Πανεπιστήμιο Αιγαίου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Ρομποτικός Έλεγχος

Διδάσκων: Καβαλλιεράτου Εργίνα

Τελική Εργασία

Icsd14215 Νικόλαος Εμμανουήλ Χαλβατζάκης

Σάμος, 20 Ιανουαρίου 2021

Contents

1 Εισαγωγή

Η χρήση μηχανημάτων και του αυτοματισμού έχουν εισαχθεί εδώ και πολλά χρόνια στις βιομηχανίες. Η χειρωνακτική εργασία όμως εξακολουθεί να είναι σημαντική σε αρκετούς τομείς των βιομηχανιών. Η έρευνα για τους ρομποτικούς εξωσκελετούς έχει αναπτυχθεί γοργά τις τελευταίες δεκαετίες για να απαντήσει σε θέματα αυτοματοποίησης της χειρωνακτικής εργασίας. Αυτό οφείλεται στην υψηλή αποτελεσματικότητα της χρήσης τους για την διευκόλυνση των εργαζόμενων που αντιμετωπίζουν βαρέα. Οι εξωσκελετοί μπορούν να μειώσουν την κόπωση, να επεκτείνουν τη λειτουργικότητα των εργαζομένων και να μειώσουν την πιθανότητα του τραυματισμού.

Σε αυτή την εργασία θα μελετήσουμε τρία επιστημονικά άρθρα που προτείνουν ρομποτικούς εξωσκελετούς στα κάτω άκρα με σκοπό βελτίωση των συνθηκών χειρωνακτικής εργασίας στις βιομηχανίες.

Πριν από αυτό όμως είναι σημαντικό να αναφέρουμε το state of the art των εξωσκελετών κάτω άκρων ώστε να έχουμε μια γενική εικόνα στο πως έχει μορφωποιηθεί η αγορά/έρευνα με βάση τις ανάγκες που υπάρχουν.

1.1 State of the Art

Οι εξωσκελετοί μπορούν να χωριστούν σε τρεις βασικές κατηγορίες (1):

1. Συστήματα επαύξησης ανθρώπινης απόδοσης
2. Βοηθητικοί εξωσκελετοί για άτομα με ειδικές ανάγκες
3. Εξωσκελετοί για θεραπευτικούς σκοπούς

Για τους σκοπούς της εργασίας θα επικεντρωθούμε στην πρώτη κατηγορία.

Το 2000, η Υπηρεσία Προηγμένων Αμυντικών Ερευνητικών Προγραμμάτων (**DARPA**) ξεκίνησε ένα πρόγραμμα για τη χρηματοδότηση ρομποτικών εξωσκελετών. Το πρόγραμμα αυτό (Exoskeletons for Human Performance Augmentation, **EHPA**) επικεντρώθηκε στην προσαύξηση των δυνατοτήτων του πεζικού στρατού. Ανάμεσα στις ερευνητικές ομάδες δύο ήταν αυτές που ξεχώρισαν.

Ο Berkeley Lower Extremity Exoskeleton (**BLEEX**) είχε σκοπό την δημιουργία φορητών εξωσκελετών που θα επέτρεπαν στους στρατιώτες να κουβαλάνε βαρέα φορτία. Ο μείζων περιορισμός όμως ήταν στο μεγάλο βάρος του ίδιου το εξωσκελετού και στον περιορισμό των κινήσεων που μπορούσαν να πραγματοποιηθούν.

Ο δεύτερος εξωσκελετός που αναδεικνύεται από το πρόγραμμα EHPA είναι ο εξωσκελετός της εταιρίας **Sarcos**(ή αργότερα **XOS**). Ο εξωσκελετός της Sarcos ήταν ένα ολόσωμο κουστούμι που υποστήριζε όλα τα άκρα του χρήστη και με την πρόθεση να ενισχύσει την δύναμη του. Όμως η υψηλή απαίτηση ενέργειας του εξωσκελετού κόστισε τη φορητότητά του.

Η Lockheed Martin ήταν μια από τις πιο σημαντικές εταιρίες ανάπτυξης εξωσκελετών για την επαύξηση της ανθρώπινης απόδοσης. Απέκτησαν την τεχνολογία HULC (Human Universal Load Carrier) από την Berkeley Bionics που ήταν βασισμένο στον σχεδιασμό του BLEEX. Από δοκιμές του Αμερικανικού στρατού αναδείχθηκε πως ο εξωσκελετός είχε αλλάξει τα χαρακτηριστικά του βαδίσματος των στρατιωτών, αύξησε την δαπάνη της μεταβολικής τους ενέργειας και μείωσε την κινητικότητα τους. Ύστερα ανακοίνωσαν το **FORTIS**, ένας εξωσκελετός σχεδιασμένος συγκεκριμένα για την βιομηχανική χρήση σε ναυπηγεία. Το FORTIS επέτρεπε τους χρήστες να χειριστούν εύκολα βαριά εργαλεία. Σε αντίθεση με το HULC το FORTIS δεν χρειάζεται πολύ ενέργεια και μπορεί να μεταφέρει το φορτίο στο έδαφος, με αποτέλεσμα να μειώσει την κόπωση και να βελτιώσει την ασφάλεια.

Αξιοσημείωτος επίσης είναι ο εξωσκελετός **X1** ο οποίος δημιουργήθηκε με την συνεργασία της NASA και του Ινστιτούτου Human and Machine Cognition στην Pensacola, Florida. Προοριζόταν ως πιθανό μέσο εκπαίδευσης αστροναυτών στο διάστημα για την πρόληψη της μυϊκής κόπωσής και της οστεοπενίας. Η συσκευή μπορεί να διαμορφωθεί σε λειτουργία ελέγχου δύναμης επιτρέποντας τόσο εκκεντρικές όσο και ομόκεντρες ασκήσεις.

