Σχήμα 26.




Σχήμα 26: Νευρωνικό δίκτυο perceptron ενός επιπέδου και πολλών νευρώνων

### 4.8.2 Νευρωνικό δίκτυο Perceptron πολλών επιπέδων

Τα πολυεπίπεδα δίκτυα perceptron περιλαμβάνουν, εκτός από το επίπεδο κόμβων εισόδου (πηγής) και το επίπεδο εξόδου (αποκρίσεων), ένα ή περισσότερα επίπεδα κρυμμένων κόμβων όπως δείχνει το Σχήμα 27.




Σχήμα 27: Πολυεπίπεδο perceptron με δύο κρυφά επίπεδα.

Τα πολυεπίπεδα δίκτυα perceptron είναι κατάλληλα για την επίλυση πλήθους πολύπλοκων προβλημάτων και εκπαιδεύονται με επιβλεπόμενη μάθηση χρησιμοποιώντας τον αλγόριθμο οπισθοδρόμησης (Back Propagation) ο οποίος στηρίζεται στον κανόνα μάθησης διόρθωσης σφάλματος.

Σε αντίθεση με τη διπολική μη γραμμική συνάρτηση ενεργοποίησης, που χρησιμοποιείται στο απλό perceptron ενός επιπέδου του Rosenblatt, εδώ χρησιμοποιείται μια παραγωγίσιμη σιγμοειδής συνάρτηση ενεργοποίησης. Συνήθως, τα πολυεπίπεδα δίκτυα perceptron έχουν υψηλή συνδεσιμότητα, δηλαδή έχουν μεγάλους αριθμούς συναπτικών βαρών (διασυνδέσεων).

Ο αλγόριθμος οπισθοδρόμησης περιλαμβάνει τη «φάση προς τα εμπρός» (ορθή φάση) και τη «φάση προς τα πίσω» (ανάστροφη φάση). Κατά τη «φάση προς τα εμπρός» ένα πρότυπο εισόδου οδηγείται προς την έξοδο και όταν φθάσει σ' αυτή παράγει τις πραγματικές εξόδους. Κατά τη «φάση προς τα εμπρός» όλα τα συναπτικά βάρη κρατούν σταθερές τιμές. Κατά τη «φάση προς τα πίσω» το σφάλμα εξόδου διαδίδεται ανάστροφα από την έξοδο προς την είσοδο και τα συναπτικά βάρη όλων των νευρώνων ανανεώνονται μέσω του κανόνα διόρθωσης σφάλματος, έτσι ώστε η πραγματική έξοδος του perceptron να πλησιάσει όσο γίνεται πιο πολύ την επιθυμητή έξοδο του. Συνεπώς, ο όρος οπισθοδρόμηση οφείλεται στον τρόπο με τον οποίο διαδίδονται τα σφάλματα μέσω των επιπέδων του νευρωνικού δικτύου.

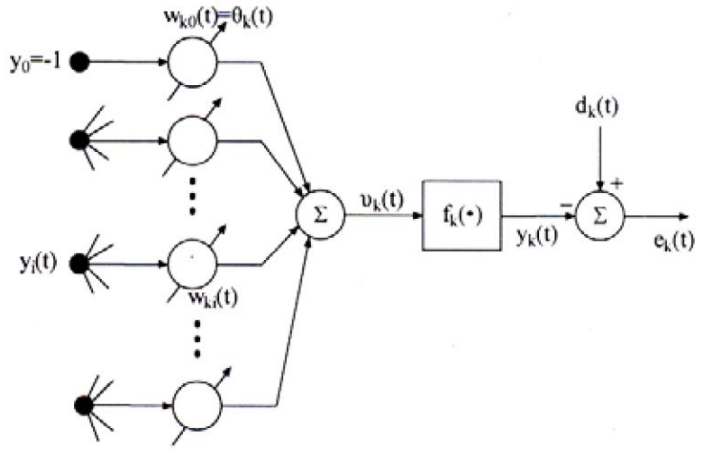
Στο σημείο αυτό, πρέπει να σημειωθεί ότι και ένα μόνο κρυφό επίπεδο αρκεί για να εκτελέσει το νευρωνικό δίκτυο λειτουργίες που μπορεί να κάνει με πολλά κρυμμένα επίπεδα. Αυτό οφείλεται στο «θεώρημα της προσέγγισης» (Approximation Theorem) του Kolmogorov [47] το οποίο λέει τα εξής:

«Δοθείσης οποιασδήποτε συνεχούς συνάρτησης $F:[0,1]^n \to \Re^m, F(x)=y$, όπου $I=[0,1]^n$ είναι ο μοναδιαίος *n*-διάστατος κύβος, η F μπορεί να πραγματοποιηθεί (προσεγγισθεί) επακριβώς με ένα perceptron (Νευρωνικό δίκτυο προσοτροφοδότησης) τριών επιπέδων που έχει *n* κόμβους στο επίπεδο εισόδου (πηγής), *2n+1* κόμβους στο μεσαίο (κρυφό) επίπεδο και *m* κόμβους στο επίπεδο εξόδου»

## 4.9 Αλγόριθμος οπισθοδρόμησης (Back-Propagation)

Ο αλγόριθμος της οπισθοδρόμησης [48] είναι μια συχνά χρησιμοποιούμενη μέθοδος εκμάθησης των τεχνητών νευρωνικών δικτύων στο να εκτελούν μια δεδομένη εργασία. Ο αλγόριθμος αυτός, αρχικά, περιγράφηκε από τους Arthur E. Bryson και Yu-Chi Ho το 1969 αλλά με τις εργασίες των David E. Rumelhart, Geoffrey E. Hinton και Ronald J.Williams το 1986 απέκτησε αναγνώριση και οδήγησε στην ανάπτυξη των τεχνητών νευρωνικών δικτύων.

Ο αλγόριθμος αυτός χρησιμοποιείται στα πρότυπα της επιβλεπόμενης μάθησης και είναι μια επέκταση του κανόνα δέλτα, που χρησιμοποιείται για την αναπροσαρμογή των συναπτικών βαρών των τεχνητών νευρώνων σε ένα δίκτυο perceptron ενός επιπέδου.




Σχήμα 28: Αναπαράσταση εσωτερικής δραστηριότητας ενός νευρώνα.

Από το Σχήμα 28 και αν θεωρηθεί ότι το σήμα της εσωτερικής δραστηριότητας του k νευρώνα σε μια χρονική στιγμή t είναι $u_k(t)$ τότε θα ισχύει:

$$u_k(t) = \sum_{i=0}^{n} w_{ki}(t)\, x_i(t)$$

Συνεπώς αν ο νευρώνας έχει μια συνάρτηση ενεργοποίησης f η έξοδός του θα δίνεται από τον τύπο:

$$y_k(t) = f\left(u_k(t)\right)$$

Το σφάλμα διόρθωσης που προκύπτει, σύμφωνα με την μέθοδο διόρθωσης σφάλματος, από τη διαφορά της πραγματικής τιμής της εξόδου $y_k(t)$ του νευρώνα από την επιθυμητή τιμή $d_k(t)$, δίνεται από τον τύπο:

$$e_k(t) = d_k(t) - y_k(t) \;\Rightarrow\; e_k(t) = d_k(t) - f\left(u_k(t)\right)$$

Αν θεωρηθεί ως μέθοδος αναπροσαρμογής των συναπτικών βαρών η μέθοδος διόρθωσης σφάλματος τότε οι διορθώσεις των συναπτικών βαρών $\Delta w_{ki}(t)$ δίνονται από τον τύπο της πιο απότομης κατάβασης (steepest descent) και είναι:

$$\Delta w_{ki}(t) = -\gamma \frac{\partial E_p(t)}{\partial w_{ki}(t)}$$

όπου $E_p(t) = \frac{1}{2}\sum_{i=0}^{n}\left(d_i^p - y_i^p\right)^2 = \frac{1}{2}\sum_{i=0}^{n}\left(e_i^p\right)^2$ το συνολικό τετραγωνικό σφάλμα για την πρότυπη είσοδο p στη έξοδο του νευρώνα k και γ η παράμετρος μάθησης.

Για να προσδιορισθούν οι νέες τιμές των βαρών πρέπει να υπολογιστεί η μερική παράγωγος – κλίση της συνάρτησης του συνολικού τετραγωνικού σφάλματος. Σύμφωνα με τον κανόνα της αλυσίδας ισχύει:

$$\frac{\partial E_p(t)}{\partial w_{ki}(t)} = \frac{\partial E_p(t)}{\partial e_k(t)} \cdot \frac{\partial e_k(t)}{\partial y_k(t)} \cdot \frac{\partial y_k(t)}{\partial u_k(t)} \cdot \frac{\partial u_k(t)}{\partial w_{ki}(t)} = e_k(t) \cdot (-1) \cdot \frac{df_k\left(u_k(t)\right)}{du_k(t)} \cdot y_i(t) \Rightarrow$$

$$\frac{\partial E_p(t)}{\partial w_{ki}(t)} = -e_k(t) \cdot f_k'\left(u_k(t)\right) \cdot y_i(t)$$

Συνδυάζοντας τον παραπάνω τύπο με τον κανόνα της πιο απότομης μετάβασης προκύπτει ο γενικευμένος κανόνας δέλτα:

$$\Delta w_{ki}(t) = \gamma \cdot e_k(t) \cdot f_k'(u_k(t)) \cdot y_i(t) \;\Rightarrow\; \Delta w_{ki}(t) = \gamma \cdot \delta_k(t) \cdot y_i(t)$$

όπου $\delta_k(t) = e_k(t) \cdot f_k'(u_k(t))$ είναι η τοπική κλίση του αντίστοιχου νευρώνα.

Η παραπάνω τοπική κλίση $\delta_k(t)$ αναφέρεται σε έναν νευρώνα εξόδου. Το ίδιο πρέπει να γίνει και για τους νευρώνες των κρυφών επιπέδων. Ωστόσο, για τους νευρώνες j των κρυμμένων επιπέδων δεν υπάρχουν συγκεκριμένες επιθυμητές έξοδοι και συνεπώς τα αντίστοιχα σφάλματα πρέπει να υπολογισθούν έμμεσα χρησιμοποιώντας τα σφάλματα όλων των άλλων νευρώνων με τους οποίους συνδέονται. Συνεπώς η τοπική κλίση των νευρώνων j των κρυμμένων επιπέδων είναι:

$$\delta_j(t) = -\frac{\partial E_p(t)}{\partial y_j(t)} \cdot \frac{\partial y_j(t)}{\partial u_j(t)} \;\Rightarrow\; \delta_j(t) = -\frac{\partial E_p(t)}{\partial y_j(t)} \cdot f_j'(u_j(t))$$

Σύμφωνα με τις παρακάτω σχέσεις:

$$E_p(t) = \frac{1}{2}\sum_m (d_m(t) - y_m(t))^2 = \frac{1}{2}\sum_m (e_m(t))^2,\; u_m(t) = \sum_{j=0}^{n} w_{mj}(t)\, y_i(t),\; y_m(t) = f_m(u_m(t))$$

όπου m ο δείκτης για τους νευρώνες εξόδου, προκύπτει ότι η τοπική κλίση είναι:

$$\delta_j(t) = -\frac{\partial E_p(t)}{\partial y_j(t)} \cdot f_j'(u_j(t)) = f_j'(u_j(t)) \cdot \sum_m e_m(t) \cdot \frac{\partial e_m(t)}{\partial y_j(t)}$$

$$\delta_j(t) = f_j'(u_j(t)) \cdot \sum_m e_m(t) \cdot \frac{\partial e_m(t)}{\partial u_m(t)} \cdot \frac{\partial u_m(t)}{\partial y_j(t)} \;\Rightarrow$$

$$\delta_j(t) = f_j'(u_j(t)) \cdot \sum_m e_m(t) \cdot f_m'(u_m(t)) \cdot w_{mj}(t)$$

Σύμφωνα όμως με την εξίσωση της τοπικής κλίσης για τους νευρώνες εξόδου ισχύει ότι:

$$\delta_j(t) = f_j'(u_j(t)) \cdot \sum_m e_m(t) \cdot f_m'(u_m(t)) \cdot w_{mj}(t) \;\Rightarrow$$

$$\delta_j(t) = f_j'(u_j(t)) \cdot \sum_m \delta_m(t) \cdot w_{mj}(t)$$

Σύμφωνα, λοιπόν, με τα παραπάνω, τα βήματα του αλγορίθμου οπισθοδρόμησης είναι:

Βήμα 1ο
Επιλογή των αρχικών συναπτικών βαρών, τα οποία, συνήθως, είναι μικρές τυχαίες θετικές τιμές.

Βήμα 2ο
Εισαγωγή στο νευρωνικό δίκτυο του διανύσματος εκπαίδευσης
$x(t) = [x_0(t), x_1(t), ..., x_n(t)]$
και του επιθυμητού διανύσματος εξόδου
$d(t) = [d_0(t), d_1(t), ..., d_m(t)]$

Βήμα 3ο
Υπολογισμός των σημάτων εξόδου όλων των νευρώνων του δικτύου διατρέχοντας το δίκτυο νευρώνα-νευρώνα προς τα εμπρός χρησιμοποιώντας τις τρέχουσες τιμές των συναπτικών βαρών και κάνοντας χρήση των σχέσεων:

$$y_j(t) = f_j\left(u_j(t)\right) \ , \ u_j(t) = \sum_{i=0}^{n} w_{ji}(t)\, y_i(t)$$

όπου $u_j(t)$ η είσοδος του τρέχοντος νευρώνα, $y_i(t)$ η έξοδος όλων των νευρώνων που συνδέονται με τον τρέχοντα νευρώνα, $w_{ji}(t)$ τα συναπτικά βάρη των αντίστοιχων νευρώνων και $y_j(t)$ η έξοδος του νευρώνα.

Βήμα 4ο
Αναπροσαρμογή των συναπτικών βαρών διατρέχοντας το δίκτυο νευρώνα-νευρώνα προς τα πίσω (οπισθοδρόμηση) χρησιμοποιώντας την μέθοδο διόρθωσης λαθών και τον κανόνα δέλτα. Τα νέα συναπτικά βάρη προκύπτουν από τη σχέση:

$$w_{ji}(t+1) = w_{ji}(t) + \gamma \cdot \delta_j(t) \cdot y_i(t)$$

όπου γ η παράμετρος μάθησης και

$$\delta_j(t) = \begin{cases} f_j'\left(u_j(t)\right) \cdot \sum_m \delta_m(t) \cdot w_{mj}(t) & , \ \textit{για νευρώνα κρυφού επιπέδου} \\ e_j(t) \cdot f_j'\left(u_j(t)\right) & , \ \textit{για νευρώνα εξόδου} \end{cases}$$

## 4.10 Αλγόριθμος Levenberg – Marquardt

Ο αλγόριθμος Levenberg – Marquardt είναι μια παραλλαγή του αλγορίθμου οπισθοδρόμησης. Η μαθηματική μορφή μιας επανάληψης του αλγορίθμου αυτού είναι η εξής:

$$w_{k+1} = w_k - \left(H + \mu \cdot I\right) \cdot \delta$$

όπου w είναι ο πίνακας συναπτικών βαρών του δικτύου, Η η μήτρα παραγώγων (Hessian Matrix), I ο μοναδιαίος πίνακας, δ η κλίση και μ ένας συντελεστής που καθορίζει το μέγεθος κάθε βήματος.

Η μήτρα Η και η κλίση δ μπορούν να προσεγγισθούν από τις εξής σχέσεις:

$$H = J^T J \ , \ \delta = J^T e$$

όπου J η Ιακωβιανή μήτρα που περιέχει τις πρώτες παραγώγους της συνάρτησης σφάλματος ως προς τα βάρη του δικτύου και e ο πίνακας των σφαλμάτων του δικτύου.

Έτσι, η επανάληψη του αλγορίθμου παίρνει την μορφή:

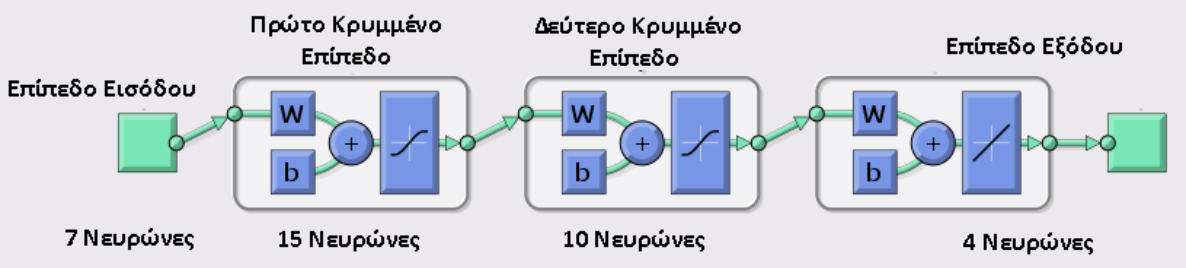
$$w_{k+1} = w_k - \left(J^T J + \mu \cdot I\right) \cdot J^T e$$

Ο αλγόριθμος Levenberg – Marquardt είναι από 10 έως 100 φορές ταχύτερος αλλά μπορεί να απαιτήσει μεγαλύτερη ποσότητα μνήμης [49].

## 4.11 Το Νευρωνικό Δίκτυο που χρησιμοποιήθηκε

Στην παρούσα εργασία υλοποιήθηκε ένα πολυστρωματικό νευρωνικό δίκτυο προσοτροφοδότησης με το Neural Network Toolbox του MATLAB. Αρχικά, δοκιμάστηκαν διάφορα μεγέθη νευρωνικού δικτύου (αριθμός νευρώνων, πλήθος κρυμμένων στρωμάτων) και οι δοκιμές έδειξαν ότι ένα κρυμμένο στρώμα δεν αρκεί για την εύρεση της αντιστοιχίας μεταξύ εισόδων και εξόδων. Έτσι, ύστερα από τις δοκιμές βρέθηκε ότι το βέλτιστο νευρωνικό δίκτυο για την συγκεκριμένη εργασία φαίνεται στο Σχήμα 29 και αποτελείται από 7 νευρώνες εισόδου, 15 νευρώνες στο πρώτο κρυμμένο επίπεδο, 10 νευρώνες στο δεύτερο κρυμμένο επίπεδο και 4 νευρώνες στο επίπεδο εξόδου.

Στα κρυμμένα επίπεδα χρησιμοποιήθηκε εφαπτομενική – σιγμοειδής συνάρτηση ενεργοποίησης και στο επίπεδο εξόδου γραμμική συνάρτηση ενεργοποίησης. Οι είσοδοι και οι έξοδοι κανονικοποιήθηκαν έτσι ώστε να κυμαίνονται στο διάστημα [-1,+1]. Για την αναπροσαρμογή των συναπτικών βαρών χρησιμοποιήθηκε ο αλγόριθμος Levenberg – Marquardt και για τον έλεγχο της επίδοσης του νευρωνικού δικτύου το μέσο τετραγωνικό σφάλμα.




**Σχήμα 29: Το νευρωνικό δίκτυο που χρησιμοποιήθηκε στην συγκεκριμένη εργασία**

Οι κώδικες που δημιουργήθηκαν στο περιβάλλον του MATLAB και που υλοποιούν αυτό το δίκτυο δίνονται στο παράρτημα Γ.

# ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΣΟΜΟΙΩΣΗΣ ΣΥΝΟΛΩΝ ΕΛΕΓΧΟΥ

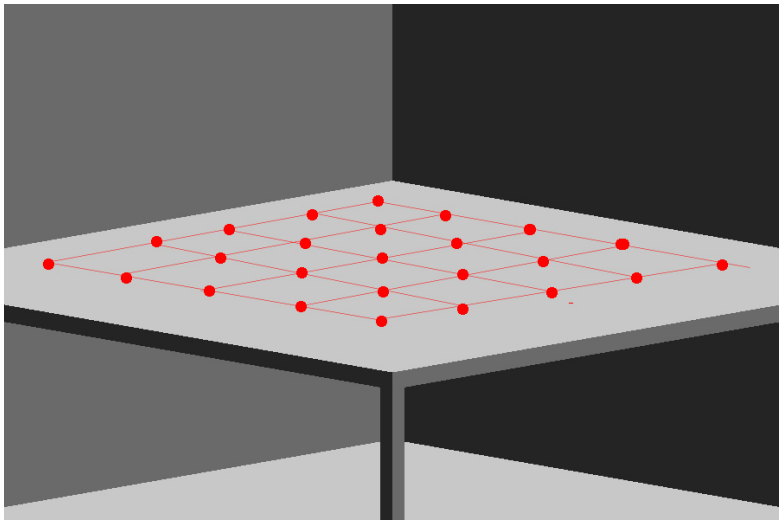
Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, δημιουργήθηκε ένα νευρωνικό δίκτυο που αποτελείται από 7 νευρώνες εισόδου, 15 νευρώνες στο πρώτο κρυμμένο επίπεδο, 10 νευρώνες στο δεύτερο κρυμμένο επίπεδο και 4 νευρώνες στο επίπεδο εξόδου.