Ένα σύγχρονο και πρωτοποριακό σχέδιο προέρχεται από το Ινστιτούτο του Wyss. Ο Δρ. Conor Walsh και η ομάδα του σχεδίασαν και ανέπτυξαν ελαφριές φορητές (**wearable**) συσκευές που μπορούν να βοηθήσουν την ανθρώπινη κίνηση. Οι εξωσκελετοί αυτοί είναι αρκετά διαφορετικοί από τους κλασσικούς εξωσκελετούς με βαριά μέταλλα και κινητήρες.

Τώρα που είδαμε το State of the Art για εξωσκελετούς επαύξησης, θα αναλύσουμε τρία άρθρα που έχουν ως σκοπό την μοντελοποίηση τέτοιων εξωσκελετών επικεντρώνοντας **στα κάτω άκρα.**

1 Στα αριστερά, ο εξωσκελετός FORTIS. Στην μέση ο εξωσκελετός X1 από την NASA και IHMC. Στα δεξιά ο εξωσκελετός από το Ινστιτούτου του Wyss.

2 Μοντελοποίηση Συστήματος Ελέγχου Αλληλεπίδρασης Ανθρώπου– Μηχανής

2.1 Εισαγωγή

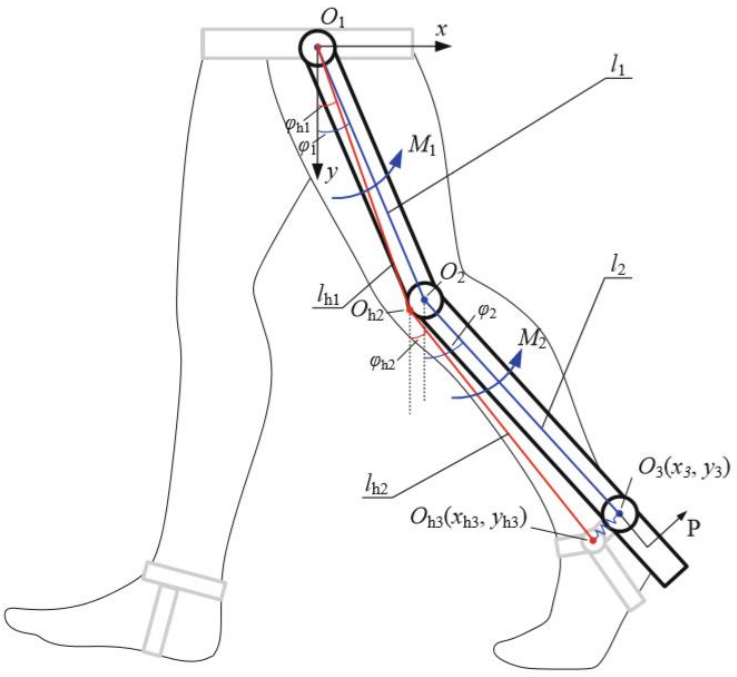
Το πρώτο επιστημονικό άρθρο επικεντρώνεται στην δημιουργία ενός συστήματος ελέγχου για τους εξωσκελετούς στα πόδια. Προτείνεται μια μέθοδος η οποία εφαρμόζει τον έλεγχο αντιγραφής drive της συσκευής (Copy Control System, **CCS**) που θα διασφαλίζει την τον συγχρονισμό στην κίνηση του χειριστή και του εξωσκελετού.

Το μοντέλο που παρουσιάζεται είναι συνεργατικής φύσης μεταξύ του χρήστη και του εξωσκελετού για την κατάλληλη βελτιστοποίηση των παραμέτρων στο σύστημα και την ζώνη στην οποία στέκεται το πόδι.

2.2 Μαθηματικό Μοντέλο

2.2.1 Δομή Εξωσκελετού

Επειδή η δημιουργία ενός CCS είναι αρκετά περίπλοκο, οι ερευνητές περιορίζουν την έρευνα στην μελέτη της κίνησης ενός ποδιού του εξωσκελετού σε μία κυλιόμενη επιφάνεια. Το διάγραμμα για μια τέτοια συσκευή φαίνεται παρακάτω στην εικόνα 2.



2 Σχήμα του αντικειμένου της έρευνας

Στο σχήμα: **L1** τμήμα του εξωσκελετού παράλληλα στο μηριαίο οστό , **L2**: συνεχίζει στο κάτω μέρος του ποδιού.

Lh1, **Lh2**: τμήματα μεταξύ τις αρθρώσεις του χειριστή, μεταξύ της άρθρωσης στο γόνατο **Oh2** και του αστραγάλου **Oh3** και του ισχίου (μπούτι) **O1**

O1, **O2**, **O3**: ισχίο, γόνατο, αστράγαλος αντίστοιχα.

φ1, **φ2**: οι γωνίες του εξωσκελετού στις οποίες μπορεί να περιστραφεί

φh1, **φh2**: οι γωνίες που μπορεί να περιστραφεί το ισχίο και το κάτω πόδι του χειριστή

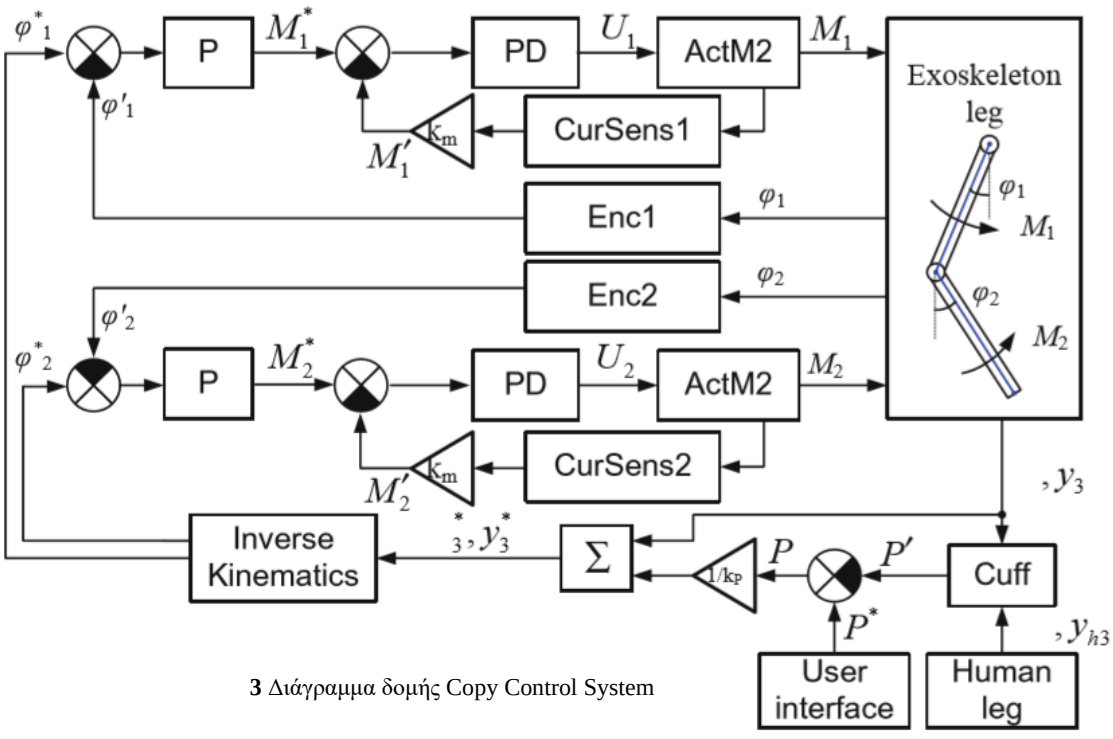
Η κίνηση του χρήστη πραγματοποιείτε από τις μυϊκές δυνάμεις και δεν εξαρτώνται από την θέση του εξωσκελετού. Ο εξωσκελετός συνδέει τις κινήσεις υπό την ενέργεια των ροπών **M1**, **M2** που αναπτύσσονται από το drive του εξωσκελετού.

Το πόδι του εξωσκελετού είναι σταθερό στο σημείο **O1** που συνυπάρχει με το σημείο αναφοράς του σταθερού πλαισίου.

Το πόδι του χειριστή και οι σύνδεσμοι του εξωσκελετού συνδέονται μέσω μίας ζώνη που περιέχει αισθητήρα μέτρησης δύο συντεταγμένων. Η ζώνη περιέχει ελαστικά στοιχεία που καθορίζουν την δύναμη της αλληλεπίδρασης **P.**

Το έργο του ελέγχου περιορίζεται από την απαιτούμενη δύναμη αλληλεπίδρασης που προέρχεται από την ταυτόχρονη κίνηση του εξωσκελετού και των άκρων. Για να τη πετύχουμε χρησιμοποιούμε το Copy Control System, του οποίου διάγραμμα δομής φαίνεται στην εικόνα 3.

2.2.2 Σύστημα Copy Control



3 Διάγραμμα δομής Copy Control System

Κατά τη κίνηση του χειριστή, το πόδι έχει αντίσταση από τον εξωσκελετό, η οποία καταγράφεται από το αισθητήρα μέτρησης της ζώνης. Η μετρημένη αντίσταση μεταξύ του ποδιού του χειριστή **P** και του εξωσκελετού **P'** συγκρίνεται με την τιμή **P*** η οποία έχει καθοριστεί από τον χειριστή με βάση τις προτιμήσεις του.

Η δύναμη αυτή προστίθεται μετά από κατάλληλη παραμόρφωση στις συντεταγμένες του **O3** (αστράγαλος).

Αφού λοιπόν καθοριστούν οι νέες συντεταγμένες τότε μέσω Inverse Kinematics δημιουργούνται οι σωστές τιμές για την περιστροφή των **φ*1**, **φ*2** (ισχίο γόνατο).

Αυτές οι γωνίες μέσω των κωδικοποιητών (**Enc1**, **Enc2**) του συστήματος συγκρίνονται με τις τρέχων γωνίες **φ'1 φ'2** και ύστερα περνάνε στο σύστημα για να ξεκινήσει η κίνηση. Τέλος, οι error τιμές που έχουμε πάρει στέλνονται στον βρόχο ελέγχου ροπής όπου τα κανάλια ανατροφοδότησης εφαρμόζουν τους τρέχων αισθητήρες (**CurSens1**, **CurSens2**) στις περιελίξεις του οπλισμού[1] (**ActM1**, **ActM2**)

Η υλοποίηση αυτή περιγράφεται με τις παρακάτω μαθηματικές εκφράσεις:

$$U_i^k = k_{pM}\left(M_i^{\,*} - M_i'\right)_k + k_{dM}\left(\frac{\left(M_i^{\,*} - M_i'\right)_k - \left(M_i^{\,*} - M_i'\right)_{k-1}}{\Delta t}\right), \qquad (1)$$

$$\phi_1 = \begin{cases} arctng\left(\frac{y_3}{x_3}\right) + \arccos\left(\frac{l_1^2 - l_2^2 + (x_3^2 + y_3^2)}{2l_1\sqrt{x_3^2 + y_3^2}}\right) & \text{if } (x_3 > 0) \\ arctng\left(\frac{y_3}{x_3}\right) + \arccos\left(\frac{l_1^2 - l_2^2 + (x_3^2 + y_3^2)}{2l_1\sqrt{x_3^2 + y_3^2}}\right) - \pi & \text{if } (x_3 \leq 0) \end{cases}, \qquad (2)$$