Στον Πίνακα 5 παρουσιάζονται τα μήκη των ορθογώνιων παραλληλεπιπέδων για το ύψος *h* (y άξονας) και για το μήκος *z* στον άξονα z, που χρησιμοποιήθηκαν για την εκπαίδευση του ρομπότ.

| Ύψος αντικειμένου (m) | Μήκη αντικειμένων στον άξονα z (m) | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.025 | 0.03 | 0.05 | 0.07 | 0.09 | 0.12 | 0.16 | 0.20 |
| 0.05 | 0.03 | 0.05 | 0.07 | 0.09 | 0.12 | 0.16 | 0.20 |
| 0.07 | 0.03 | 0.05 | 0.07 | 0.09 | 0.12 | 0.16 | 0.20 |
| 0.09 | 0.03 | 0.05 | 0.07 | 0.09 | 0.12 | 0.16 | 0.20 |
| 0.12 | 0.03 | 0.05 | 0.07 | 0.10 | 0.12 | 0.16 | 0.20 |
| 0.16 | 0.03 | 0.06 | 0.09 | 0.12 | 0.14 | 0.16 | 0.20 |
| 0.20 | 0.03 | 0.06 | 0.09 | 0.12 | 0.15 | 0.18 | 0.20 |

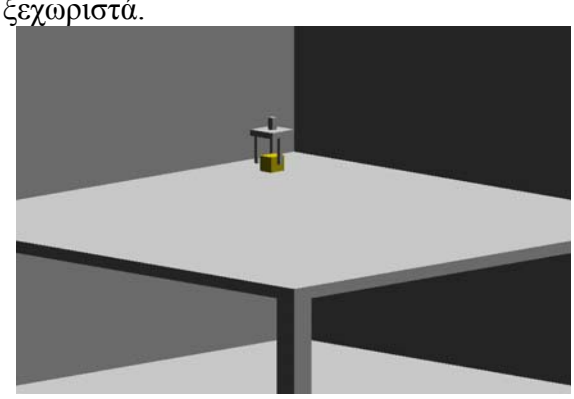
**Πίνακας 5: Πίνακας μηκών χειριζόμενων αντικειμένων εκπαίδευσης**

Η εκμάθηση του ρομπότ έγινε σε ένα πλαίσιο 5x5 όπως φαίνεται στο Σχήμα 30.

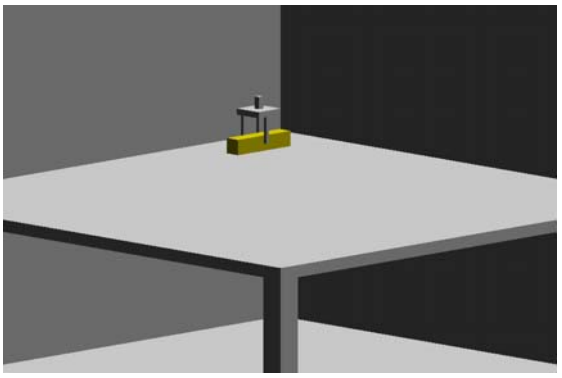


**Σχήμα 30: Πλαίσιο εκπαίδευσης ρομποτικής αρπάγης**

Στα παρακάτω σχήματα εικονίζονται όλα τα αντικείμενα που χρησιμοποιήθηκαν για την εκπαίδευση του ρομπότ καθώς και η ρομποτική αρπάγη για κάθε αντικείμενο ξεχωριστά.

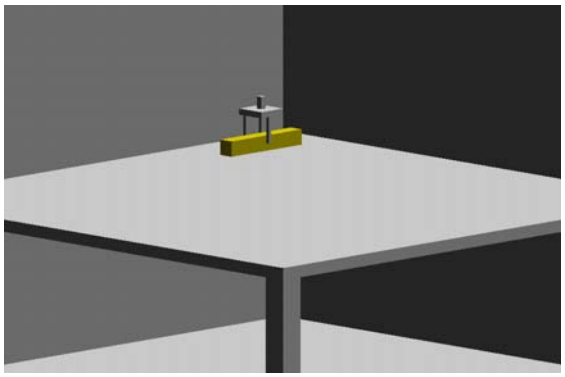


**Σχήμα 31: Αντικείμενο με h=0.025m και z=0.03m**

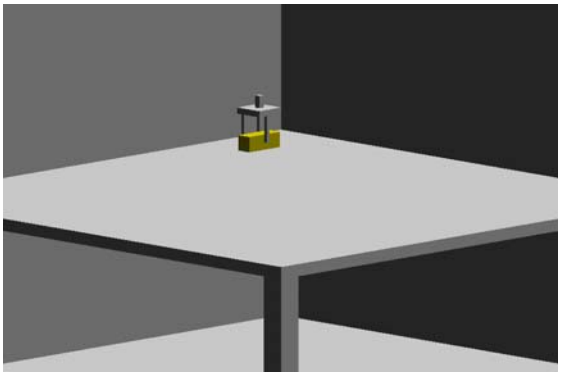


**Σχήμα 35: Αντικείμενο με h=0.025m και z=0.12m**

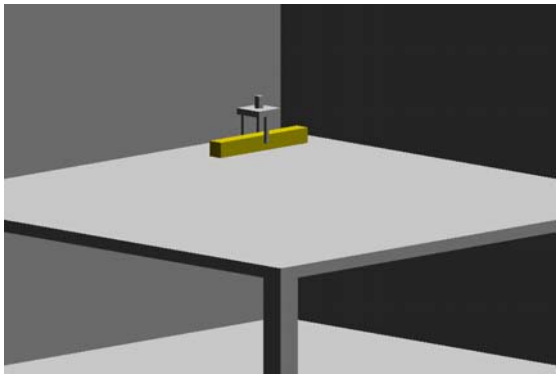
**Σχήμα 32: Αντικείμενο με h=0.025m και z=0.05m**



**Σχήμα 36: Αντικείμενο με h=0.025m και z=0.16m**

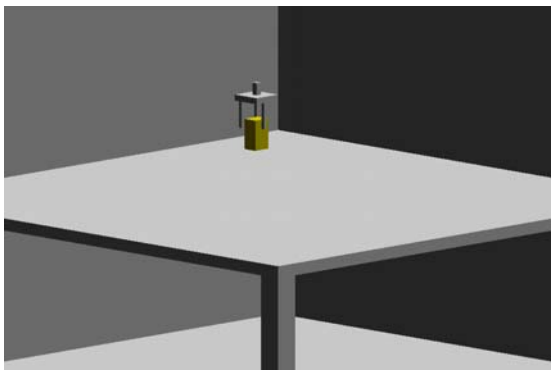


**Σχήμα 33: Αντικείμενο με h=0.025m και z=0.07m**

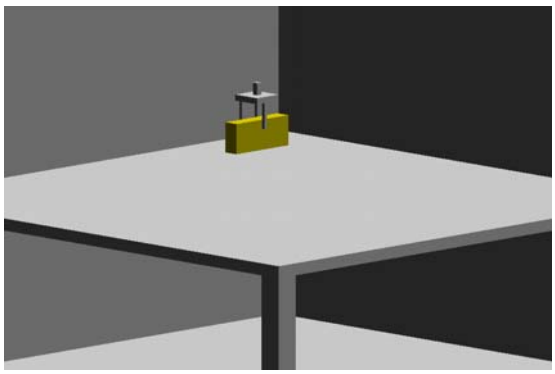


**Σχήμα 37: Αντικείμενο με h=0.025m και z=0.20m**

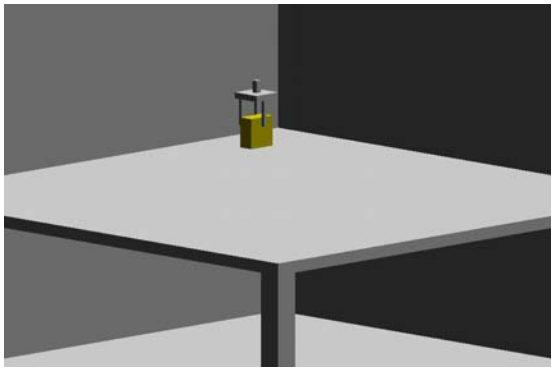
**Σχήμα 34: Αντικείμενο με h=0.025m και z=0.09m**



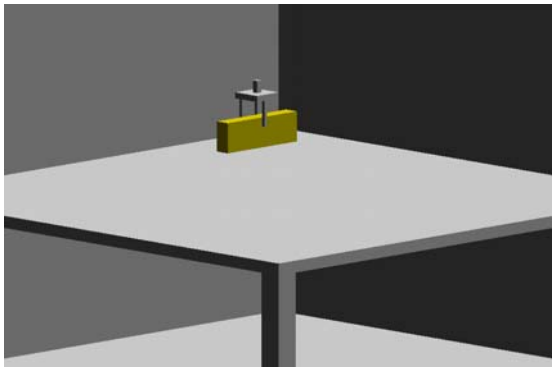
**Σχήμα 38: Αντικείμενο με h=0.05m και z=0.03m**



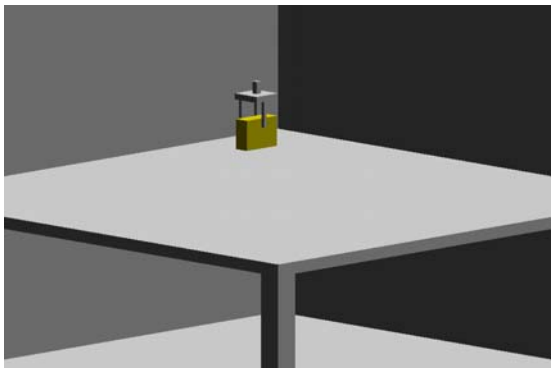
**Σχήμα 42: Αντικείμενο με h=0.05m και z=0.12m**



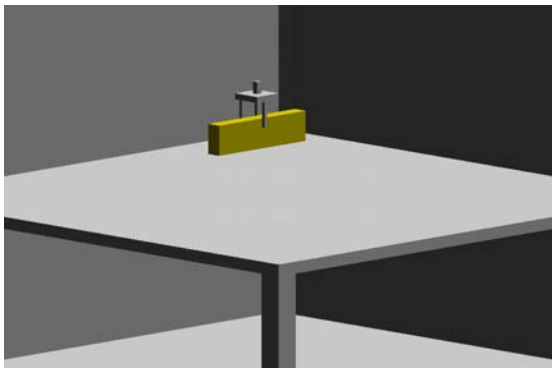
**Σχήμα 39: Αντικείμενο με h=0.05m και z=0.05m**



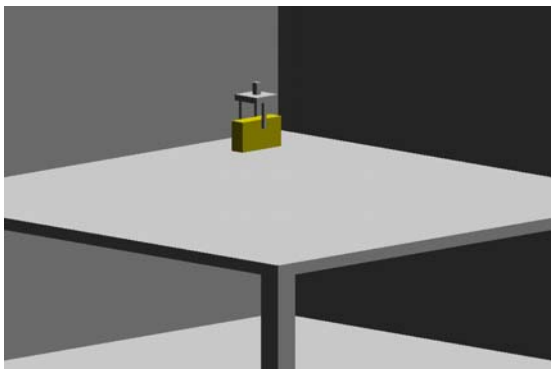
**Σχήμα 43: Αντικείμενο με h=0.05m και z=0.16m**



**Σχήμα 40: Αντικείμενο με h=0.05m και z=0.07m**



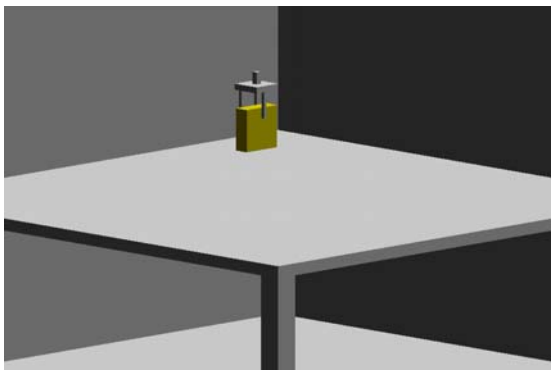
**Σχήμα 44: Αντικείμενο με h=0.05m και z=0.20m**



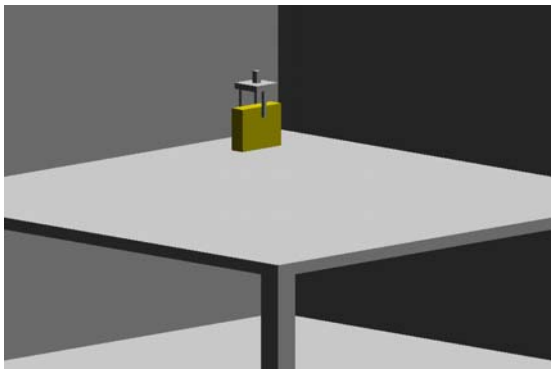
**Σχήμα 41: Αντικείμενο με h=0.05m και z=0.09m**

**Σχήμα 45: Αντικείμενο με h=0.07m και z=0.03m**

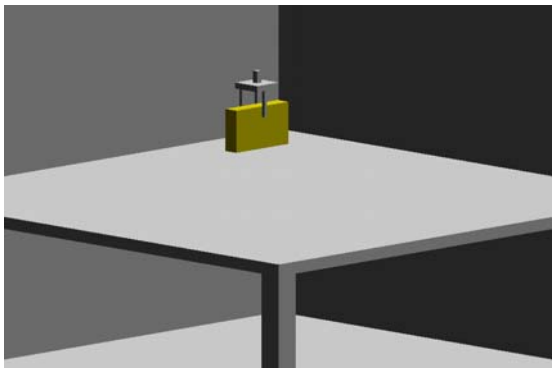
**Σχήμα 46: Αντικείμενο με h=0.07m και z=0.05m**



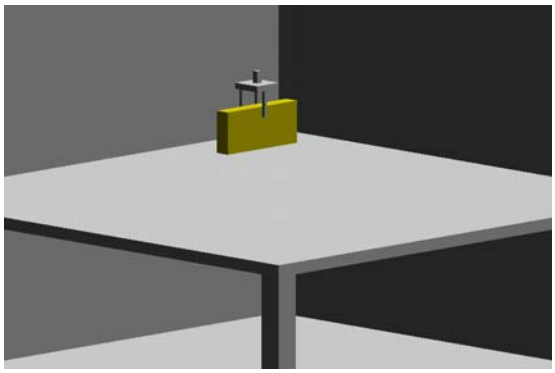
**Σχήμα 47: Αντικείμενο με h=0.07m και z=0.07m**



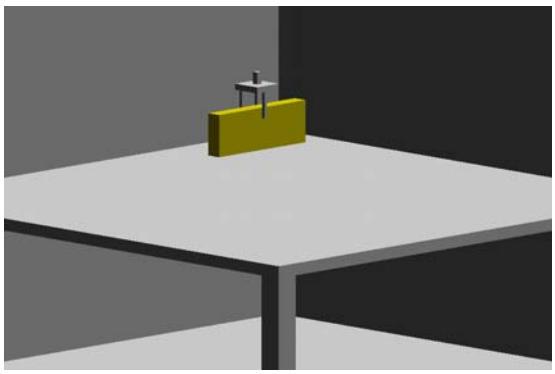
**Σχήμα 48: Αντικείμενο με h=0.07m και z=0.09m**



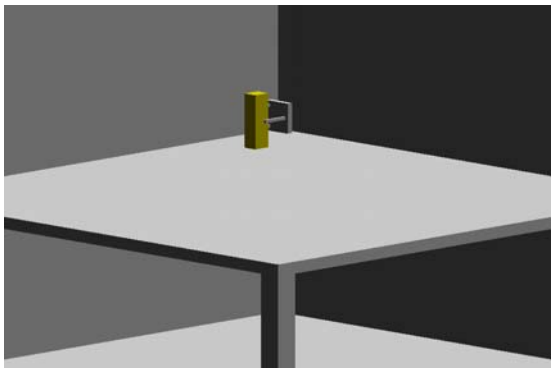
**Σχήμα 49: Αντικείμενο με h=0.07m και z=0.12m**



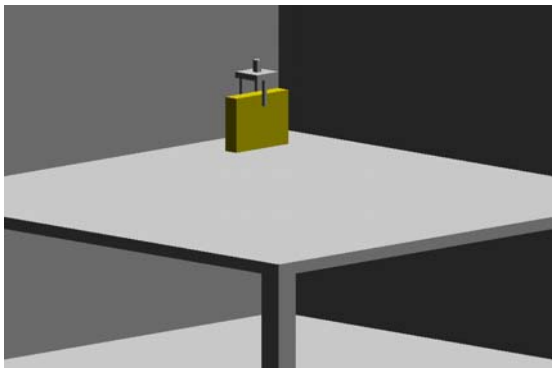
**Σχήμα 50: Αντικείμενο με h=0.07m και z=0.16m**



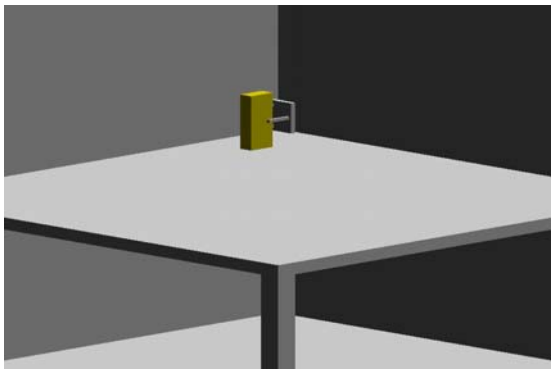
**Σχήμα 51: Αντικείμενο με h=0.07m και z=0.20m**



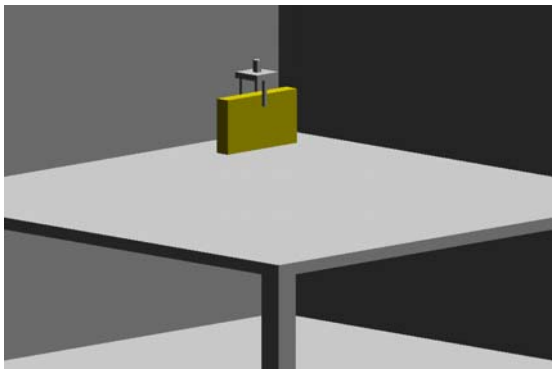
**Σχήμα 52: Αντικείμενο με h=0.09m και z=0.03m**



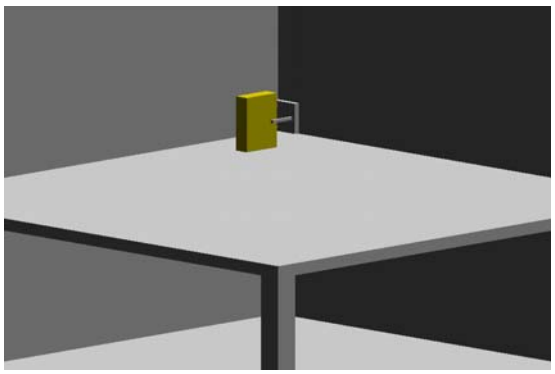
**Σχήμα 56: Αντικείμενο με h=0.09m και z=0.12m**



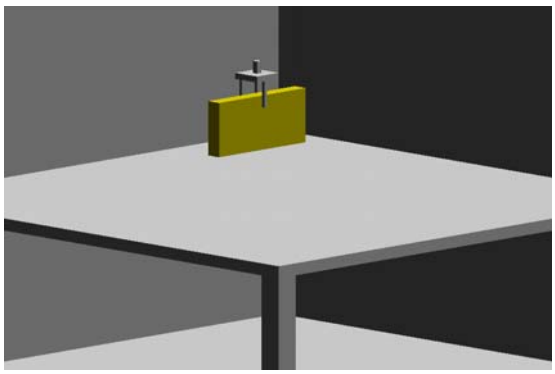
**Σχήμα 53: Αντικείμενο με h=0.09m και z=0.05m**



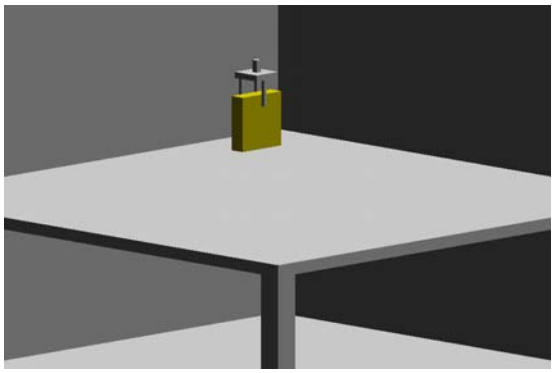
**Σχήμα 57: Αντικείμενο με h=0.09m και z=0.16m**