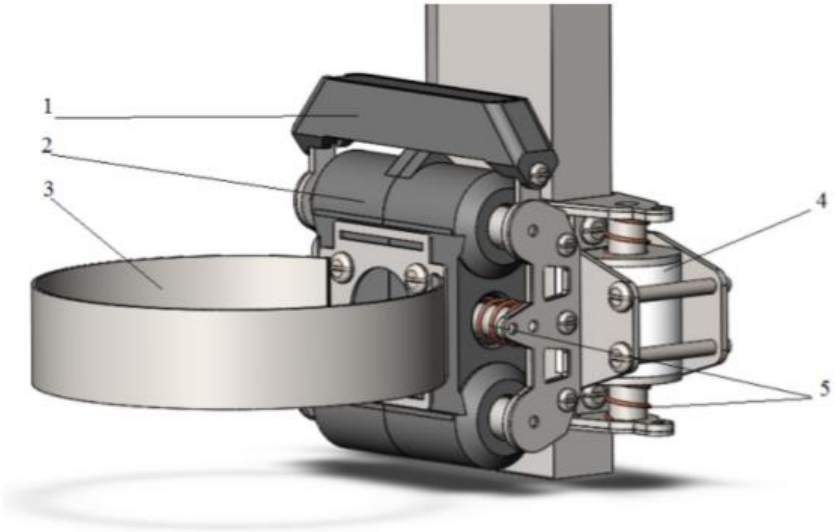
$$\phi_2 = \phi_1 + \left(\pi + \arccos\left(\frac{l_1^2 + l_2^2 - (x_3^2 + y_3^2)}{2l_1 l_2}\right)\right). \qquad (3)$$

Χρησιμοποιώντας τις εξισώσεις (2) και (3) μπορούμε να βρούμε τις απόλυτες γωνίες περιστροφής που θα κινηθεί το σημείο **O3** στα χαμηλότερα τερταμόρια του επιπέδου.

[1] Armature - Μετάφραση από αγγλικά - Στην ηλεκτρολογία, ο οπλισμός είναι το συστατικό μιας ηλεκτρικής μηχανής που φέρει εναλλασσόμενο ρεύμα. Οι περιελίξεις οπλισμού μεταφέρουν εναλλασσόμενο ρεύμα ακόμη και σε μηχανήματα συνεχούς ρεύματος, λόγω της μετακίνησης ή λόγω ηλεκτρονικής μετατροπής, όπως σε κινητήρες DC χωρίς ψήκτρες (**Wikipedia**)

2.2.3 Ζώνη με Μετρητή Δύο Συντεταγμένων

Για να βρούμε τις επιθυμητές από τον χειριστή συντεταγμένες $\mathbf{x}^*_3$, $\mathbf{y}^*_3$ θα χρειαστούμε τον μετρητή στην ζώνη.



4 Μετρητής Δύο Συντεταγμένων

1. Αισθητήρας Μετατόπισης
2. Κινητή βάση στήριξης (Οριζόντια)
3. Ελαστικά υφάσματα για την προσάρτηση του άκρου
4. Κινητή βάση στήριξης (Κάθετα)
5. Ελατήρια για την ανάρτηση της ζώνης

Κατά τη κίνηση, ο χειριστής δρα μέσω των ιμάντων της ζώνης (3) στον αισθητήρα μέτρησης ο οποίος αποτελείτε από δύο κινητά σταθερά ελατήρια (5) τις βάσεις (2) και (4) που κινούνται αντίστοιχα οριζόντια ή κάθετα.

Οι μετατοπίσεις καταγράφονται από κατάλληλους αισθητήρες. Η ζώνη είναι στηρίζεται μέσω ενός συνδέσμου. Θεωρούμε πως η ζώνη είναι πάντα σωστά προσανατολισμένη στον δεσμό. Η θέση της καθορίζεται από τον προσανατολισμό του ποδιού το οποίο δεν μελετάται σε αυτό το άρθρο. Πρακτικά η γωνία περιστροφής της ζώνης είναι ανάλογη του δεσμού που μπορούμε να βρούμε εύκολα μέσω του αισθητήρα που έχουν τοποθετήσει στον σύνδεσμο.

Αν θεωρήσουμε τους συντελεστές της αντίσταστασης των ελατηρίων ($\mathbf{c}_{3}\mathbf{x}$, $\mathbf{c}_{3}\mathbf{y}$) ίσα και μετρώντας την ανάλογη μετατόπιση του σημείου σύνδεσης της ζώνης στο πόδι και του εξωσκελετού, μπορούμε να προσδιορίσουμε τη δύναμη της αντίστασης στην κίνηση ως εξής:

$$P = \sqrt{P_x^2 + P_y^2} = \sqrt{c_{3x}(x_{h3} - x_3)^2 + c_{3y}(y_{h3} - y_3)^2}, \qquad (4)$$

Όπου **Px**, **Py** είναι οι οριζόντιοι και κάθετοι συντελεστές της δύναμης μεταξύ του εξωσκελετού και του χειριστή.

Γενικά ο χρήστης μπορεί να θέσει ότι τιμή επιθυμεί για την δύναμη **P*** και σε προηγούμενο έργο των ερευνητών έχουν δείξει πως αυτό μπορεί να χρησιμοποιηθεί για την βοήθεια στο περπάτημα.

Σε αυτή τη περίπτωση όμως θα θεωρήσουμε ότι το **P*** είναι 0 δηλαδή ο εξωσκελετός θα κουνηθεί έτσι ώστε να ελαχιστοποιεί την αντίσταση του στη κίνηση του χειριστή

Ύστερα υπολογίζουμε το σημείο $\mathbf{O_3}$ σύμφωνα με την δύναμη που έχει ασκήσει ο χρήστης.

$$x_3^* = x_3 + k_P P \sin(\gamma), \quad y_3^* = y_3 + k_P P \cos(\gamma), \tag{5}$$