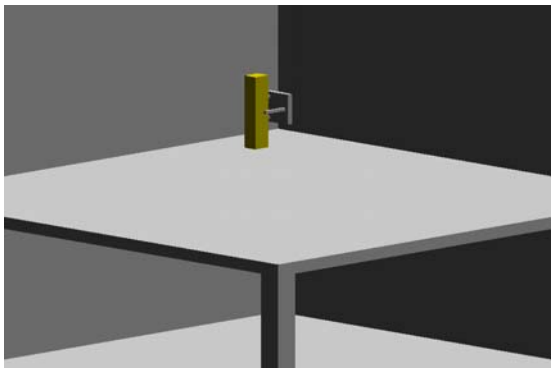
**Σχήμα 54: Αντικείμενο με h=0.09m και z=0.07m**



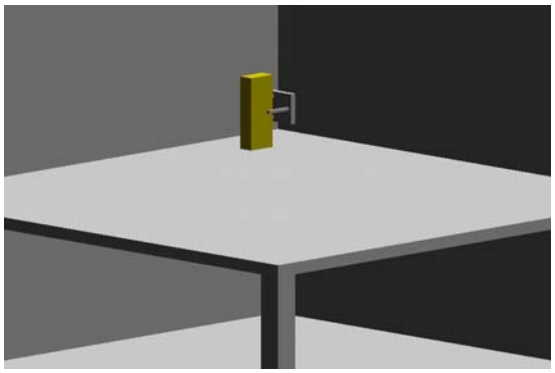
**Σχήμα 58: Αντικείμενο με h=0.09m και z=0.20m**



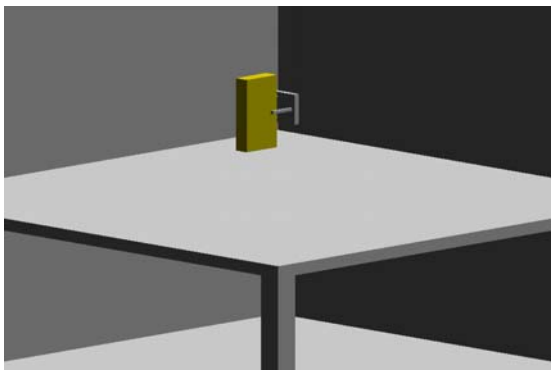
**Σχήμα 55: Αντικείμενο με h=0.09m και z=0.09m**



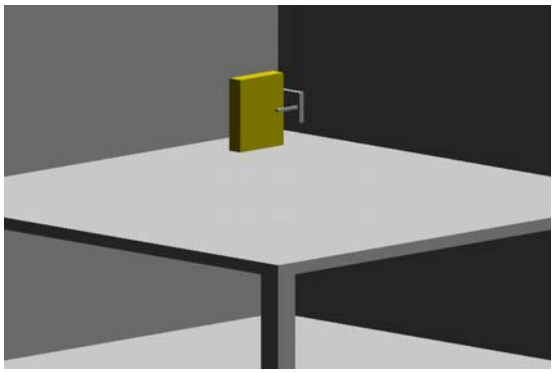
**Σχήμα 59: Αντικείμενο με h=0.12m και z=0.03m**



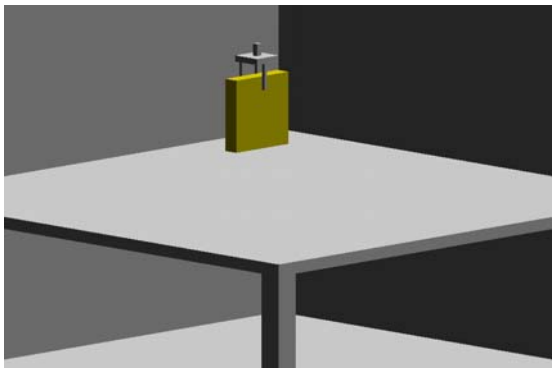
**Σχήμα 60: Αντικείμενο με h=0.12m και z=0.05m**



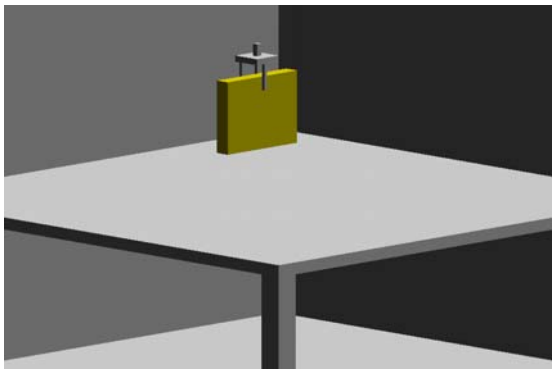
**Σχήμα 61: Αντικείμενο με h=0.12m και z=0.07m**



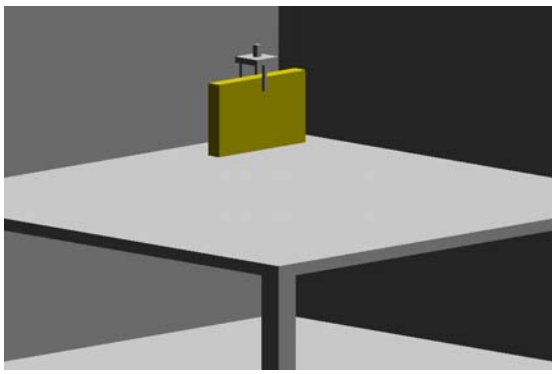
**Σχήμα 62: Αντικείμενο με h=0.12m και z=0.10m**



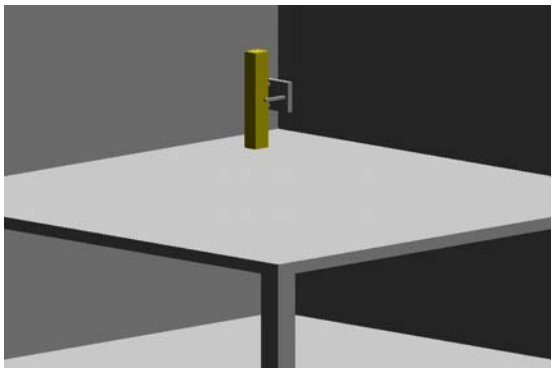
**Σχήμα 63: Αντικείμενο με h=0.12m και z=0.12m**



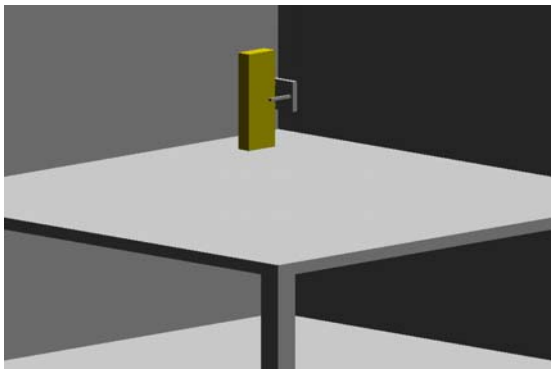
**Σχήμα 64: Αντικείμενο με h=0.12m και z=0.16m**



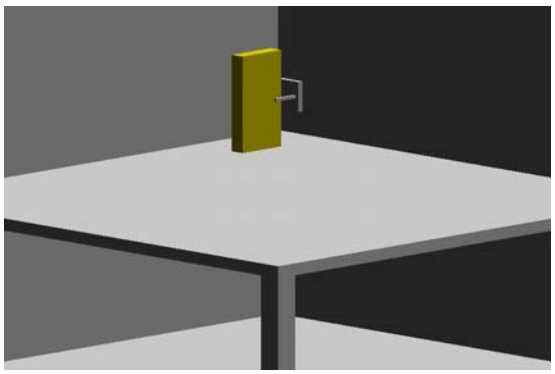
**Σχήμα 65: Αντικείμενο με h=0.12m και z=0.20m**



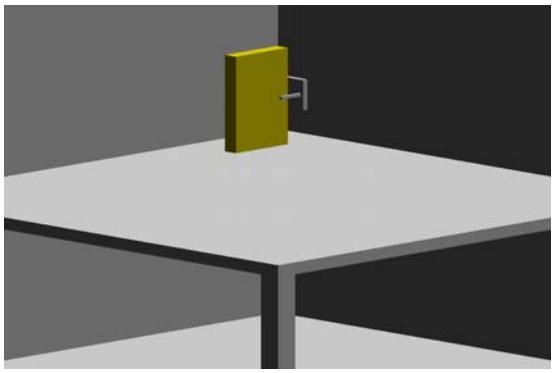
**Σχήμα 66: Αντικείμενο με h=0.16m και z=0.03m**



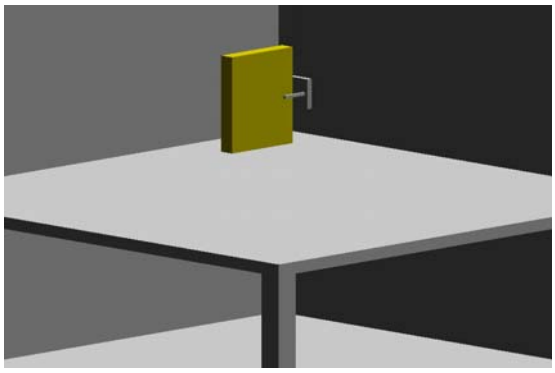
**Σχήμα 67: Αντικείμενο με h=0.16m και z=0.06m**



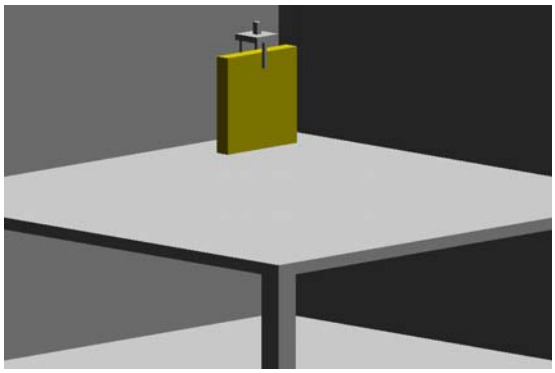
**Σχήμα 68: Αντικείμενο με h=0.16m και z=0.09m**



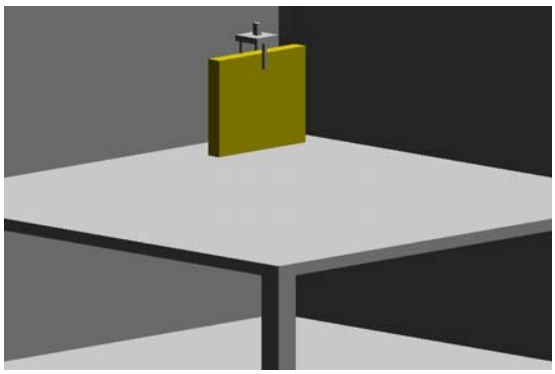
**Σχήμα 69: Αντικείμενο με h=0.16m και z=0.12m**



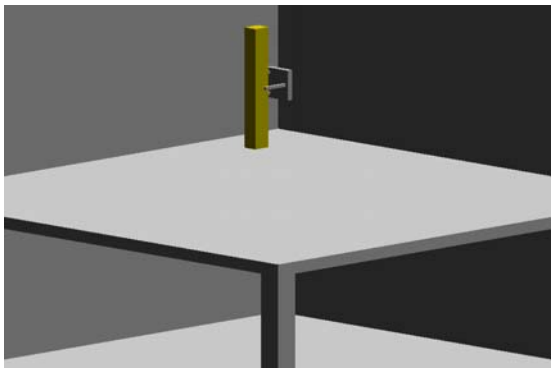
**Σχήμα 70: Αντικείμενο με h=0.16m και z=0.14m**



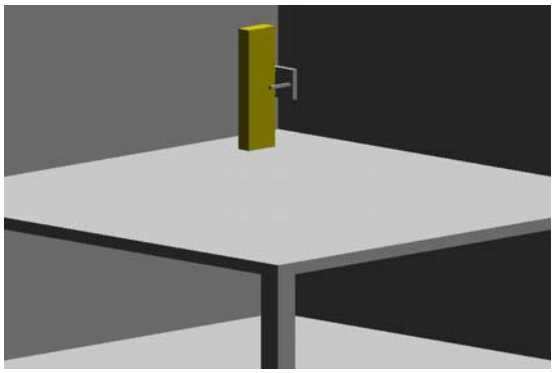
**Σχήμα 71: Αντικείμενο με h=0.16m και z=0.16m**



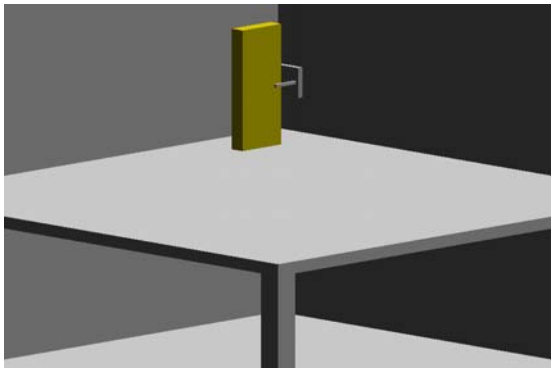
**Σχήμα 72: Αντικείμενο με h=0.16m και z=0.20m**



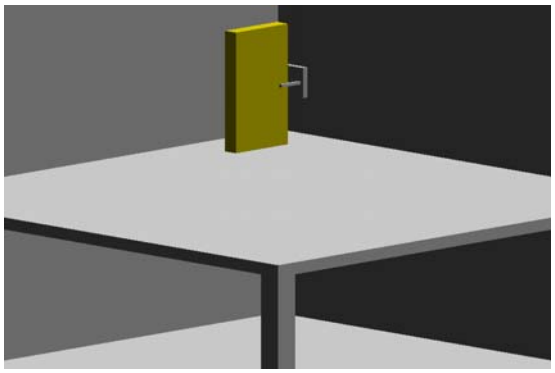
**Σχήμα 73: Αντικείμενο με h=0.20m και z=0.03m**



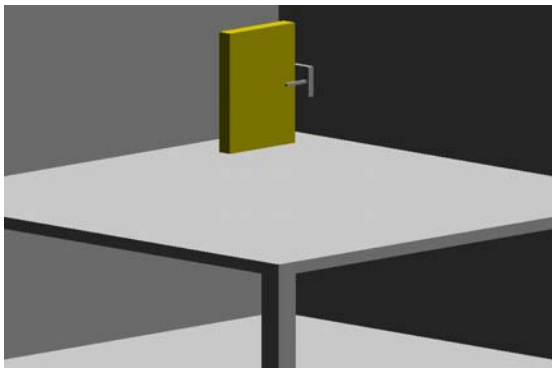
**Σχήμα 74: Αντικείμενο με h=0.20m και z=0.06m**



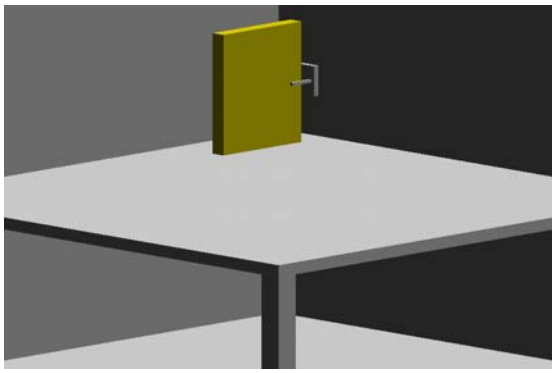
**Σχήμα 75: Αντικείμενο με h=0.20m και z=0.09m**



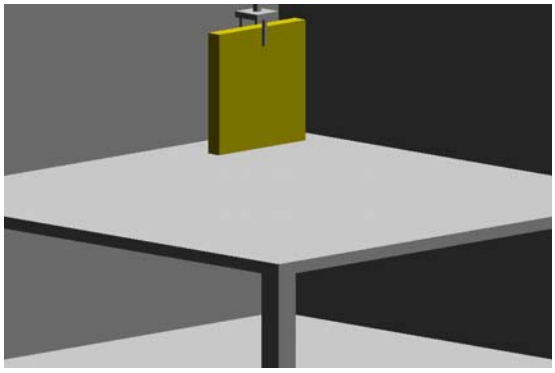
**Σχήμα 76: Αντικείμενο με h=0.20m και z=0.12m**