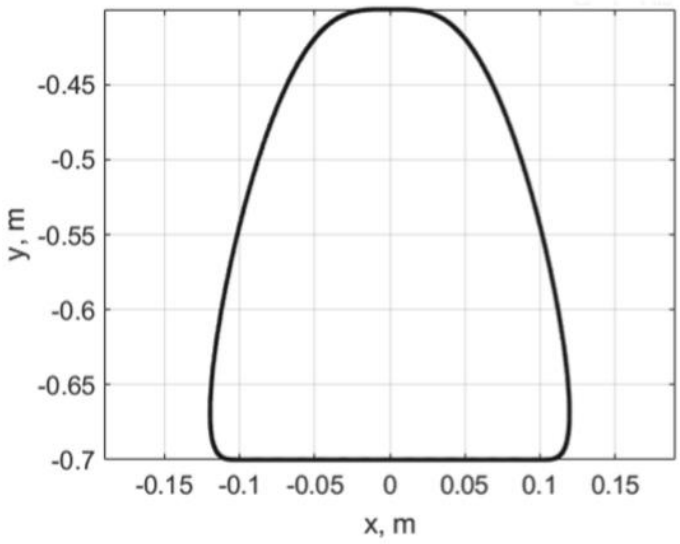
$\mathbf{x_3}$, $\mathbf{y_3}$ είναι οι τρέχων συντεταγμένες

$\boldsymbol{\gamma}$ είναι η γωνία της εφαρμογής της δύναμης

$$\gamma = \arcsin\left(\frac{P_x}{\sqrt{P_x^2 + P_y^2}}\right). \tag{6}$$

Έτσι οι γωνίες $\boldsymbol{\chi}^*_3$, $\mathbf{y}^*_3$ μεταδίδονται στους ρυθμιστές κίνησης που οδηγούν την κίνηση των ποδιών του εξωσκελετού

Η ώθηση των ποδιών του χειριστή αναπαριστώνται ως τη τροχιά του σημείου $\mathbf{O_{h3}(x_{h3}, y_{h3})}$ (η θέση του αστραγάλου)



5 Η τροχιά του Oh3 κατά το βήμα του χειριστή. Το βήμα περιγράφεται από τις παραμέτρους: step width 0.2m, step height 0.3m

Για να περιγράψουμε την κίνηση των συνδέσμων υπο την δράση των ροπών, χρησιμοποιούμε την εξής διαφορική εξίσωση. $\mathbf{M}(q)\ddot{\mathbf{q}} + \mathbf{V}(q, \ddot{q}) + \mathbf{G}(q) = Q.$

Σε αυτή την εξίσωση, **M**(q) είναι ο πίνακας αδράνειας, **V**(q, q*) το διάνυσμα των γενικών δυνάμεων της αδράνειας, **G**(q) το διάνυσμα της βαρύτητας που επιδρά στου συνδέσμους.

$$\mathbf{M}(q) = \begin{bmatrix} \frac{m_1 l_1^2}{3} + m_2 l_1^2 & \frac{m_2}{2} l_1 l_2 \cos(\phi_1 - \phi_2) \\ \frac{m_2}{2} l_1 l_2 \cos(\phi_1 - \phi_2) & \frac{m_2 l_2^2}{3} \end{bmatrix}, \tag{7}$$

$$\mathbf{V}(q, \dot{q}) = \begin{bmatrix} \dot{\phi}_2^2 \frac{m_2}{2} l_1 l_2 \sin(\phi_1 - \phi_2) \\ \dot{\phi}_1^2 \frac{m_2}{2} l_1 l_2 \sin(\phi_1 - \phi_2) \end{bmatrix}, \ \mathbf{G}(q) = \begin{bmatrix} -\left(m_2 + \frac{m_1}{2}\right) g l_1 \cos\phi_1 \\ -m_2 g \frac{l_2}{2} \cos\phi_2 \end{bmatrix}. \tag{8}$$

Q – διάνυσμα των εξωτερικών παραγόντων που επιδρούν στους μηχανισμούς.

$$\mathbf{Q} = \begin{bmatrix} M_1 - PL_1 \sin(\gamma - \phi_1) \\ M_2 - PL_2 \sin(\gamma - \phi_2) \end{bmatrix}. \tag{9}$$

Με αυτές τις εξισώσεις θα μπορέσουμε να πειραματιστούμε και να διερευνήσουμε τη δυναμική της αλληλεπίδρασης Ανθρώπου – Μηχανής.

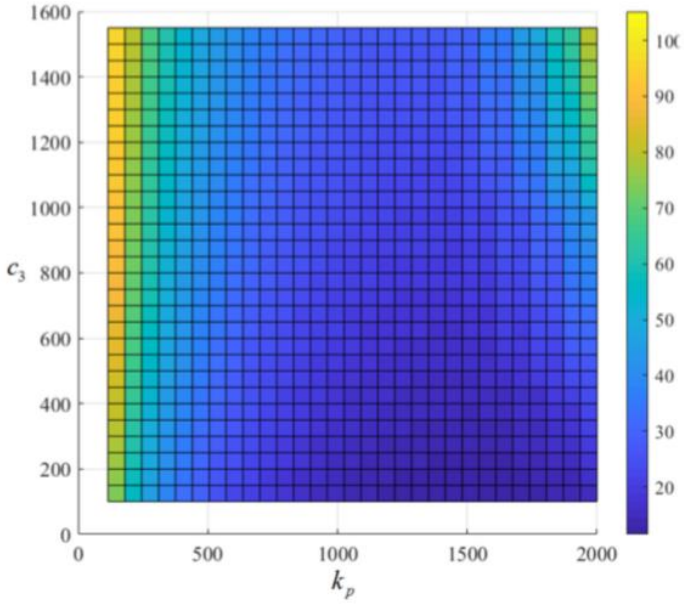
2.3 Πειράματα

Μέσω πειραμάτων που έγιναν ψηφιακά μέσω matlab οι ερευνητές προσπάθησαν να βρουν ποιες είναι οι κατάλληλες τιμές για την ακαμψία του εξωσκελετού ώστε να μην υπάρχει δυσφορία στον χρήστη όταν εκτελεί έργα με τον εξωσκελετό.