**Σχήμα 77: Αντικείμενο με h=0.20m και z=0.15m**

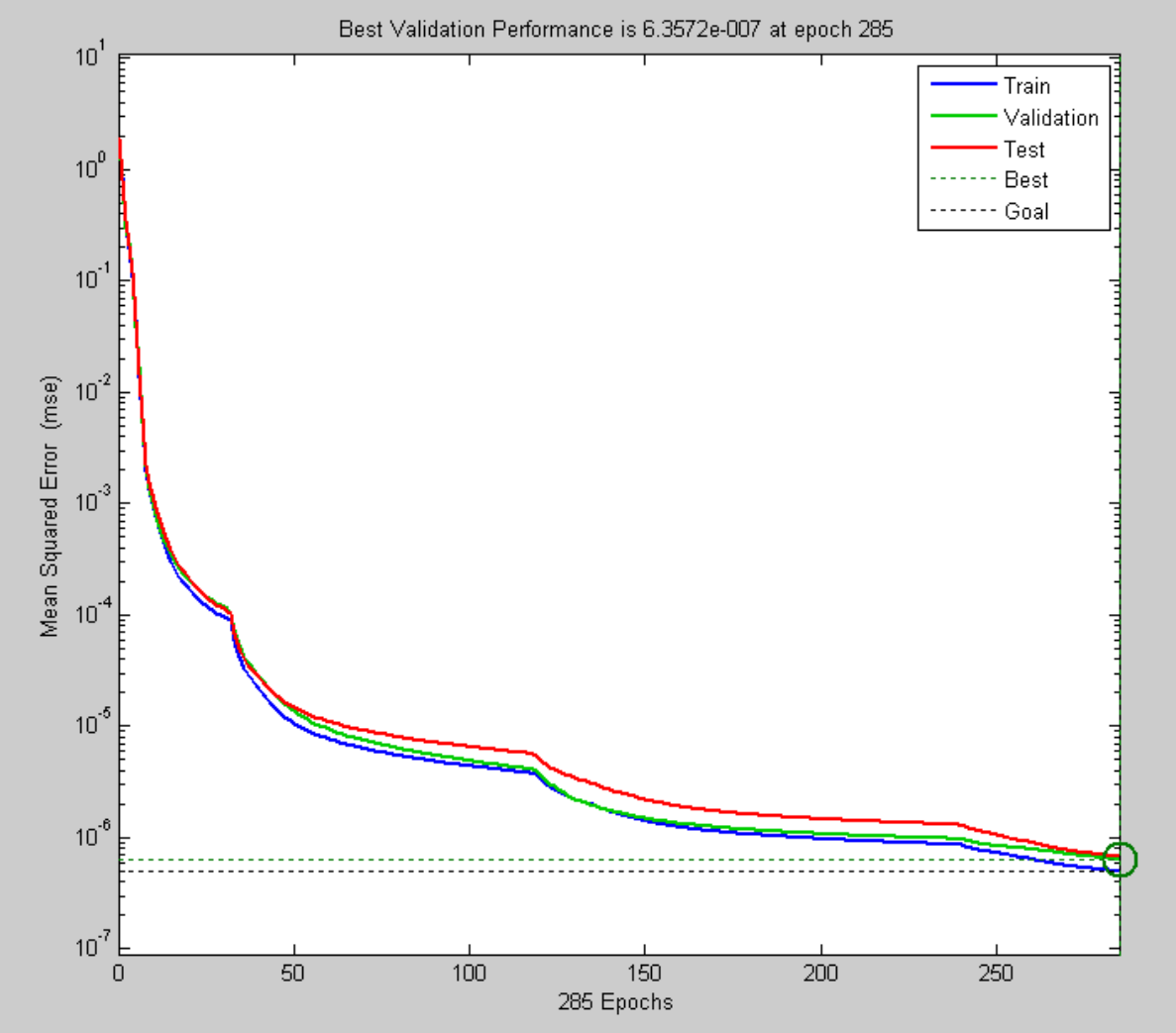


**Σχήμα 78: Αντικείμενο με h=0.20m και z=0.18m**



**Σχήμα 79: Αντικείμενο με h=0.20m και z=0.20m**

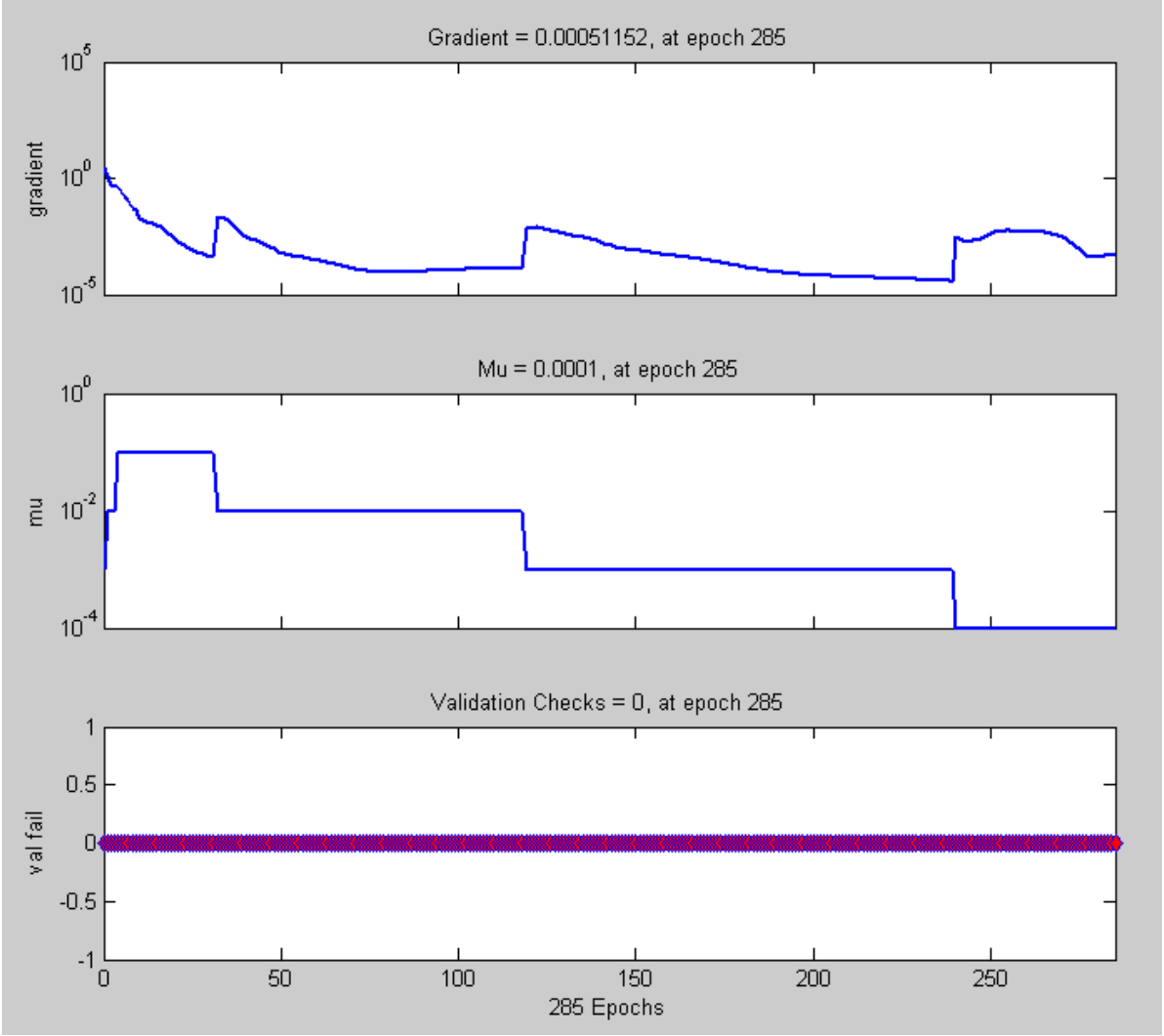
Η επίδοση του νευρωνικού δικτύου με βάση το μέσο τετραγωνικό σφάλμα φαίνεται στο Σχήμα 80.




**Σχήμα 80: Επίδοση εκπαίδευσης του νευρωνικού δικτύου με βάση το μέσο τετραγωνικό σφάλμα**

Στην ιδανική περίπτωση, το μέσο τετραγωνικό σφάλμα είναι μηδέν, δηλαδή, η έξοδος του νευρωνικού δικτύου ταυτίζεται με την επιθυμητή έξοδο. Ωστόσο, κάτι τέτοιο δεν είναι δυνατόν να συμβεί και κατά συνέπεια επιθυμούμε το μέσο τετραγωνικό σφάλμα να τείνει όσο το δυνατόν περισσότερο στο μηδέν. Στη συγκεκριμένη περίπτωση, από το Σχήμα 80 παρατηρούμε ότι το μέσο τετραγωνικό σφάλμα συγκλίνει στην τιμή $6 \cdot 10^{-7}$ σε λίγες επαναλήψεις του αλγορίθμου που χρησιμοποιήθηκε. Συνεπώς, η επίδοση του συγκεκριμένου νευρωνικού δικτύου είναι αρκετά καλή.

Η κλίση δ και ο συντελεστής μ, που χρησιμοποιούνται κατά την εκτέλεση του αλγορίθμου Levenberg – Marquardt καθώς και ο έλεγχος που γίνεται κατά τη διάρκεια εκπαίδευσης του νευρωνικού δικτύου για την ορθή λειτουργία του φαίνονται στο Σχήμα 81.




**Σχήμα 81: Γραφική παράσταση της κλίσης δ και του συντελεστή μ του αλγορίθμου Levenberg – Marquardt καθώς και του ελέγχου ορθής λειτουργίας του νευρωνικού δικτύου**

Στη συνέχεια, αφού εκπαιδεύτηκε το ρομπότ στο σύνολο εκμάθησης, δημιουργήθηκε και ένα νέο σύνολο από δεδομένα που αποτελεί το σύνολο ελέγχου. Το σύνολο αυτό αποτελείται από τρία μέρη. Το πρώτος μέρος προέρχεται από αντικείμενα με γνωστό ύψος h, δηλαδή, για ύψη στα οποία έχει ήδη εκπαιδευτεί το ρομπότ, αλλά με άγνωστα μήκη z στον άξονα z. Το δεύτερο μέρος προέρχεται από αντικείμενα με άγνωστα ύψη h και άγνωστα μήκη z. Πιο συγκεκριμένα, στον Πίνακα 6 και στον Πίνακα 7 παρουσιάζονται τα σύνολα ελέγχου που δημιουργήθηκαν.

| Ύψος αντικειμένου (m) | Μήκη αντικειμένων στον άξονα z (m) | | | | | |
|---|---|---|---|---|---|---|
| 0.025 | 0.04 | 0.06 | 0.08 | 0.11 | 0.14 | 0.19 |
| 0.05 | 0.04 | 0.06 | 0.08 | 0.11 | 0.14 | 0.19 |
| 0.07 | 0.04 | 0.06 | 0.08 | 0.11 | 0.14 | 0.19 |
| 0.09 | 0.04 | 0.06 | 0.10 | 0.12 | 0.15 | 0.19 |
| 0.12 | 0.04 | 0.06 | 0.09 | 0.13 | 0.15 | 0.18 |
| 0.16 | 0.04 | 0.07 | 0.11 | 0.13 | 0.17 | 0.19 |
| 0.20 | 0.04 | 0.07 | 0.11 | 0.13 | 0.17 | 0.20 |

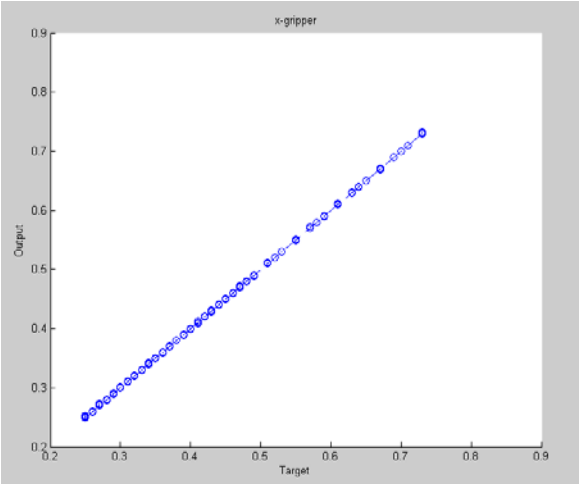
**Πίνακας 6: Πίνακας μηκών χειριζόμενων αντικειμένων ελέγχου με γνωστό ύψος**

| Ύψος αντικειμένου (m) | Μήκη αντικειμένων στον άξονα z (m) | | | | | |
|---|---|---|---|---|---|---|
| 0.04 | 0.04 | 0.06 | 0.10 | 0.13 | 0.15 | 0.18 |
| 0.06 | 0.04 | 0.06 | 0.10 | 0.13 | 0.15 | 0.18 |
| 0.10 | 0.04 | 0.06 | 0.08 | 0.10 | 0.12 | 0.17 |
| 0.14 | 0.04 | 0.06 | 0.10 | 0.12 | 0.14 | 0.16 |
| 0.18 | 0.04 | 0.06 | 0.10 | 0.15 | 0.18 | 0.20 |

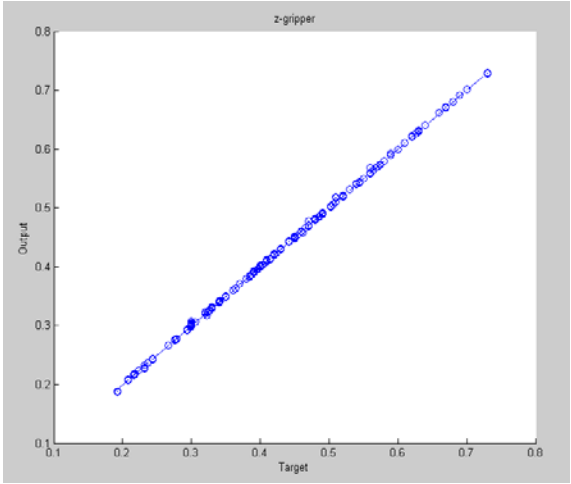
**Πίνακας 7: Πίνακας μηκών χειριζόμενων αντικειμένων ελέγχου με άγνωστο ύψος**

Παρακάτω παρουσιάζονται τα αποτελέσματα από τον έλεγχο γενίκευσης του νευρωνικού δικτύου στα σύνολα ελέγχου. Στα αποτελέσματα αυτά υπάρχουν οι γραφικές παραστάσεις των εξόδων του νευρωνικού δικτύου συναρτήσει των επιθυμητών εξόδων για τις συντεταγμένες (x-gripper, y-gripper, z-gripper) και της γωνίας κύλισης (Roll) της ρομποτικής αρπάγης καθώς και των σφαλμάτων απόκλισης των εξόδων του νευρωνικού δικτύου από τις επιθυμητές τιμές. Ως αποτυχία σωστής προσέγγισης του αντικειμένου θεωρήθηκε η απόκλιση της εξόδου του νευρωνικού δικτύου από την επιθυμητή τιμή κατά 0.01m για τις συντεταγμένες θέσης της ρομποτικής αρπάγης ή κατά $4^o$ απόκλιση στη γωνία κύλισης. Η αποτυχία της σωστής προσέγγισης παρουσιάζεται με κόκκινο κύκλο στα σχήματα και η επιτυχία με μπλε κύκλο. Επίσης στα σχήματα παρουσιάζονται με οριζόντιες διακεκομμένες γραμμές τα όρια αποτυχίας του ρομπότ

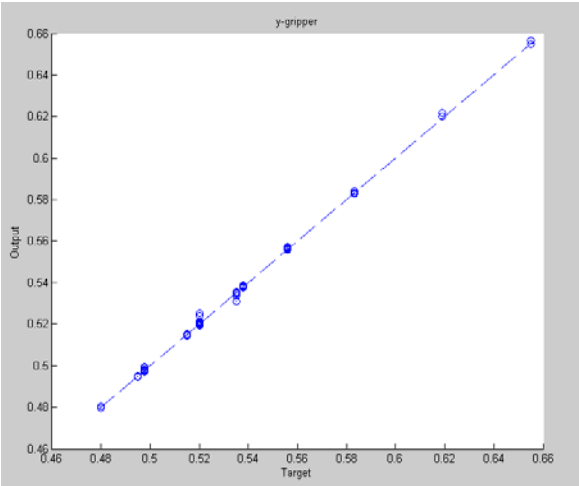
Από το Σχήμα 82 έως και το Σχήμα 89 παρουσιάζονται τα αποτελέσματα για το πρώτο μέρος του συνόλου ελέγχου.



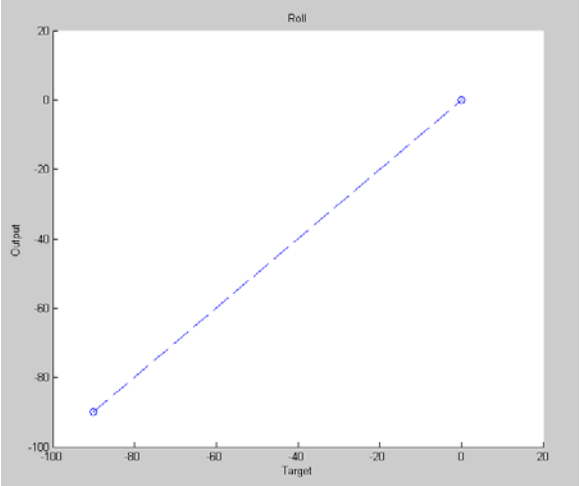

**Σχήμα 82: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη x της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



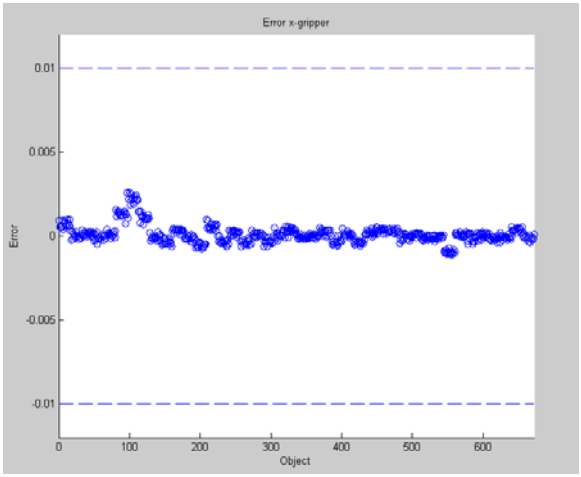

**Σχήμα 84: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη z της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



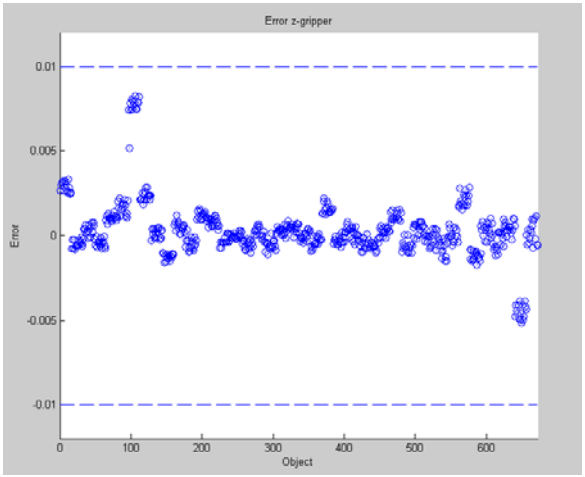

**Σχήμα 83: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη y της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



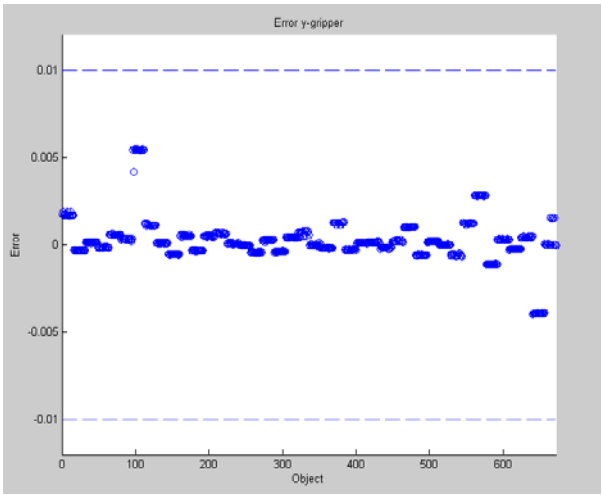

**Σχήμα 85: Έξοδος νευρωνικού δικτύου για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



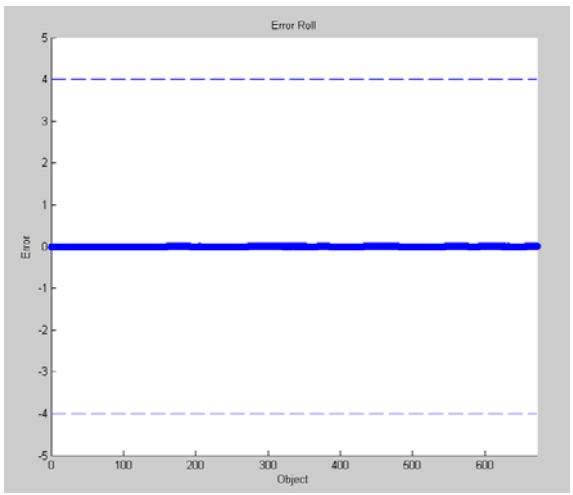

**Σχήμα 86: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη x της ρομποτικής αρπάγης**




**Σχήμα 88: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη z της ρομποτικής αρπάγης**



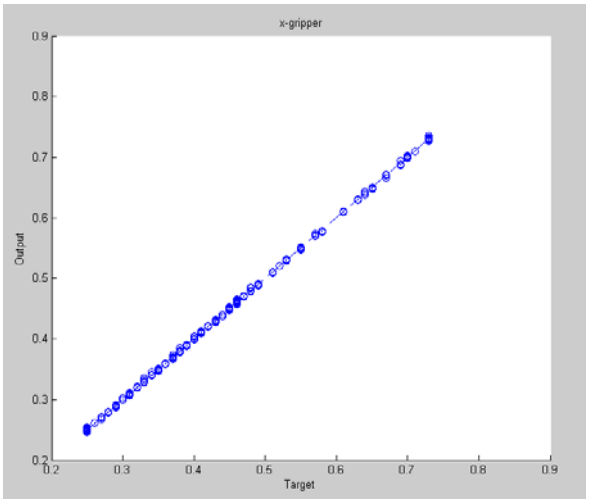

**Σχήμα 87: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη y της ρομποτικής αρπάγης**



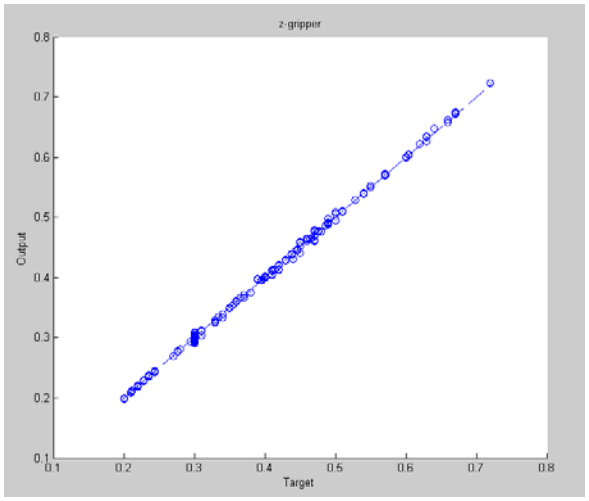

**Σχήμα 89: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης**

Από τις παραπάνω γραφικές παραστάσεις, γίνεται αντιληπτό, ότι το ρομπότ δεν απέτυχε καμία φορά στο να προσεγγίσει και να πιάσει σωστά το αντικείμενο. Οι έξοδοι του νευρωνικού δικτύου προσεγγίζουν αρκετά καλά τις επιθυμητές εξόδους καθώς όπως φαίνεται και από τα διαγράμματα των σφαλμάτων των εξόδων, οι τιμές των σφαλμάτων βρίσκονται εντός των ορίων που τέθηκαν εξαρχής.