Πίνακας 1 Παράμετροι Μαθηματικού Μοντέλου

Parameter name	Designation	Value
First link length	L_1	0,44 m
Second link length	L_2	0,44 m
First link mass	m_1	8 kg
Second link mass	m_1	8 kg
Torque constant	k_m	13.9 Nm/A

Πίνακας 2 Γράφημα μεταξύ της ακαμψίας (**c3**) και του συντελεστή **kp**



Παρατηρούμε πως οι τιμές της δύναμης αυξάνονται στο σημείο που προσαρτάται ο χειριστής με τον εξωσκελετό το οποίο μπορεί να προκαλέσει δυσφορία. Όμως όσο πιο δύσκαμπτος είναι ο εξωσκελετός τόσο πιο εύστοχη είναι η κίνηση, για αυτό το αντισταθμίζουμε κατάλληλα μέσω του συντελεστή **kp**

2.4 Συμπεράσματα

Το άρθρο παρουσιάζει ένα μαθηματικό μοντέλο για την αλληλεπίδραση Ανθρώπου – Μηχανής ανάμεσα στα πόδια του χειριστή και του βιομηχανικού εξωσκελετού που διαθέτει ένα Copying Control System. Αυτό είναι συνεργατικής φύσης μεταξύ του χρήστη και του εξωσκελετού για την κατάλληλη βελτιστοποίηση των παραμέτρων στο σύστημα και την ζώνη στην οποία στέκεται το πόδι. Τα αποτελέσματα που έχουμε λάβει από τα πειράματα που κάναμε ως προς την ακαμψία μας δείχνουν πως χρειάζονται ελατήρια ανάρτησης χαμηλής ακαμψίας για να μπορέσουμε να είμαστε όσο πιο εύστοχοι γίνεται στην κίνηση του εξωσκελετού. Επειδή όμως είναι δύσκολη η εύρεση μαλακών ελαστικών επειδή ανάλογα τον χειριστή μπορεί να αλλάζει η ποσότητα ακαμψίας που μπορεί να χρειάζεται. Οπότε είναι σημαντικό να γίνεται σωστή παραμετροποίηση για κάθε χειριστή για την βέλτιστη χρήση του εξωσκελετού.

3 Έλεγχος ενεργής Εμπέδησης για τα Κάτω Άκρα ενός Εξωσκελετού

3.1 Εισαγωγή

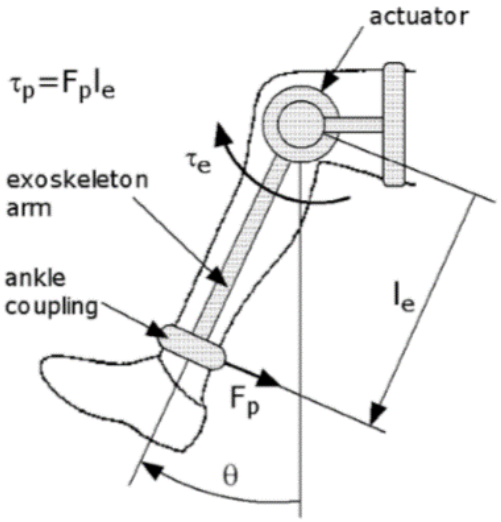
Τα τελευταία χρόνια έχουν υλοποιηθεί αρκετοί εξωσκελετοί για να βοηθήσουν ανθρώπους με δυσλειτουργίες στα άκρα, συμπληρώνοντας τη δύναμη που απαιτείτε για τις κινήσεις που κάνουν .Για τον έλεγχο της δύναμης αυτής συνήθως χρησιμοποιούνται τεχνικές για την εκτίμηση της κίνησης του χρήστη σε πραγματικό χρόνο. Οι εκτιμήσεις αυτές γίνονται με την μελέτη της ηλεκτρομυογραφικής δραστηριότητας όπου και πρόσφατες έρευνες έχουν φέρει αρκετή επιτυχία

Στο άρθρο παραθέτουμε μία νέα προσέγγιση στους μυοηλεκτρικούς εξωσκελετούς, δηλαδή εξωσκελετούς που μελετάνε την μυοηλεκτρική δραστηριότητα οι οποίοι όμως χρειάζονται συχνά παραμετροποίηση από χρήστη σε χρήστη.

Έτσι λοιπόν με τη χρήση τον έλεγχο της ενεργής εμπέδησης εξαρτάται λιγότερο από την εκτίμηση του χρήστη για την παραμετροποίηση και αποφεύγουμε τυχών συγκρούσεις που προκύπτουν από ανακριβείς εκτιμήσεις.

3.2 Έλεγχος Ενεργής Εμπέδησης

Η τρέχουσα έρευνα επικεντρώνεται στην ανάπτυξη μεθόδων ελέγχου για την υποστήριξη κινήσεων μεμονωμένων αρθρώσεων με σκοπό αργότερα την δημιουργία ενός εξωσκελετού με πολλαπλούς βαθμούς ελευθερίας.



6 Υποθετικός 1-DOF σκελετός

Ο έλεγχος της εμπέδησης θα χρησιμοποιείτε για να ρυθμίσει την αλληλεπίδραση μεταξύ του εξωσκελετού και του χρήστη. Ο εξωσκελετός θα έχει μια ψηφιακή παράμετρο εμπέδησης ανάλογα την προτίμηση μας, η οποία παράμετρος θα αποτελείτε από τη στιγμή αδράνειας, την απόσβεση και την ακαμπτότητα

Οπότε η επιθυμητή εμπέδηση είναι: $$Z_e^d(s) = I_e^d s + b_e^d + \frac{k_e^d}{s} \qquad (1)$$