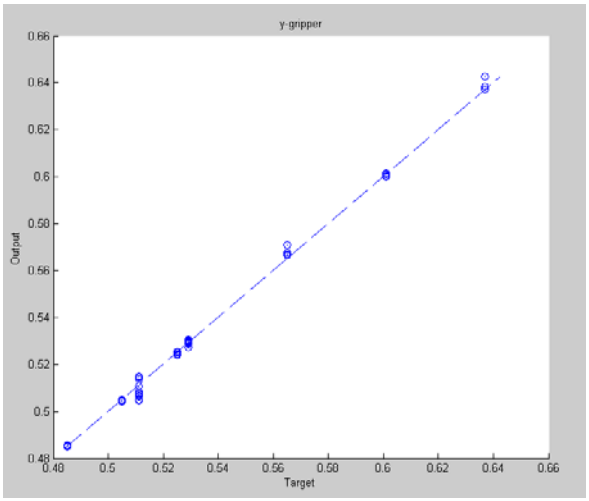
Από το Σχήμα 90 έως και το Σχήμα 97 παρουσιάζονται τα αποτελέσματα για το δεύτερο μέρος του συνόλου ελέγχου.



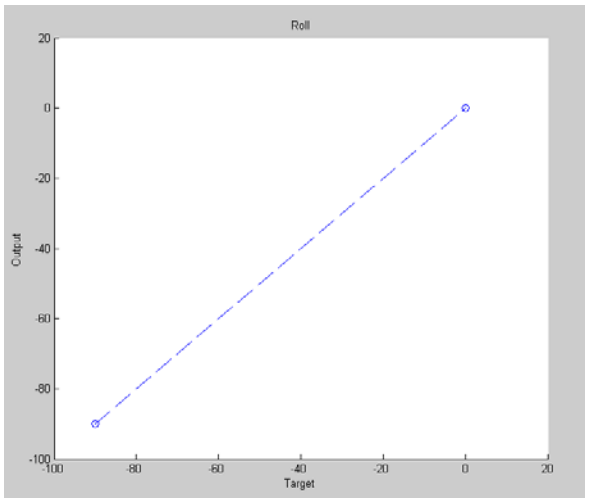

**Σχήμα 90: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη x της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



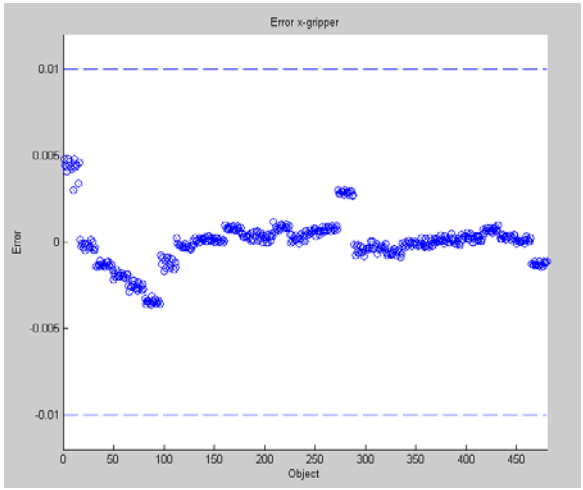

**Σχήμα 92: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη z της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



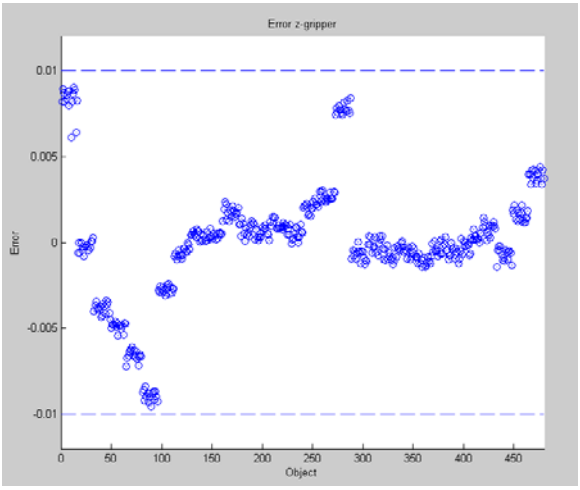

**Σχήμα 91: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη y της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



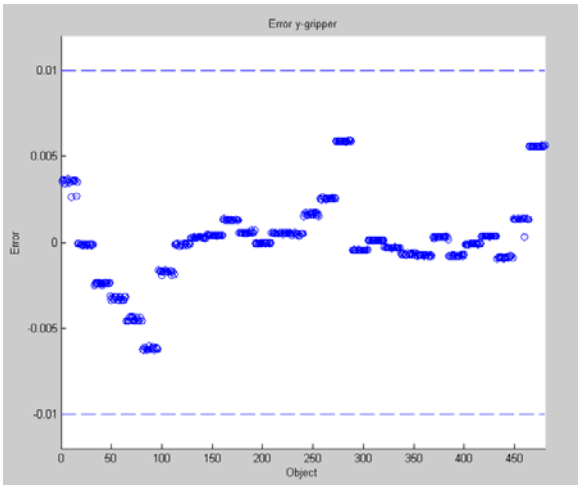

**Σχήμα 93: Έξοδος νευρωνικού δικτύου για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου**



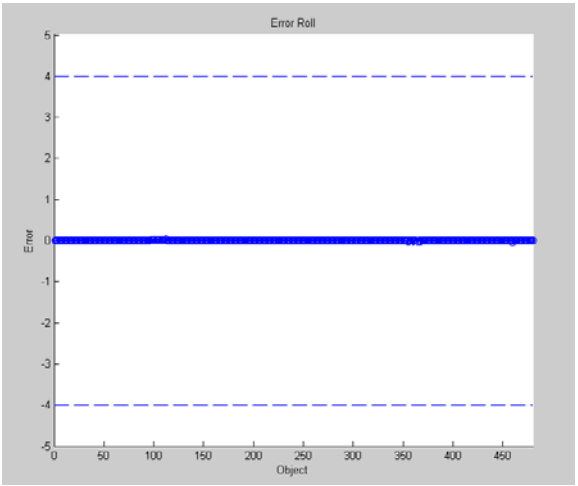

**Σχήμα 94: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη x της ρομποτικής αρπάγης**




**Σχήμα 96: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη z της ρομποτικής αρπάγης**



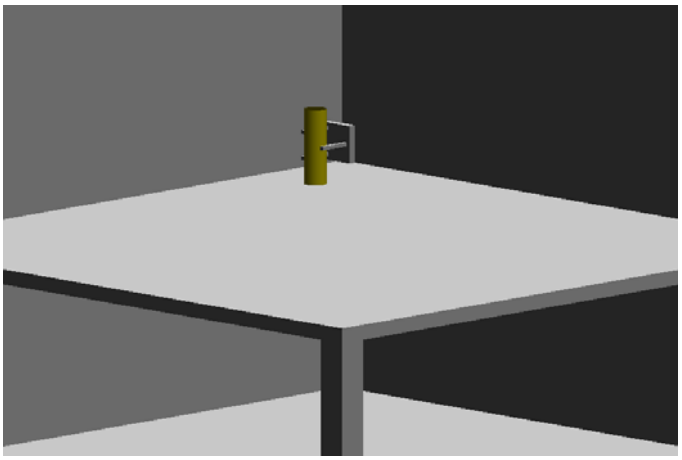

**Σχήμα 95: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη y της ρομποτικής αρπάγης**



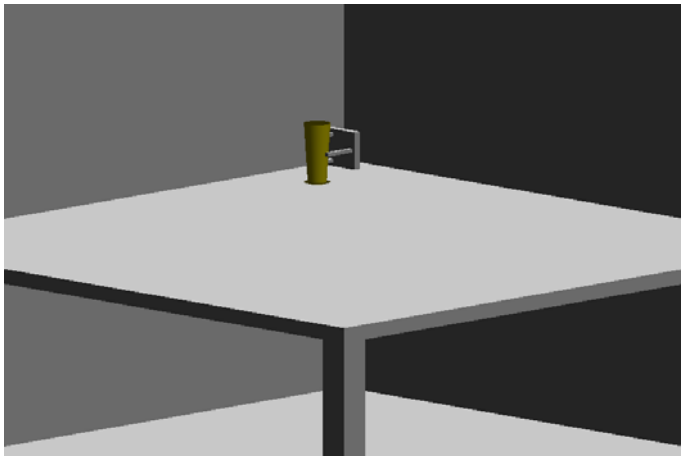

**Σχήμα 97: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης**

Όσον αφορά το δεύτερο μέρος του συνόλου ελέγχου, παρατηρείται ότι οι έξοδοι του νευρωνικού δικτύου προσεγγίζουν αρκετά καλά τις επιθυμητές εξόδους καθώς, όπως φαίνεται και από τα διαγράμματα των σφαλμάτων των εξόδων, οι τιμές των σφαλμάτων βρίσκονται εντός των ορίων που τέθηκαν εξαρχής. Πιο συγκεκριμένα, τα σφάλματα για τη συντεταγμένη z για την αρπαγή ορισμένων αντικειμένων πλησιάζουν, αρκετά, την οριακή τιμή. Αυτό συμβαίνει, κυρίως, για μικρά αντικείμενα καθώς το πρόγραμμα που δημιουργήθηκε στο περιβάλλον του MATLAB για την εξαγωγή των χαρακτηριστικών των αντικειμένων, αδυνατεί, εν μέρει, να εξάγει σωστά δεδομένα. Ωστόσο, ακόμα και σε αυτή την περίπτωση το ρομπότ δεν απέτυχε καμία φορά στο να προσεγγίσει και να πιάσει σωστά το αντικείμενο.

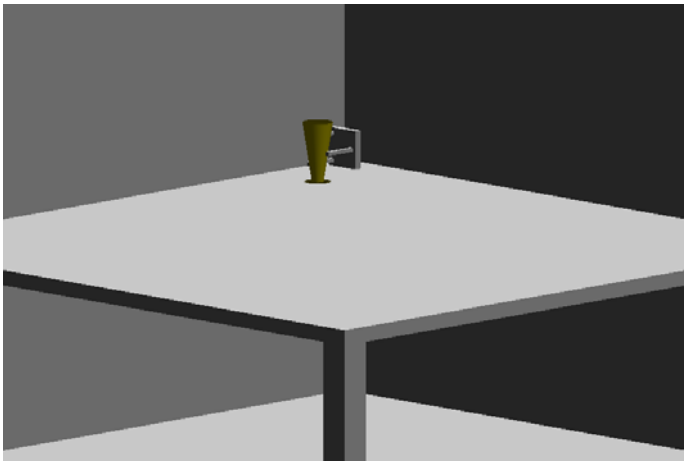
Τέλος, για τον έλεγχο γενίκευσης του νευρωνικού δικτύου σε διαφορετικά αντικείμενα από αυτά της εκπαίδευσης, δημιουργήθηκαν και δύο νέα διαφορετικά αντικείμενα τα οποία παρουσιάζουν γεωμετρικές ομοιότητες με τα γνωστά αντικείμενα της εκπαίδευσης. Πιο συγκεκριμένα, όπως φαίνεται στο Σχήμα 98, στο Σχήμα 99 και στο Σχήμα 100 δημιουργήθηκαν ένας κύλινδρος και δύο ποτήρια, ένα εκ των οποίων (Σχήμα 100), όμως, δεν παρουσιάζει αρκετή γεωμετρική ομοιότητα με τα αντικείμενα της εκπαίδευσης. Αυτά τα αντικείμενα αποτελούν το τρίτο μέρος του ελέγχου.



**Σχήμα 98: Κύλινδρος για τον έλεγχο γενίκευσης της εκπαίδευσης**

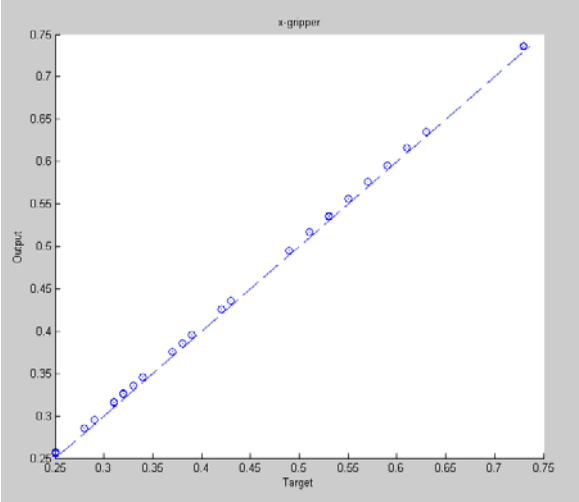


**Σχήμα 99: Ποτήρι για τον έλεγχο της γενίκευσης της εκπαίδευσης**

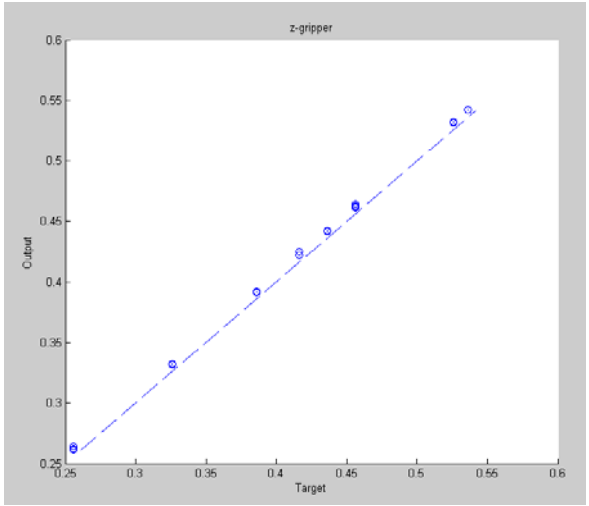


**Σχήμα 100: Ποτήρι για τον έλεγχο της γενίκευσης της εκπαίδευσης**

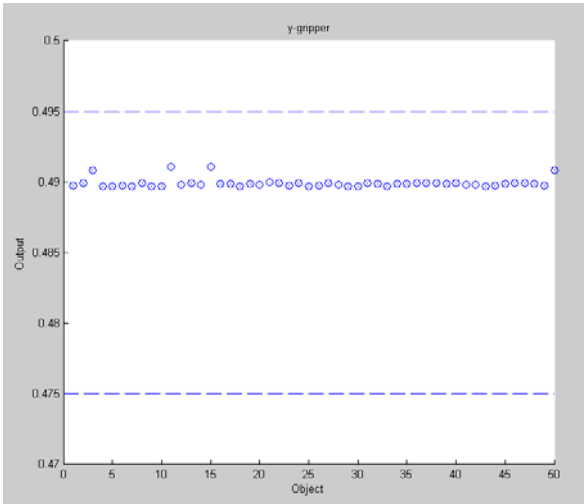
Από το Σχήμα 101 έως και το Σχήμα 108 παρουσιάζονται τα αποτελέσματα για τον έλεγχο του νευρωνικού δικτύου χρησιμοποιώντας ως αντικείμενο ελέγχου τον κύλινδρο.



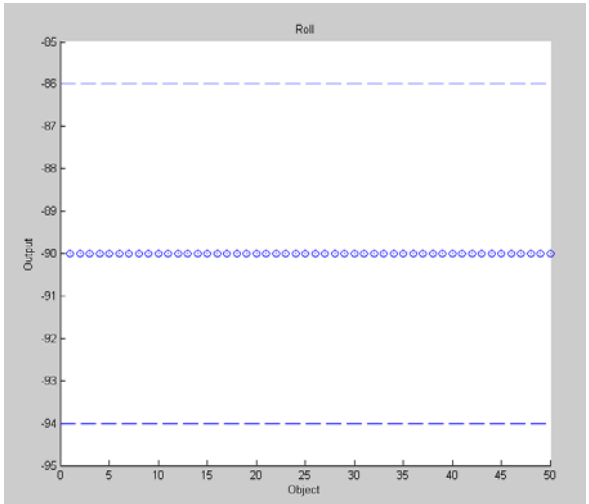

**Σχήμα 101: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη x της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή του κυλίνδρου**



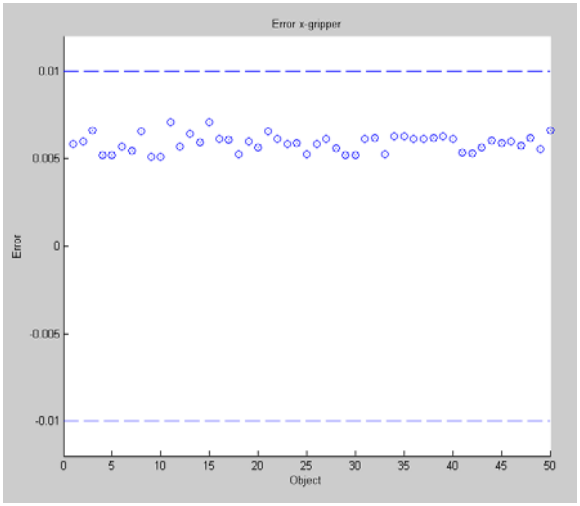

**Σχήμα 103: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη z της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή του κυλίνδρου**



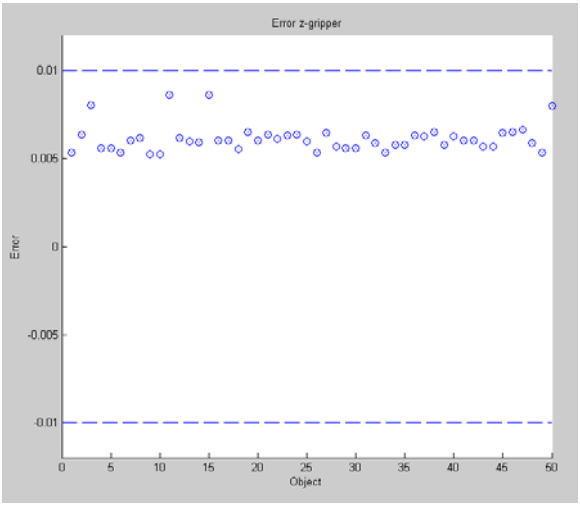

**Σχήμα 102: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη y της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή του κυλίνδρου**



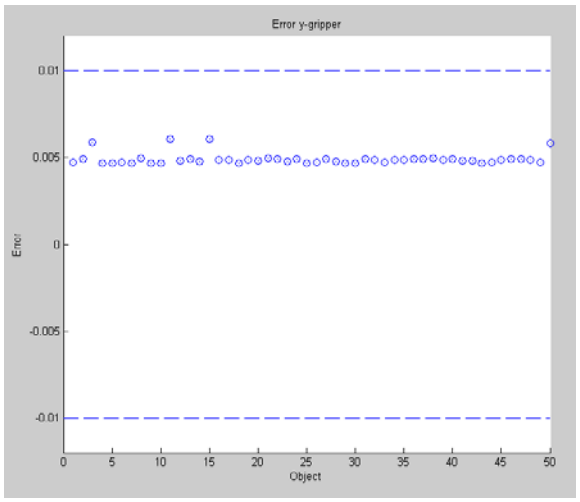

**Σχήμα 104: Έξοδος νευρωνικού δικτύου για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή του κυλίνδρου**



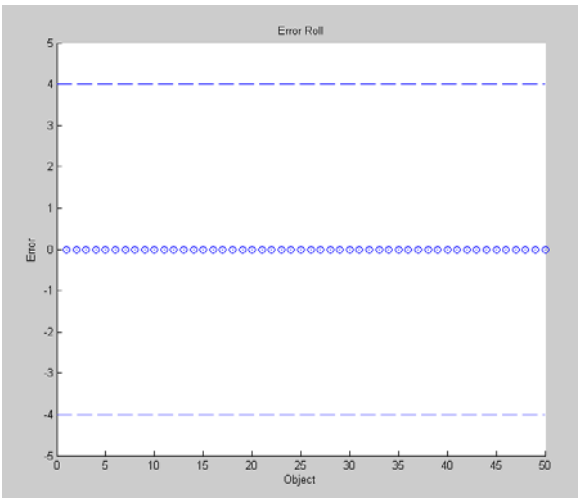

**Σχήμα 105: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη x της ρομποτικής αρπάγης για την αρπαγή του κυλίνδρου**




**Σχήμα 107: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη z της ρομποτικής αρπάγης για την αρπαγή του κυλίνδρου**



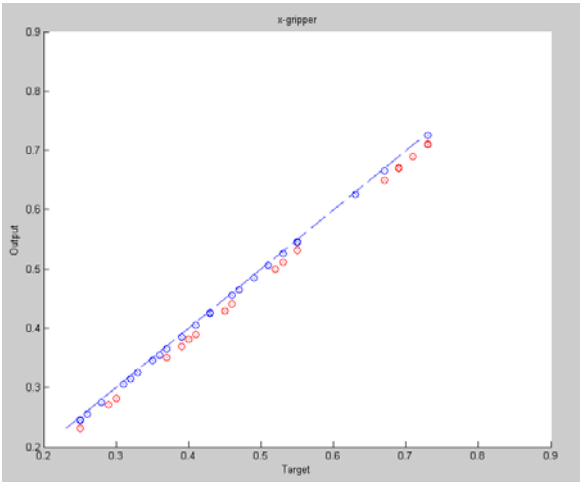

**Σχήμα 106: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη y της ρομποτικής αρπάγης για την αρπαγή του κυλίνδρου**



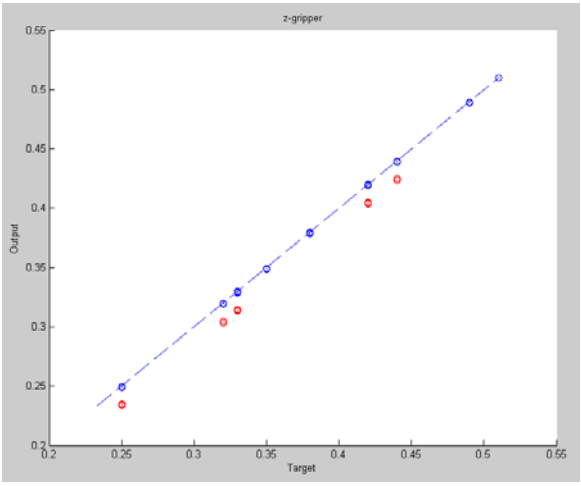

**Σχήμα 108: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης για την αρπαγή του κυλίνδρου**

Ο κύλινδρος που χρησιμοποιήθηκε για τον έλεγχο της γενίκευσης της εκπαίδευσης παρουσιάζει μεγάλη ομοιότητα με τα αντικείμενα της εκπαίδευσης. Έτσι, όπως φαίνεται και από τα διαγράμματα των σφαλμάτων των εξόδων του νευρωνικού δικτύου, η ρομποτική αρπάγη επιτυγχάνει όλες τις φορές στο να πιάσει σωστά τον κύλινδρο καθώς όλα τα σφάλματα βρίσκονται εντός των αρχικών ορίων που τέθηκαν.