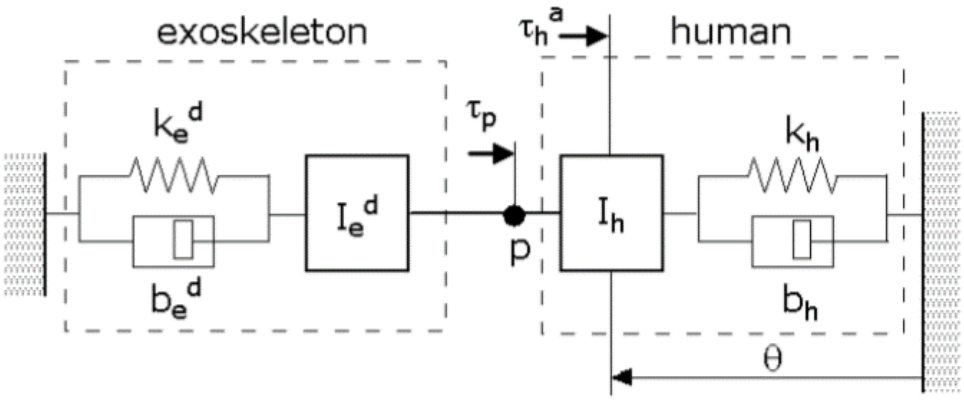
Στην εικόνα 7 βλέπουμε ένα συζυγής σύστημα που σχηματίζεται από την εικονική αντίσταση (εμπέδηση) του εξωσκελετού και του ανθρώπινου ποδιού

k = stiffness

b = damping

I = inertia moment

tp = συνολική ροπή που προκαλεί ο εξωσκελετός στο ανθρώπινο πόδι



7 Γραμμικό συζυγές σύστημα εξωσκελετού 1-DOF

Οπότε το σύστημα εκφράζεται από τη σχέση

$$\left[\left(I_h + I_e^d\right) s + b_h + b_e^d + \left(k_h + k_e^d\right) \frac{1}{s}\right] \omega = \tau_h^a$$

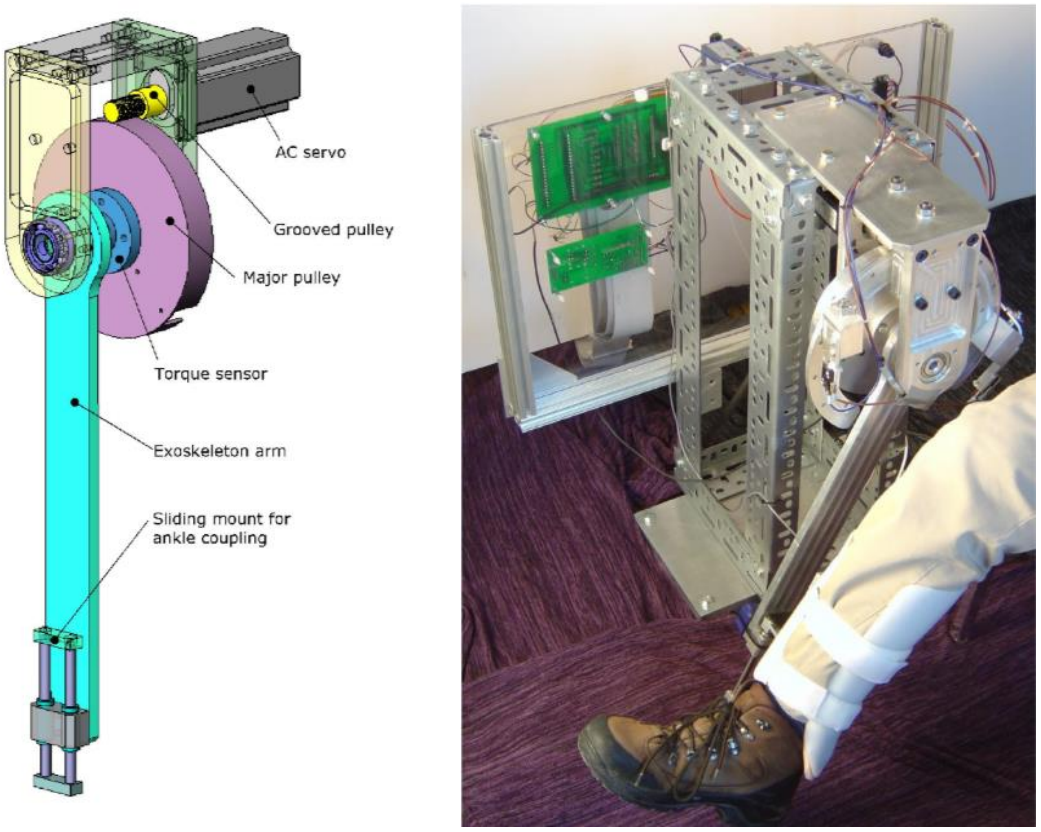
Εξίσωση a

Από αυτό βλέπουμε πως έχουμε ελευθερία στην δοκιμή τιμών για την γωνιακή ταχύτητα **ω** και την συνολική ροπή των μυών $\boldsymbol{\tau_h^a}$ διαλέγοντας κατάλληλες τιμές στις παραμέτρους ψηφιακής εμπέδησης $\mathbf{I_e^d}$ $\mathbf{b_d^e}$ και $\mathbf{k_d^e}$.

3.3 Υλοποίηση

3.3.1 1-DOF Εξωσκελετός

Στην εικόνα 8 δείχνουμε τι περιέχει ο εξωσκελετός για ένα πόδι, ένα servo κινητήρα, μια κυρτή τροχαλία, μια μεγάλη τροχαλία , ένα αισθητήρα ροπής, το άκρο του εξωσκελετού και μια συρόμενη βάση για την σύνδεση του στον αστράγαλο

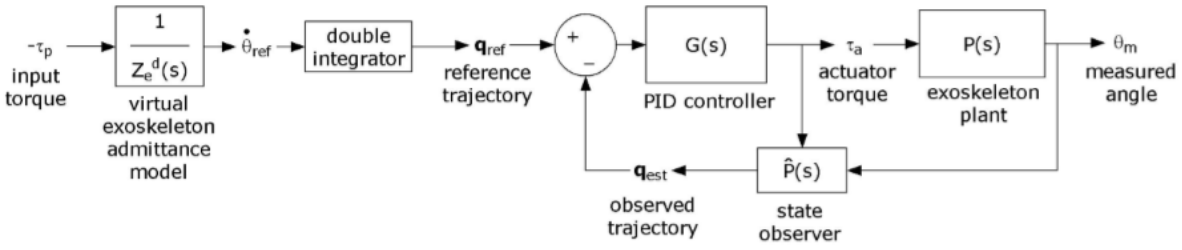


8 Διάγραμμα δομής 1-DOF εξωσκελετού και το τελικό αποτέλεσμα

3.3.2 Υλοποίηση Ελεγκτή Εμπέδησης

Ροπή που ασκήθηκε από τον σκελετό στον άνθρωπο χρησιμοποείτε σαν input σε έναν εικονικό ελεγκτή που περιέχει το αντίστροφο της επιθυμητής εμπέδησης του χρήστη.