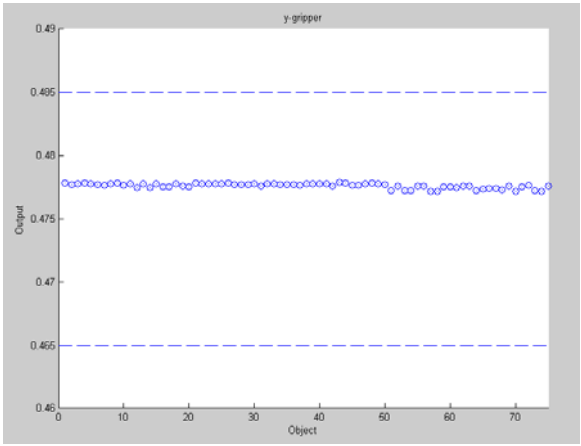
Από το Σχήμα 109 έως και το Σχήμα 116 παρουσιάζονται τα αποτελέσματα για τον έλεγχο του νευρωνικού δικτύου χρησιμοποιώντας ως αντικείμενο ελέγχου το ποτήρι.



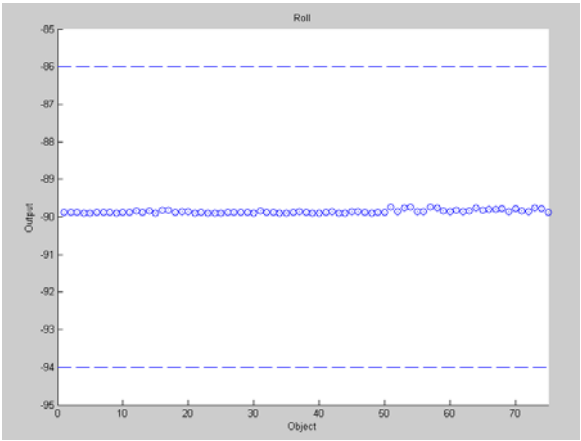

**Σχήμα 109: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη x της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή των ποτηριών**



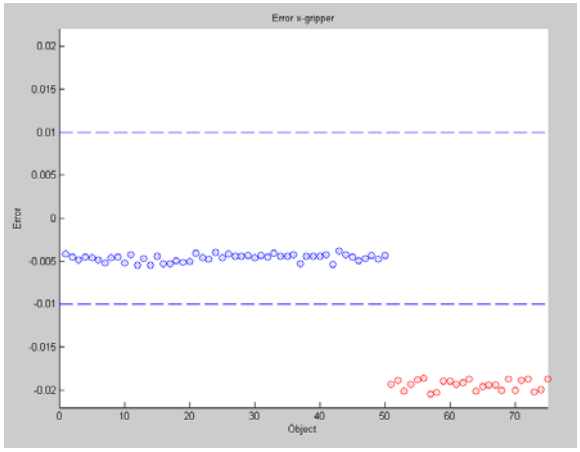

**Σχήμα 111: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη z της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή των ποτηριών**



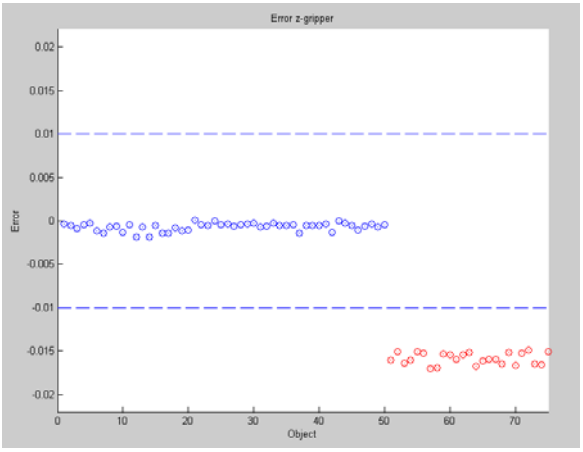

**Σχήμα 110: Έξοδος νευρωνικού δικτύου για τη συντεταγμένη y της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή των ποτηριών**



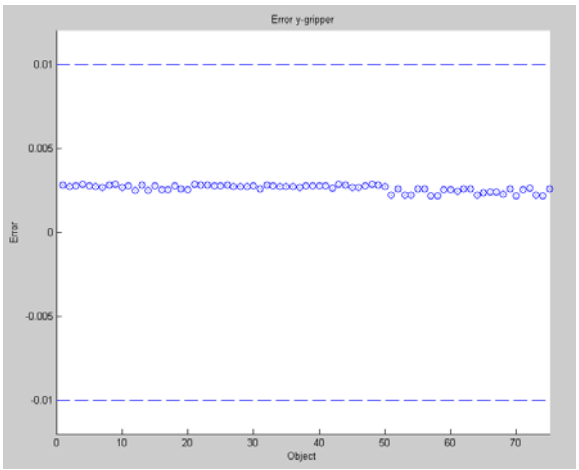

**Σχήμα 112: Έξοδος νευρωνικού δικτύου για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης συναρτήσει της επιθυμητής εξόδου για την αρπαγή των ποτηριών**



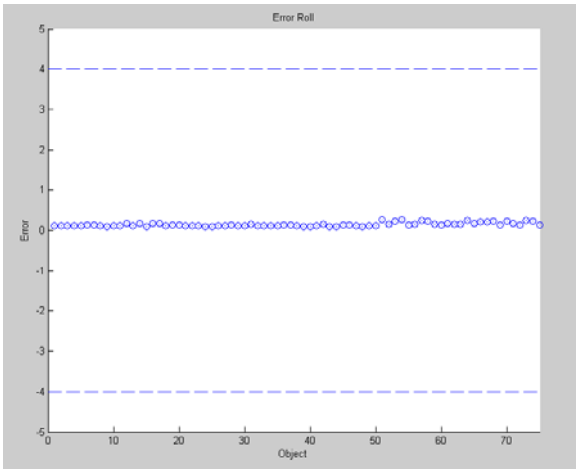

**Σχήμα 113: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη x της ρομποτικής αρπάγης για την αρπαγή των ποτηριών**




**Σχήμα 115: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη z της ρομποτικής αρπάγης για την αρπαγή των ποτηριών**




**Σχήμα 114: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη συντεταγμένη y της ρομποτικής αρπάγης για την αρπαγή των ποτηριών**




**Σχήμα 116: Σφάλμα εξόδου του νευρωνικού δικτύου από την επιθυμητή έξοδο για τη γωνία κύλισης (Roll) της ρομποτικής αρπάγης για την αρπαγή των ποτηριών**

Όπως φαίνεται από τα παραπάνω αποτελέσματα, το ρομπότ αποτυγχάνει να πιάσει σωστά ορισμένα αντικείμενα. Αυτά τα αντικείμενα είναι τα ποτήρια που δεν παρουσιάζουν γεωμετρική ομοιότητα με τα αντικείμενα της εκπαίδευσης. Εξαιτίας της αποτυχίας του ρομπότ να πιάσει σωστά αυτά τα αντικείμενα, παρουσιάζονται στις γραφικές παραστάσεις με κόκκινο κύκλο ενώ τα ποτήρια που έχουν γεωμετρική ομοιότητα με τα αντικείμενα της εκπαίδευσης παρουσιάζονται με μπλε κύκλο καθώς το ρομπότ τα έπιασε σωστά.

ΚΕΦΑΛΑΙΟ 6

# ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΚΑΤΕΥΘΥΝΣΕΙΣ ΤΗΣ ΕΡΕΥΝΑΣ

## 6.1 Συμπεράσματα

Η παρούσα εργασία είχε ως βασικό σκοπό τη δημιουργία ενός αυτοματοποιημένου προγράμματος για την εκ των προτέρων εκπαίδευση ενός ρομποτικού χειριστή στο να προσεγγίζει και να πιάνει ορισμένα αντικείμενα χρησιμοποιώντας αποκλειστικά οπτική (μονοσκοπική) πληροφορία. Στόχο της εργασίας αποτελεί η διερεύνηση της δυνατότητας εκπαίδευσης του ρομποτικού χειριστή με τρόπο αυτοματοποιημένο επί ενός εικονικού περιβάλλοντος. Για την επίτευξη του στόχου αυτού, δημιουργήθηκε ένα τριδιάστατο εικονικό περιβάλλον, στο οποίο έγινε η αυτοματοποιημένη εκμάθηση του ρομπότ και τα συμπεράσματα των αποτελεσμάτων αυτής παρουσιάζονται παρακάτω.

Αρχικά, το σύνολο εκπαίδευσης που χρησιμοποιήθηκε ήταν αρκετά μεγάλο. Έτσι, ένα από τα συμπεράσματα που εξήχθησαν είναι η επίτευξη πολύ μικρής απόκλισης στις τιμές των εξόδων του νευρωνικού δικτύου από τις επιθυμητές. Ωστόσο, ορισμένες φορές τα σφάλματα των εξόδων του νευρωνικού δικτύου από τις επιθυμητές εξόδους προσέγγιζαν τις οριακές τιμές που τέθηκαν αρχικώς, γεγονός που οφείλεται, κυρίως, στην αδυναμία εύρεσης των χαρακτηριστικών των πολύ μικρών αντικειμένων από το πρόγραμμα που δημιουργήθηκε στο περιβάλλον MATLAB και όχι στο νευρωνικό δίκτυο που δημιουργήθηκε. Όπως προέκυψε από δοκιμές, σε περίπτωση μικρότερου συνόλου εκπαίδευσης, τα σφάλματα είναι μεγαλύτερα και το ρομπότ αποτυγχάνει αρκετές φορές στο να προσεγγίσει και να πιάσει σωστά τα αντικείμενα.

Στη συνέχεια, ένα άλλο συμπέρασμα που εξάγεται, είναι η δυνατότητα γενίκευσης της εκπαίδευσης σε αντικείμενα διαφορετικά από τα αυτά που αναγνωρίζει το ρομπότ, τα οποία, ωστόσο, παρουσιάζουν αρκετή γεωμετρική ομοιότητα με τα αντικείμενα της εκπαίδευσης. Στην περίπτωση που τα άγνωστα αντικείμενα, όμως, διαφέρουν γεωμετρικώς από τα αντικείμενα της εκπαίδευσης, τότε τέτοιου είδους γενίκευση δεν είναι εφικτή και το ρομπότ αποτυγχάνει να τα πιάσει σωστά.

Τέλος, παρατηρείται η αδυναμία της συγκεκριμένης υλοποίησης για την εκπαίδευση του ρομπότ σε αντικείμενα με διαφορετικό προσανατολισμό από αυτόν που χρησιμοποιήθηκε. Αυτό οφείλεται στο γεγονός ότι χρησιμοποιήθηκε μόνο μία κάμερα για την όραση του ρομπότ.

## 6.2 Μελλοντικές κατευθύνσεις της έρευνας

Στην παρούσα εργασία, υλοποιήθηκε ένα πρόγραμμα σε αντικειμενοστραφή προγραμματισμό (C++) κάνοντας χρήση και της βιβλιοθήκης OpenGL για την δημιουργία ενός τριδιάστατου εικονικού περιβάλλοντος, στο οποίο υπήρχε μόνο η ρομποτική αρπάγη και όχι ολόκληρος ο ρομποτικός χειριστής. Σκοπός του προγράμματος ήταν η συστηματική και αυτοματοποιημένη εκμάθηση ενός ρομποτικού χειριστή στο να προσεγγίζει και να πιάνει ορισμένα αντικείμενα. Για την προσομοίωση της όρασης του ρομπότ δημιουργήθηκε μια στατική εικονική κάμερα στο περιβάλλον. Με τη χρήση ενός προγράμματος, που υλοποιήθηκε στο περιβάλλον του MATLAB, το οποίο καλείται μέσα από το πρόγραμμα σε C++ και εκτελείται αυτόματα, έγινε η εξαγωγή ορισμένων χαρακτηριστικών των αντικειμένων που χρησιμοποιήθηκαν για την εκπαίδευση του ρομπότ. Στη συνέχεια, δημιουργήθηκε ένα νευρωνικό δίκτυο, στο οποίο εισήχθησαν τα δεδομένα της εκπαίδευσης και έγινε έλεγχος της γενίκευσης της εκπαίδευσης σε ένα πλήθος αντικειμένων, συμπεριλαμβανομένων και εντελώς άγνωστων αντικειμένων. Η συγκεκριμένη έρευνα μπορεί να εξελιχθεί σε αρκετούς τομείς.

Αρχικά, μπορεί να δημιουργηθεί ένα πρόγραμμα το οποίο να ελέγχει απευθείας έναν ρομποτικό χειριστή σε πραγματικό, πλέον, περιβάλλον, ή αν η εκπαίδευση γίνεται σε εικονικό περιβάλλον τότε να απεικονίζεται ολόκληρο το ρομπότ και να προσομοιώνεται ολόκληρη η κινηματική αλυσίδα του ρομπότ. Ακόμα, η όραση του ρομπότ μπορεί να υλοποιηθεί με δύο ή περισσότερες κάμερες, γεγονός το οποίο θα δώσει ακριβέστερα αποτελέσματα όσον αφορά στα χαρακτηριστικά των αντικειμένων καθώς και στη δυνατότητα εκμάθησης του ρομπότ σε αντικείμενα με διαφορετικούς προσανατολισμούς. Έτσι, θα υπάρχει γενίκευση της εκπαίδευσης του ρομπότ σε μεγάλο πλήθος αντικειμένων, παρουσιάζοντας, επομένως, περισσότερες δυνατότητες σε εφαρμογές της καθημερινότητας του ανθρώπου.

Επίσης, μπορούν να χρησιμοποιηθούν και άλλα χαρακτηριστικά περιγραφής των αντικειμένων, πέρα από αυτά που χρησιμοποιήθηκαν στη συγκεκριμένη εργασία, ή και διαφορετικοί τρόποι εξαγωγής των χαρακτηριστικών αυτών με γρηγορότερα και αποτελεσματικότερα προγράμματα.

Τέλος, για την εκπαίδευση του ρομπότ μπορούν να χρησιμοποιηθούν και κάποιες άλλες μέθοδοι, που αναφέρθηκαν στη συγκεκριμένη εργασία, πέρα από το νευρωνικό δίκτυο, για τη σύγκριση των αποτελεσμάτων και την επιλογή της καταλληλότερης μεθόδου για το συγκεκριμένο πρόβλημα. Επίσης, προτείνεται η εύρεση νέων μεθόδων, οι οποίες να απαιτούν μικρότερο σύνολο εκπαίδευσης με καλύτερα αποτελέσματα έτσι ώστε να είναι δυνατή η εκπαίδευση του ρομπότ από την αλληλεπίδραση ανθρώπου – ρομπότ σε πραγματικό περιβάλλον.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] B. Myers, "A Brief History of Human Computer Interaction Technology", in: *ACM interactions*, Volume, 5, Issue 2, pp. 44-54, March 1998.

[2] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot, "Programming by Demonstration", in: *Handbook of Robotics*, Springer, 2008.

[3] J. R. Wolpawab, N. Birdaumercd, D. J. McFarlanda, G. Pfurtschellere, T. M. Vaughana, "Brain – computer interfaces for communication and control", in: *Clinical Neurophysicology*, Volume 113, Issue 6, pp. 767-791, June 2002.

[4] B. Argall, S. Chernova, M. Veloso, B. Browning, "A survey of robot learning from demonstration" in: *Robotics and Autonomous Systems*, Volume 57, Issue 5, pp. 469-483, May 2009.

[5] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Second Edition, Springer Series in Statistics, 2008.

[6] S. Chernova, M. Veloso, "Confidence-based learning from demonstration using Gaussian Mixture Models", in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, AAMAS'07, 2007.

[7] C. Sammut, S. Hurst, D. Kedzier, D. Michie, "Learning to fly", in: *Proceedings of the Ninth International Workshop on Machine Learning*, 1992.

[8] T. Inamura, M. Inaba, H. Inoue, "Acquisition of probabilistic behaviour decision model based on the interactive teaching method", in: *Proceedings of the Ninth International Conference on Advanced Robotics*, ICAR'99, 1999.

[9] J. Saunders, C. Nehaniv, K. Dautenhahn, "Teaching robots by moulding behaviour and scaffolding the environment", in: *Proceedings of the 1st ACM/IEEE International Conference on Human-Robot Interaction*s, HRI'06, 2006.

[10] P. Pook, D. Ballard, "Recognizing teleoperated manipulations", in: *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'93, 1993.

[11] G. Hovland, P. Sikka, B. McCarragher, "Skill acquisition from human demonstration using a hidden Markov model", in: *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'96, 1996.

[12] K. Dixon, P. Khosla, "Learning by observation with mobile robots: A computational approach", in: *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'04, 2004.

[13] M. Matarić, K. Dautenhahn, C. Nehaniv, "Linking perception to action and biology to robotics", in: *Sensory-motor primitives as a basis for learning by imitation*, Chapter 15, MIT Press, Cambridge, MA, USA, 2002.

[14] O. Jenkins, M. Matarić, S. Weber, "Primitive-based movement classification for humanoid imitation", in: *Proceedings of the 1st IEEE-RAS International Conference on Humanoid Robotics*, 2000.

[15] A. Billard, M. Matarić, "Learning human arm movements by imitation: Evaluation of biologically inspired connectionist architecture", in: *Robotics and Autonomous Systems*, Volume 37, Issues 2-3, pp. 145-160, 30 November 2001.

[16] M. Nicolescu, M. Matarić, "Learning and interacting in human-robot domains", in: *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, Volume 31, Issue 5, pp. 419-430, 2001.

[17] R. Voyles, P. Khosla, "A multi-agent system for programming robots by human demonstration", *Integrated Computer-Aided Engineering*, Volume 8, Issue 51 pp. 59-67, 2001.

[18] P. Rybski, R. Voyles, "Interactive task training of a mobile robot through human gesture recognition", in: *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'99, 1999.

[19] A. Lockerd, C. Breazeal, "Tutelage and socially guided robot learning", in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'04, 2004.

[20] S. Chernova, M. Veloso, "Teaching multi-robot coordination using demonstration of communication and state sharing", in: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, AAMAS'08, 2008.

[21] C. Atkeson, A. Moore, S. Schaal, "Locally weighted learning", *Artificial Intelligence Review*, Chapter 11, pp. 11-73, 1997.

[22] D. Bentivegna, "Learning from Observation Using Primitives", Ph.D. Thesis, College of Computing, Georgia Institute of Technology, Atlanta, GA, July 2004.

[23] B. Argall, B. Browning, M. Veloso, "Learning from demonstration with the critique of a human teacher", in: *Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions*, HRI'07, 2007.

[24] W. Cleveland, C. Loader, "Smoothing by local regression: Principles and methods", Technical Report, AT & T Bell Laboratories, Murray Hill, NJ, 1995.

[25] S. Schaal, D. Sternad, "Programmable pattern generators", in: *3rd International Conference on Computational Intelligence in Neuroscience*, 1998.

[26] A. Ijspeert, J. Nakanishi, S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators", in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'02, 2002.

[27] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, M. Kawato, "Learning from demonstration and adaptation of biped locomotion", *Robotics and Autonomous Systems*, Chapter 47, pp,79-91, 2004.

[28] S. Schaal, C. Atkeson, "Constructive incremental learning from only local information", *Neural Computation*, Volume 10, Issue 8, pp. 2047-2084, 1998.

[29] S. Vijayakumar, S. Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space", in: *Proceedings of the 17th International Conference on Machine Learning*, ICML'00, 2000.

[30] D. Grollman, O. Jenkins, "Dogged learning for robots", in: *Proceedings of the IEEE International Conference on Robotics and Automation,* ICRA'07, 2007.

[31] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation", *Neural Computation*, Volume 3, Issue 1, pp. 88-97, 1991.

[32] R. Dillmann, M. Kaiser, A. Ude, "Acquisition of elementary robot skills from human demonstration", in: *International Symposium on Intelligent Robotic Systems*, SIRS'95, 1995.

[33] A. Ude, C. Atkeson, M. Riley, "Programming full-body movements for humanoid robots by observation", *Robotics and Autonomous Systems*, Chapter 47, pp. 93-108, 2004.

[34] J. Steil, F. Røthling, R. Haschke, H. Ritter, "Situated robot learning for multi-modal instruction and imitation of grasping", *Robot Learning by Demonstration, Robotics and Autonomous Systems,* Volume 47, Issues 2-3, pp. 129-141, 30 June 2004.

[35] S. Calinon, A. Billard, "Incremental learning of gestures by imitation in a humanoid robot", in: *Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions*, HRI'07, 2007.

[36] D. Grollman, O. Jenkins, "Sparse incremental learning for interactive robot control policy estimation", in: *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA'08, 2008.

[37] Α. Μπολοβίνου, "Ανάπτυξη συστήματος διαχείρισης μηνυμάτων για εφαρμογές έξυπνων αυτοκινήτων", Διπλωματική εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα, Οκτώβριος 2004

[38] H. Demuth, M. Beale, M. Hagan, *Neural Network Design*, MATLAB Neural Network Toolbox User's Guide

[39] M. Agoston, *Computer graphics and geometric modelling: implementation and algorithms*, Springer, 2005.

[40] L. Breiman, "Heuristics of Instability and Stabilization in Model Selection", in: *The Annals of Statistic,* Volume 24, Issue 6, pp. 2350-2383, December 1996.

[41] G. Hinton, T. Sejnowski, *Unsupervised Learning: Foundations of Neural Computation*, MIT Press, 1999.

[42] R. Duda, P. Hart, D. Stork: "Unsupervised Learning and Clustering", in: *Pattern classification (2nd edition)*, Chapter 10, 2001.

[43] R. Sutton, "Temporal Credit Assignment in Reinforcement Learning", PhD thesis, 1984.

[44] R. Williams, "A class of gradient-estimating algorithms for reinforcement learning in neural networks", in: *Proceedings of the IEEE First International Conference on Neural Networks*, 1987.

[45] B. Yegnanarayana, *Artificial Neural Networks*, New Delhi: Prentice Hall, 2006.

[46] T. Kohonen, "Self Organizing Maps", *Springer Series in Information Sciences*, 1994.

[47] E. Blum, L. K. Li, "Approximation theory and feedforward networks", in: *Neural Networks*, Volume 4, Issue 4, pp. 511-515, 1991.

[48] F. Pineda, "Generalization of Back-propagation to Recurrent Neural Networks", *Physical Review Letters*, Volume 59, Issue 19, pp. 2229-2232, 1987.

[49] S. Haykin, *Neural Networks a Comprehensive Foundation*, Second edition, Prentice Hall, 1999.

# ΠΑΡΑΡΤΗΜΑ Α

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <GL/glut.h>
#include <GL/glui.h>
#include "engine.h"
#include <time.h>
#include <math.h>


/********** User IDs for callbacks ********/
#define TRAIN_ENABLED_ID    200
#define OBJECT_ID             201
#define TRAIN_ROWS_ID         202
#define ZTRAINS_ID                  203
#define TEST_ENABLED_ID       204
#define QUIT_ID                     205

/* Define function prototypes */
double getRand(int num);

/* Define global parameters */
const double screenWidthd = 900.0, screenHeightd = 600.0;
const int screenWidthi=900, screenHeighti=600;
int temp=1, temp2=1, rows, nSize = screenHeighti * screenWidthi * 3,
objectsrow[10], obj=1, objects=0, allobjects, i, automatic=0, traintest=1;
int counter1=0, counter2=0, counter3=0, counter4=0, counter5=0,
counter6=0, draw=0;
unsigned char *imageData = (unsigned char*)malloc(nSize);
unsigned char *i_ptr = (unsigned char*)malloc(nSize);
double xgripper, ygripper, zgripper, rollobj, yawobj, roll, yaw, pitch,
*RED, *GREEN, *BLUE, *MODE;
double newline=0.0, xTr, yTr, zTr, xTrnew=0.0, yTrnew=0.0, zTrnew=0.0,
xgr, ygr, zgr, xsc;
float zsc[10], ysc;

/* Define the materials for the environment and the objects */
GLfloat mat_ambient[] = { 0.7f, 0.7f, 0.7f, 0.1f };
GLfloat mat_ambientsphere[] = { 1.0f, 0.93f, 0.0f, 0.1f };
GLfloat mat_diffuse[] = { 1.0f, 1.0f, 1.0f, 0.1f };
GLfloat mat_diffusesphere[] = { 1.0f, 0.93f, 0.0f, 0.1f };
GLfloat mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat mat_shininess[] = { 50.0f };

/* Define MATLAB pointers */
Engine *matlab;
mxArray *X = NULL, *Y = NULL, *Z = NULL, *M = NULL;
GLUquadric *Cyli;
GLUquadric *Disk;
```

```
/* Define the pointers for GLUI */
GLUI *glui;
GLUI_Listbox *objectlist;
GLUI_Panel *trainpanel;
GLUI_EditText *trainrows, *trainrow[10], *ztrains, *ztrain[10],
*objectheight;

/* Define the file pointers */
FILE *fp;
FILE *fp2;



/*************************   Idle function  ************************/

void myGlutIdle( void )
{
      GLUI_Master.sync_live_all();
      glutPostRedisplay();
}

/******************  GLUI control callback function  ******************/

void control_cb( int control )
{
  if ( control == TRAIN_ENABLED_ID ) {     // Train button pressed
      automatic=1;                         // Automatic creation of data
      traintest=1;                         // Create train data
      printf("Started collecting train data...\n");
      printf("\n");
      if (draw==1){
      printf("   Collecting data for object height = %6.4f\n", ysc);
      printf("   Object length = %6.4f\n", zsc[1]);}
      glutPostRedisplay();
  }
  if ( control == TEST_ENABLED_ID ) {      // Test button pressed
      automatic=1;                         // Automatic creation of data
      traintest=0;                         // Create test data
      printf("Started collecting test data...\n");
      printf("\n");
      if (draw==1){
      printf("   Collecting data for object height = %6.4f\n", ysc);
      printf("   Object length = %6.4f\n", zsc[1]);}
      glutPostRedisplay();
  }
  if ( control == QUIT_ID ) {
      engEvalString(matlab, "clear all;");      // Clear workspace
                                                  //of matlab and close it
      engEvalString(matlab, "close all;");
      engClose(matlab);
      exit(0);
      }
  if ( control == OBJECT_ID ) {            // Drop-Down list selected
      if (objectlist->get_int_val()==1){// Box selected to be drawn
         draw=1;
         for (int i=1; i<=ztrains->get_int_val(); i++){
            ztrain[i]->enable();
          }
          ztrains->enable();
          objectheight->enable();
      }
```

```
        if (objectlist->get_int_val()==2){// Cylinder selected to be drawn
              draw=2;
              for (int i=1; i<=7; i++)
                  ztrain[i]->disable();
              ztrains->set_int_val(1);
              ztrains->disable();
              objectheight->disable();
        }
        if (objectlist->get_int_val()==3){// Glass selected to be drawn
              draw=3;
              for (int i=1; i<=7; i++)
                  ztrain[i]->disable();
              ztrains->set_int_val(1);
              ztrains->disable();
              objectheight->disable();
        }
        GLUI_Master.sync_live_all();      // Sync all live parameters
        glutPostRedisplay();
  }
  if ( control == TRAIN_ROWS_ID ){        // Change in the number of rows
      for (int i=1; i<=7; i++){
         if (i<=trainrows->get_int_val()){
            trainrow[i]->enable();
         }
         if (i>trainrows->get_int_val())
            trainrow[i]->disable();
      }
      GLUI_Master.sync_live_all();
      for (int i=1; i<=trainrows->get_int_val(); i++){
         objects=objects+objectsrow[i];
      }
      allobjects=objects;
   }
   if ( control == ZTRAINS_ID ){// Change in the number of length trains
      for (int i=1; i<=7; i++){
          if (draw==1){
              if (i<=ztrains->get_int_val()){
                  ztrain[i]->enable();
              }
              if (i>ztrains->get_int_val())
                  ztrain[i]->disable();
              }
          else
              ztrain[i]->disable();
          }
      }
}



/****************   Function to get a random number  ******************/

double getRand(int num) {
           srand((unsigned)time(NULL)); //Seed rand func
           double theRand = (rand() % num ) / 100.0;
           return(theRand);
}
```

```
/************   Function to draw the gripper's fingers  ************/

void gripperHand(double thick, double len)
{
     glPushMatrix();
         glTranslated(0, -len/2.0, 0);
         glScaled(thick, len, thick);
         glutSolidCube(1.0);
     glPopMatrix();
}


/******************   Function to draw the gripper  *************/

void gripper(double topWidth, double topThick, double topHeight, double
gripHandThick, double gripHandLen)
{
     glPushMatrix();
         glScaled(topWidth, topThick, topHeight);
         glutSolidCube(1.0);
     glPopMatrix();
     double distx=0.8*topWidth/2.0 - gripHandThick/2.0;
       double distz=0.8*topHeight/2.0 - gripHandThick/2.0;
     glPushMatrix();
         glPushMatrix();
            glTranslated(0, gripHandLen/2.0, 0);
            gripperHand(gripHandThick*2.0, gripHandLen/2.0);
         glPopMatrix();
         glTranslated(distx, 0, 0);
         gripperHand(gripHandThick, gripHandLen);
         glTranslated(-2*distx, 0, -distz);
         gripperHand(gripHandThick, gripHandLen);
         glTranslated(0, 0, 2*distz);
         gripperHand(gripHandThick, gripHandLen);
    glPopMatrix();
}


/*****************   Function to draw the wall  ********************/

void wall(double thickness)
{
     glPushMatrix();
         glTranslated(2.5, 0, 2.5);
         glScaled(5.0, thickness, 5.0);
         glutSolidCube(1.0);
     glPopMatrix();
}

/****************   Function to draw the table's legs  *************/

void tableLeg(double thick, double len)
{
     glPushMatrix();
         glTranslated(0, len/2, 0);
         glScaled(thick, len, thick);
         glutSolidCube(1.0);
     glPopMatrix();
}
```

```
  /*******************  Function to draw the table  *****************/

void table(double topWid, double topThick, double legThick, double legLen)
{
       glPushMatrix();
       glTranslated(0.1, 0.125, 0.1);
     glPushMatrix();
         glTranslated(0, legLen, 0);
         glScaled(topWid, topThick, topWid);
         glutSolidCube(1.0);
     glPopMatrix();
     double dist=0.95*topWid/2.0 - legThick/2.0;
     glPushMatrix();
         glTranslated(dist, 0, dist);
         tableLeg(legThick, legLen);
         glTranslated(0, 0, -2*dist);
         tableLeg(legThick, legLen);
         glTranslated(-2*dist, 0, 2*dist);
         tableLeg(legThick, legLen);
         glTranslated(0, 0, -2*dist);
         tableLeg(legThick, legLen);
     glPopMatrix();
       glPopMatrix();
}




void myInit()
{
  /*************************************/
  /*       Set up OpenGL lights        */
  /*************************************/

     GLfloat light_intensity[] = { 0.7f, 0.7f, 0.7f, 0.1f };
     GLfloat light_position[] = { 2.0f, 6.0f, 3.0f, 0.0f };
     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
     glLightfv(GL_LIGHT0, GL_DIFFUSE, light_intensity);
     glEnable(GL_LIGHTING);
     glEnable(GL_LIGHT0);

     glShadeModel(GL_SMOOTH);        // Shade model

     glDepthFunc(GL_LESS);           //Create and enable Depth model
     glEnable(GL_DEPTH_TEST);
     glEnable(GL_NORMALIZE);

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     glClearColor(0.1f, 0.1f, 0.1f, 0.0f);
     glViewport(0, 0, screenWidthi, screenHeighti);
     gluLookAt(2.0, 0.8, 2.0, 0.0, 0.3, 0.0, 0.0, 1.0, 0.0);

}
```

```
/*********************  Main GFX Function  *********************/

void displaySolid(void)
{

     double scr=0.3;
     int flag=0;

     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
     glColor3d(0, 0, 0);

     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
      glOrtho(-scr*screenWidthd/screenHeightd,
scr*screenWidthd/screenHeightd, -scr, scr, -10, 100);
     glMatrixMode(GL_MODELVIEW);

       if (obj>objectsrow[temp]){         // Finished creating data for
                                                    //object in a row
            obj=1;
            temp=temp+1;
            objects=objects-objectsrow[temp-1];
            if (traintest==1)             // If data are for training then
                                                  //the next row is fixed
                  newline=newline+0.1;
            if (traintest==0)             // If data are for testing then
                                                 //the next row is random
           {
                 newline=newline+getRand(18);
                 if (newline>0.7)
                       newline=0.7;
                 }
       }
       if (temp>rows){        // Finished creating data for
                                                 //object in all rows
            automatic=1;            // Reset all parameters to create data
                                       //for the next length of the object
            objects=allobjects;
            obj=1;      temp2=temp2+1;    temp=1;
            newline=0.0; xTrnew=0.0; yTrnew=0.0; zTrnew=0.0;
            mxArray *X = NULL, *Y = NULL, *Z = NULL, *M = NULL;
            if (temp2<=ztrains->get_int_val())
                  printf("   Object length = %6.4f\n", zsc[temp2]);
       }
       if (temp2>ztrains->get_int_val()){       // Finished creating data
                                          //for all lengths of the object

            automatic=0;      objects=allobjects;     // Reset all
                           //parameters to start creating a new set of data
            obj=1;      temp=1; temp2=1;
            newline=0.0; xTrnew=0.0; yTrnew=0.0; zTrnew=0.0;
            mxArray *X = NULL, *Y = NULL, *Z = NULL, *M = NULL;
            printf("\n");
            printf("Finished collecting data!\n");
            printf("\n");
       }
```

```
        glPushMatrix();
             if (traintest==1)
                        xTr=0.25+(obj-1)*0.12;
             if (traintest==0)
                   {xTr=0.25+(obj-1)*getRand(25);
                   if (xTr>0.73)
                         xTr=0.73;
                   }
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffusesphere);
                                        //Change the material of the object
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambientsphere);

if (draw==1)            // Draw a box
{
             zTr=0.3+newline;
             yTr=0.435+ysc/2.0;
             glTranslated(xTr, yTr, zTr);  // Translate object
                                                   to the new location
             glRotated(0, 1.0, 0.0, 0.0);  // Roatete the object
                                                   at the new location
             glRotated(yawobj, 0.0, 1.0, 0.0);
             glRotated(0, 0.0, 0.0, 1.0);
             xsc=0.025;              //Standard object length in the x-axis
             glScaled(xsc, ysc, zsc[temp2]);           // Scale the object
                                                      before drawing it
             glutSolidCube(1.0);
              if ((zsc[temp2]>=ysc)||(ysc<0.09))        // EDIT THIS IF YOU
                   WANT TO CHANGE WHEN THE GRIPPER CHANGES IT'S ORIENTATION
    The default is: If object's height is either less than 0.09 or less or
                  equal to the object's length then the gripper has no roll
       {xgr=xTr; ygr=yTr+ysc*0.4+0.04; zgr=zTr;  roll=0.0;   yaw=yawobj; }
                        // Gripper data for cathing the onject without roll
             if ((zsc[temp2]<ysc)&&(ysc>=0.09))        // EDIT THIS IF YOU
WANT TO CHANGE WHEN THE GRIPPER CHANGES IT'S ORIENTATION
{xgr=xTr-(zsc[temp2]*0.4+0.04)*sin(yawobj*3.14/180);        ygr=yTr;
zgr=zTr-(zsc[temp2]*0.4+0.04)*cos(yawobj*3.14/180);   roll=-90.0;
yaw=yawobj; }     // Gripper data for cathing the onject with roll
}

if (draw==2)            // Draw a cylinder
{
      xsc=0.015;  ysc=0.015;  zsc[1]=0.10;      // Standar lengths
                                                      //for the cylinder
      zTr=0.3+newline;
      yTr=0.435+zsc[1];
      yawobj=0.0;
      glTranslated(xTr, yTr, zTr);              // Translate object
                                                     //to the new location
      glRotated(90, 1.0, 0.0, 0.0);             // Rotate the object
                                                     at the new location
      glRotated(yawobj, 0.0, 1.0, 0.0);
      glRotated(0, 0.0, 0.0, 1.0);
      Cyli=gluNewQuadric();
      Disk=gluNewQuadric();
      glScaled(xsc, ysc, zsc[1]);
      gluCylinder( Cyli, 1.0, 1.0, 1.0, 30, 30 );
      gluDisk(Disk, 0.0, 1.0, 30, 30);
xgr=xTr-(ysc*0.4+0.04)*sin(yawobj*3.14/180);    ygr=yTr-zsc[1]/2.0;
zgr=zTr-(xsc*0.4+0.04)*cos(yawobj*3.14/180)+0.0041/2.0;     roll=-90.0;
yaw=yawobj;       // gripper for the cylinder
}
```

```
if (draw==3)            // Draw a glass
{
      xsc=0.025;  ysc=0.025;  zsc[1]=0.08;            // Standar
                                                  //lengths for the cylinder
      zTr=0.3+newline;
      yTr=0.435+zsc[1];
      yawobj=0.0;
      glTranslated(xTr, yTr, zTr);                  // Translate
                                                //object to the new location
      glRotated(90, 1.0, 0.0, 0.0);                 // Rotate the
                                                //object at the new location
      glRotated(yawobj, 0.0, 1.0, 0.0);
      glRotated(0, 0.0, 0.0, 1.0);
      Cyli=gluNewQuadric();
      Disk=gluNewQuadric();
      glScaled(xsc, ysc, zsc[1]);
      gluCylinder( Cyli, 0.7, 0.5, 1.0, 30, 30 );
      glTranslated(0, 0, 0.98);
      gluDisk(Disk, 0.0, 0.7, 30, 30);
xgr=xTr-(ysc*0.4+0.04)*sin(yawobj*3.14/180);    ygr=yTr-zsc[1]/2.0;
zgr=zTr-(xsc*0.4+0.04)*cos(yawobj*3.14/180);    roll=-90.0; yaw=yawobj;
      // gripper για glass
}
glPopMatrix();

       glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);     // Reset the
                                    //correct material for the environment
       glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);

      glPushMatrix();         // Create the surounding walls
            wall(0.25);
      glPopMatrix();
      glPushMatrix();
            glRotated(90, 0, 0.0, 1.0);
            wall(0.25);
      glPopMatrix();
      glPushMatrix();
            glRotated(-90, 1.0, 0, 0);
            wall(0.25);
      glPopMatrix();
      glPushMatrix();         // Create the table of the environment
            glTranslated(0.5, 0, 0.5);
            table(0.9, 0.02, 0.04, 0.3);
      glPopMatrix();

      xgripper=xgr;
      ygripper=ygr;
      zgripper=zgr;

     glPushMatrix();
            glTranslated(xgr, ygr, zgr);        // Translate gripper to
                                    the correct location to grab the object
            glRotated(yaw, 0.0, 1.0, 0.0);      // Rotate the gripper at
                                                    the new location
            glRotated(roll, 1.0, 0.0, 0.0);
            glRotated(pitch, 0.0, 0.0, 1.0);
            gripper(0.05, 0.01, 0.05, 0.005, 0.05);   // Draw the gripper
      glPopMatrix();
      glutSwapBuffers();            // Swap buffers to display the results
                                                at the main gfx window
       if (automatic == 1) {        // When the train or test button is
```

```
                                                    pressed start creating data
            glReadBuffer(GL_BACK);
            glPixelStorei(GL_PACK_ALIGNMENT, 1);
            imageData=i_ptr;
            glReadPixels(0, 0 , screenWidthi , screenHeighti , GL_RGB ,
GL_UNSIGNED_BYTE , i_ptr);    // Take a "print-screen" of the main
                                                                  gfx window
      X = mxCreateDoubleMatrix(screenHeighti, screenWidthi, mxREAL);
      // Create arrays to pass data to matlab
      Y = mxCreateDoubleMatrix(screenHeighti, screenWidthi, mxREAL);
      Z = mxCreateDoubleMatrix(screenHeighti, screenWidthi, mxREAL);
      M = mxCreateDoubleMatrix(1, 1, mxREAL);
      RED = (double *)mxGetPr(X);
      GREEN = (double *)mxGetPr(Y);
      BLUE = (double *)mxGetPr(Z);
      MODE = (double *)mxGetPr(M);
      MODE[0] = traintest;
      for (int o=0, k=0; o<screenWidthi; o++){        // Get the RGB
                                                      values of each pixel
      for (int p=0; p<screenHeighti; p++){
      RED[o*screenHeighti+p]=imageData[nSize-(p+1)*screenWidthi*3+k];
      GREEN[o*screenHeighti+p]=imageData[nSize-(p+1)*screenWidthi*3+k+1];
      BLUE[o*screenHeighti+p]=imageData[nSize-(p+1)*screenWidthi*3+k+2]
                        }
                        k=k+3;
                  }
      engPutVariable(matlab, "X", X);     // Place variables into MATLAB
                                                            workspace
      engPutVariable(matlab, "Y", Y);
      engPutVariable(matlab, "Z", Z);
      engPutVariable(matlab, "traintest", M);
      engEvalString(matlab, "RGB(:,:,1)=[X];"); // Create in MATLAB an
                                              array with all the RGB values
      engEvalString(matlab, "RGB(:,:,2)=[Y];");
      engEvalString(matlab, "RGB(:,:,3)=[Z];");
      engEvalString(matlab, "RGB=uint8(RGB);"); // Convert RGB values to
                                                      uint8 - [0 , 255]
      engEvalString(matlab, "diplwmatikh(RGB, traintest);");
                           // Call a MATLAB function to create object data

      mxDestroyArray(X);      // Destroy used arrays to free memory for
                                                        the next object
      mxDestroyArray(Y);
      mxDestroyArray(Z);
      mxDestroyArray(M);

      if (traintest==1)       // Save gripper's training or testing data
            fp=fopen("c:\\traindata.txt","a+");
      if (traintest==0)
            fp=fopen("c:\\testdata.txt","a+");
            fprintf(fp,"%f\t", xgripper); fprintf(fp,"%f\t", ygripper);
            fprintf(fp,"%f\t", zgripper); fprintf(fp,"%f\t",roll);
            fprintf(fp,"%f\t",yaw); fprintf(fp,"%f\n",pitch);
            fclose(fp);
            obj=obj+1;        // Reset the parameters for the next object
            glutPostRedisplay();
       }
}
```

```
/*********************  Keyboard Function  ***********************/

void myKeyboard(unsigned char theKey, int mouseX, int mouseY)
{
      switch(theKey)
      {
       case 27:                        // ESC - Quit
             PostQuitMessage(0);
             break;
       default:
            break;
      }
}


/***********************  MAIN Function  ************************/

int main(int argc, char** argv)
{
  /**************************************/
  /*   Initialize GLUT and create window  */
  /**************************************/
    matlab = engOpen("\0"); // Open MATLAB
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(screenWidthi, screenHeighti);
    glutInitWindowPosition(450,0);
    int window1=glutCreateWindow("SPYROB v1.0");

    myInit();
    glutKeyboardFunc(myKeyboard);
    glutDisplayFunc(displaySolid);

  /**************************************/
  /*         Here's the GLUI code         */
  /**************************************/

glui = GLUI_Master.create_glui("Options", 0, 0, 0);          // Create GLUI
                                                                  window
glui->set_main_gfx_window(window1);                       // Register the main
                                                          gfx window with GLUT
GLUI_Master.set_glutIdleFunc(myGlutIdle);        // Register the idle
                                                 callback with GLUI, not with GLUT
objectlist = glui->add_listbox("Object: ", &draw, OBJECT_ID, control_cb);
// Create list of objects to draw
      objectlist->add_item(0, "");
      objectlist->add_item(1, "Box");
      objectlist->add_item(2, "Cylinder");
      objectlist->add_item(3, "Glass");
objectheight=glui->add_edittext("Object Height = ", GLUI_EDITTEXT_FLOAT,
&ysc);            // Edit text for the height of the object
      objectheight->set_float_limits(0.025,0.20,GLUI_LIMIT_CLAMP);
trainpanel = glui->add_panel("Grid", GLUI_PANEL_EMBOSSED);         // Panel
                                                          for the grid options
      trainrows=glui->add_edittext_to_panel(trainpanel, "Rows = ",
GLUI_EDITTEXT_INT, &rows, TRAIN_ROWS_ID, control_cb); // Edit text for the
                                                            rows of the grid
      trainrows->set_int_limits(1,7,GLUI_LIMIT_CLAMP);
       glui->add_statictext_to_panel(trainpanel, "Number of objects in:");
                                  // Edit text for the objects in each row
```

```
            trainrow[1]=glui->add_edittext_to_panel(trainpanel, "Row 1 =
", GLUI_EDITTEXT_INT, &objectsrow[1]);
            trainrow[2]=glui->add_edittext_to_panel(trainpanel, "Row 2 =
", GLUI_EDITTEXT_INT, &objectsrow[2]);
            trainrow[3]=glui->add_edittext_to_panel(trainpanel, "Row 3 =
", GLUI_EDITTEXT_INT, &objectsrow[3]);
            trainrow[4]=glui->add_edittext_to_panel(trainpanel, "Row 4 =
", GLUI_EDITTEXT_INT, &objectsrow[4]);
            trainrow[5]=glui->add_edittext_to_panel(trainpanel, "Row 5 =
", GLUI_EDITTEXT_INT, &objectsrow[5]);
            trainrow[6]=glui->add_edittext_to_panel(trainpanel, "Row 6 =
", GLUI_EDITTEXT_INT, &objectsrow[6]);
            trainrow[7]=glui->add_edittext_to_panel(trainpanel, "Row 7 =
", GLUI_EDITTEXT_INT, &objectsrow[7]);
            for (int i=1; i<=7; i++){
                  trainrow[i]->set_int_val(trainrows->get_int_val());
                  trainrow[i]->set_int_limits(1,7,GLUI_LIMIT_CLAMP);
                  trainrow[i]->disable();
            }
            trainrow[1]->enable();
      glui->add_column_to_panel(trainpanel, true);
                                    // Create new column in the grid panel
glui->add_statictext_to_panel(trainpanel, "Number of object's lengths: ");
            ztrains=glui->add_edittext_to_panel(trainpanel, "",
GLUI_EDITTEXT_INT, NULL, ZTRAINS_ID, control_cb);
                        // Edit text for the number of lengths of the obgect
            ztrains->set_int_limits(1,7,GLUI_LIMIT_CLAMP);
            for (int i=1; i<=7; i++){
                  ztrain[i]=glui->add_edittext_to_panel(trainpanel,
"length = ", GLUI_EDITTEXT_FLOAT, &zsc[i]);
                  ztrain[i]->set_float_limits(0.03,0.20,GLUI_LIMIT_CLAMP);
                  ztrain[i]->disable();
            }
            ztrain[1]->enable();

glui->add_column(false);
glui->add_button( "TRAIN", TRAIN_ENABLED_ID, control_cb);
                                          // Button to create train data
glui->add_button( "TEST", TEST_ENABLED_ID, control_cb);
                                           // Button to create test data
glui->add_button( "Quit", QUIT_ID, control_cb );
                                                        // Quit button
    glutMainLoop();                       // Regular Glut Main Loop
}
```

# ΠΑΡΑΡΤΗΜΑ Β

```
function diplwmatikh(I,traintest)
    j = size(I);              % Υπολογισμός μεγέθους εικόνας
    HSL=rgb2hsl(I);           % Μετατροπή του χρωματικού χώρου της εικόνας
                              % από το χώρο RGB στο χώρο HSL
    for k = 1:j(1)            % Υπολογισμός για όλες τις τετμημένες
        for l = 1:j(2)        % Υπολογισμός για όλες τις τεταγμένες
            if HSL(k,l,1)> 0.13 && HSL(k,l,1) < 0.18;
               d(k,l) = 1;  % Ορισμός του εικονοστοιχείου με χρώμα άσπρο
            else
               d(k,l) = 0;  % Ορισμός του εικονοστοιχείου με χρώμα μαύρο
                end
            end
        end

    [label,num] = bwlabeln (d);  % Εύρεση όλων των ιδιοτήτων όλων των
περιοχών που μας ενδιαφέρουν
    statsArea = regionprops (label, 'Area');
    statsCentroid = regionprops (label, 'Centroid');
    statsOrientation = regionprops (label, 'Orientation');
    statsMaxAxis = regionprops (label, 'MajorAxisLength');
    statsMinAxis = regionprops (label, 'MinorAxisLength');
    statsEcce = regionprops (label, 'Eccentricity');

    k=1;
    for i=1:num            % Εύρεση της περιοχής με το μεγαλύτερο εμβαδόν
        if (statsArea(i).Area > 500)
            biggrain(k) = i;
            k=k+1;
        end
    end

    % Αποθήκευση όλων των ιδιοτήτων του αντικειμένου στα αντίστοιχα αρχεία
    if (traintest==1)
        fid = fopen('c:\\traindata.txt', 'a+');
    else
        fid = fopen('c:\\testdata.txt', 'a+');
    end

    for i=1:max(size(biggrain))
        x0(i)=statsCentroid(biggrain(i)).Centroid(:,1);
        y0(i)=statsCentroid(biggrain(i)).Centroid(:,2);
        el(i)=statsOrientation(biggrain(i)).Orientation(1,1);
        if (el(i)<0)
            el(i)=el(i)+180;
        end
        maxax(i)=statsMaxAxis(biggrain(i)).MajorAxisLength(1,1);
        minax(i)=statsMinAxis(biggrain(i)).MinorAxisLength(1,1);
```

```
        factor(i)=maxax(i)/minax(i);
        ecce(i)=statsEcce(biggrain(i)).Eccentricity;
        fprintf(fid,'%f\t',x0(i));
        fprintf(fid,'%f\t',y0(i));
        fprintf(fid,'%f\t',el(i));
        fprintf(fid,'%f\t',maxax(i));
        fprintf(fid,'%f\t',minax(i));
        fprintf(fid,'%f\t',factor(i));
        fprintf(fid,'%f\t',ecce(i));
    end
    fclose(fid);
  end



function [h,s,l]=rgb2hsl(r,g,b)

% RGB2HSL Convert red-green-blue colors to hue-saturation-luminance.
%   H = RGB2HSL(M) converts an RGB color map to an HSL color map.
%   Each map is a matrix with any number of rows, exactly three columns,
%   and elements in the interval 0 to 1.  The columns of the input matrix,
%   M, represent intensity of red, blue and green, respectively.  The
%   columns of the resulting output matrix, H, represent hue, saturation
%   and color luminance, respectively.
%
%   HSL = RGB2HSL(RGB) converts the RGB image RGB (3-D array) to the
%   equivalent HSL image HSL (3-D array).
%
%   CLASS SUPPORT
%   -------------
%   If the input is an RGB image, it can be of class uint8, uint16, or
%   double; the output image is of class double.  If the input is a
%   colormap, the input and output colormaps are both of class double.
%
%   See also HSL2RGB, COLORMAP, RGBPLOT.
%
%   Undocumented syntaxes:
%   [H,S,L] = RGB2HSL(R,G,B) converts the RGB image R,G,B to the
%   equivalent HSL image H,S,L.
%
%   HSL = RGB2HSL(R,G,B) converts the RGB image R,G,B to the
%   equivalent HSL image stored in the 3-D array (HSL).
%
%   [H,S,L] = RGB2HSL(RGB) converts the RGB image RGB (3-D array) to
%   the equivalent HSL image H,S,L.
%
%  Armin Ghorssi Anbaran, Mohammad Afshar
%    Department of Electrical Engineering
%    Amir Kabir University of Technology - Tehran - Iran
%  Revision: 1.0   Date:  June 25,2003

switch nargin
    case 1,
        if isa(r, 'uint8'),
            r = double(r) / 255;
        elseif isa(r, 'uint16')
            r = double(r) / 65535;
        end
    case 3,
        if isa(r, 'uint8'),
            r = double(r) / 255;
```

```
        elseif isa(r, 'uint16')
            r = double(r) / 65535;
        end

        if isa(g, 'uint8'),
            g = double(g) / 255;
        elseif isa(g, 'uint16')
            g = double(g) / 65535;
        end

        if isa(b, 'uint8'),
            b = double(b) / 255;
        elseif isa(b, 'uint16')
            b = double(b) / 65535;
        end

    otherwise,
        error('Wrong number of input arguments.');
end

threeD = (ndims(r)==3); % Determine if input includes a 3-D array

if threeD,
    g = r(:,:,2); b = r(:,:,3); r = r(:,:,1);
    siz = size(r);
    r = r(:); g = g(:); b = b(:);
elseif nargin==1,
    g = r(:,2); b = r(:,3); r = r(:,1);
    siz = size(r);
else
    if ~isequal(size(r),size(g),size(b)),
        error('R,G,B must all be the same size.');
    end
    siz = size(r);
    r = r(:); g = g(:); b = b(:);
end

mx = max(max(r,g),b);   mn = min(min(r,g),b);

%  Luminance
l = (mn+mx)/2;
s = zeros(size(l));
h = zeros(size(l));

% Saturation
i = find( mx~=mn );
s(i) = ( mx(i)-mn(i) )./( mx(i));
i = find( mx==mn );
s(i) = 0;

%Hue
i = find( mx==r & mx~=mn );%Red is the max color
h(i) = ((g(i)-b(i))./(mx(i)-mn(i))  )/6;
i = find( mx == g & mx~=mn);%Green is the max color
h(i) = (2+(b(i)-r(i))./(mx(i)-mn(i)))/6;
i = find( mx== b & mx~=mn);%Blue is the max color
h(i) = ( 4+ (r(i)-g(i))./(mx(i)-mn(i)))/6;

i = find(h<0);    h(i) = h(i)+1;
```

```
if nargout<=1,
  if (threeD | nargin==3),
    h = reshape(h,siz);
    s = reshape(s,siz);
    l = reshape(l,siz);
    h=cat(3,h,s,l);
  else
    h=[h s l];
  end
else
  h = reshape(h,siz);
  s = reshape(s,siz);
  l = reshape(l,siz);
end
```

# ΠΑΡΑΡΤΗΜΑ Γ

```
clear all
close all

% Άνοιγμα του αρχείου με τα δεδομένα
fp = fopen('c:\\traindata.txt', 'r');
    in=fscanf(fp, '%f %f %f %f %f %f %f %f %f %f %f %f %f', [13 inf]);
    A=in';
    input=A(:,1:7)';
    output=A(:,8:11)';
fclose(fp);

% Κανονικοποίηση των δεδομένων εισόδου και εξόδου στο διάστημα [-1,1]
[inputstad,PSin] = mapminmax(input);
[outputstad,PSout] = mapminmax(output);

% Δημιουργία και εκμάθηση Νευρωνικού Δικτύου
net = newff(inputstad,outputstad,[15 10],{'tansig','tansig'});
net.trainParam.goal = 5*10^(-7);
net.trainParam.max_fail=6;
[net,tr]=train(net,inputstad,outputstad);


% Εμφάνιση αποτελεσμάτων
fp3 = fopen('c:\\testdata.txt', 'r');
in2=fscanf(fp3, '%f %f %f %f %f %f %f %f %f %f %f %f %f', [13 inf]);
B=in2';
testinput=B(:,1:7)';
testoutput=B(:,8:13)';
fclose(fp3);

testinputstad=mapminmax ('apply',testinput,PSin);
yteststad = sim(net,testinputstad);
ytest=mapminmax ('reverse',yteststad,PSout);

for i=1:max(size(testoutput))
    xerror(i) = ytest(1,i)-testoutput(1,i);
    yerror(i) = ytest(2,i)-testoutput(2,i);
    zerror(i) = ytest(3,i)-testoutput(3,i);
    rollerror(i) = ytest(4,i)-testoutput(4,i);
end

figure;
    hold on;
    for i=1:max(size(testoutput))
       if ((xerror(i)>=0.01)||(xerror(i)<=-0.01))
          plot(testoutput(1,i),ytest(1,i), 'Color', 'Red', 'Marker', 'o');
       else
         plot(testoutput(1,i),ytest(1,i), 'Color', 'Blue', 'Marker', 'o');
       end
    end
```

```
        plot([min(ytest(1,:)) max(ytest(1,:))],[min(ytest(1,:))
max(ytest(1,:))],'--');
        xlabel('Target'); ylabel('Output'); title('x-gripper');
        hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
       if ((yerror(i)>=0.01)||(yerror(i)<=-0.01))
         plot(testoutput(2,i),ytest(2,i), 'Color', 'Red', 'Marker', 'o');
       else
         plot(testoutput(2,i),ytest(2,i), 'Color', 'Blue', 'Marker', 'o');
       end
    end
        plot([min(ytest(2,:)) max(ytest(2,:))],[min(ytest(2,:))
max(ytest(2,:))],'--');
        xlabel('Target'); ylabel('Output'); title('y-gripper');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
       if ((zerror(i)>=0.01)||(zerror(i)<=-0.01))
         plot(testoutput(3,i),ytest(3,i), 'Color', 'Red', 'Marker', 'o');
       else
         plot(testoutput(3,i),ytest(3,i), 'Color', 'Blue', 'Marker', 'o');
       end
    end
        plot([min(ytest(3,:)) max(ytest(3,:))],[min(ytest(3,:))
max(ytest(3,:))],'--');
        xlabel('Target'); ylabel('Output'); title('z-gripper');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
       if ((rollerror(i)>=4)||(rollerror(i)<=-4))
         plot(testoutput(4,i),ytest(4,i), 'Color', 'Red', 'Marker', 'o');
       else
         plot(testoutput(4,i),ytest(4,i), 'Color', 'Blue', 'Marker', 'o');
       end
    end
        plot([min(ytest(4,:)) max(ytest(4,:))],[min(ytest(4,:))
max(ytest(4,:))],'--');
        xlabel('Target'); ylabel('Output'); title('Roll');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
        if ((xerror(i)>=0.01)||(xerror(i)<=-0.01))
            plot(i,xerror(i), 'Color', 'Red', 'Marker', 'o');
        else
            plot(i,xerror(i), 'Color', 'Blue', 'Marker', 'o');
        end
    end
    plot([0,max(size(testoutput))],[0.01,0.01],'--');
    plot([0,max(size(testoutput))],[-0.01,-0.01],'--');
    axis([0 max(size(testoutput)) -0.012 0.012]);
    xlabel('Object'); ylabel('Error'); title('Error x-gripper');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
        if ((yerror(i)>=0.01)||(yerror(i)<=-0.01))
```

```
            plot(i,yerror(i), 'Color', 'Red', 'Marker', 'o');
        else
            plot(i,yerror(i), 'Color', 'Blue', 'Marker', 'o');
        end
    end
    plot([0,max(size(testoutput))],[0.01,0.01],'--');
    plot([0,max(size(testoutput))],[-0.01,-0.01],'--');
    axis([0 max(size(testoutput)) -0.012 0.012]);
    xlabel('Object'); ylabel('Error'); title('Error y-gripper');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
        if ((zerror(i)>=0.01)||(zerror(i)<=-0.01))
            plot(i,zerror(i), 'Color', 'Red', 'Marker', 'o');
        else
            plot(i,zerror(i), 'Color', 'Blue', 'Marker', 'o');
        end
    end
    plot([0,max(size(testoutput))],[0.01,0.01],'--');
    plot([0,max(size(testoutput))],[-0.01,-0.01],'--');
    axis([0 max(size(testoutput)) -0.012 0.012]);
    xlabel('Object'); ylabel('Error'); title('Error z-gripper');
    hold off;
figure;
    hold on;
    for i=1:max(size(testoutput))
        if ((rollerror(i)>=4)||(rollerror(i)<=-4))
            plot(i,rollerror(i), 'Color', 'Red', 'Marker', 'o');
        else
            plot(i,rollerror(i), 'Color', 'Blue', 'Marker', 'o');
        end
    end
    plot([0,max(size(testoutput))],[4,4],'--');
    plot([0,max(size(testoutput))],[-4,-4],'--');
    axis([0 max(size(testoutput)) -5 5]);
    xlabel('Object'); ylabel('Error'); title('Error Roll');
    hold off;
```