Ο ελεγκτής μετά παράγει μία γωνιακή ταχύτητα αναφοράς $\boldsymbol{\theta}_{\mathbf{ref}}$ και μία τροχιά αναφοράς $\mathbf{q}_{\mathbf{ref}}$ οι οποίες ελέχονται συνεχώς με τις τιμές εκτίμησης της πραγματικής τροχιάς χρήστη.



3.3.3 Αντισταθμιστής Βάρους

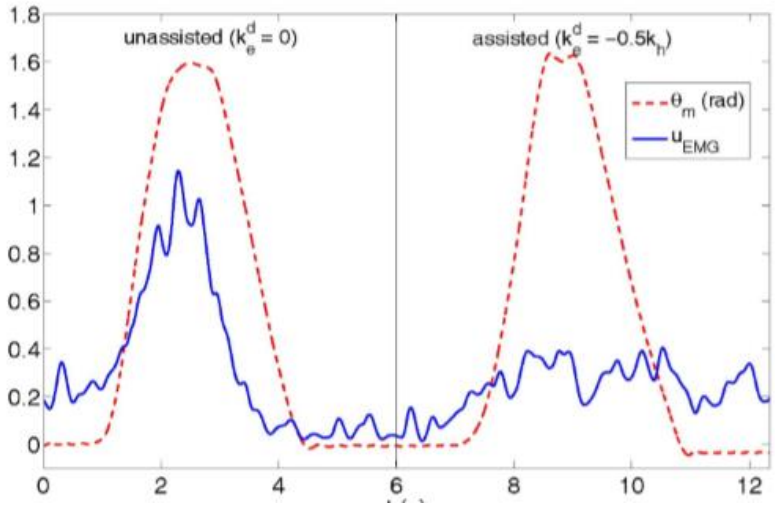
Παρόλο που αυτό το άρθρο δεν επικεντρώνεται στην αντιστάθμιση του βάρους, βοηθάει να έχουμε μια απλή εκτίμηση της αλληλεπίδρασης του βάρους στον εξωσκελετό όταν το φοράει ο χειριστής.

Όπως έχουμε δει στην εικόνα 7 και στην εξίσωση a ο όρος $\mathbf{k_h}$ χρησιμοποιείται για τον γραμμικό μετασχηματισμό της επίδρασης του βάρους. Οπότε για την αντιστάθμιση αρκεί να δώσουμε αρνητικές τιμές στην ακαμπτότητα του ψηφιακού εξωσκελετού $\mathbf{k_d^e}$ έτσι ώστε $\mathbf{k^d_h < k_h}$.

Στην εικόνα 9 βλέπουμε την ηλεκτρομυογραφική δραστηριότητα σε σχέση με την κίνηση του ποδιού σε δύο περιπτώσεις.

Στην πρώτη περίπτωση ο εξωσκελετός δεν βοηθούσε δηλαδή είχε k (παράμετρος ακαμψίας) 0 ενώ στη δεύτερη περίπτωση δώσαμε παράμετρο -0.5k δηλαδή ο εξωσκελετός μείωνε το βάρος του ποδιού κατά 50%.

Παρατηρούμε λοιπόν πως η ηλεκτρομυογραφική δραστηριότητα πέφτει και έτσι λοιπόν γίνεται πιο εύκολη η κίνηση.



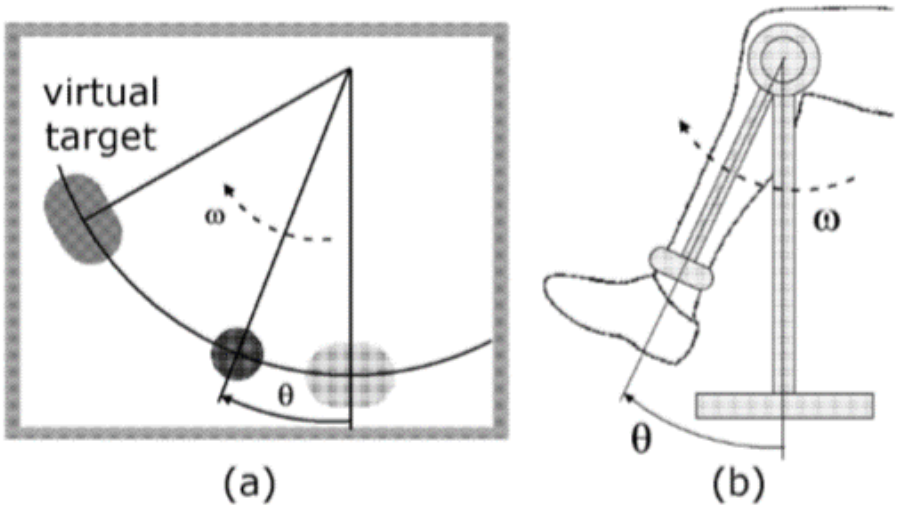
9 Γραφήματα αντιστάθμισης βάρους βάση της ηλεκτρομυογραφικής δραστηριότητας

3.4 Πειράματα με αρνητική απόσβεση

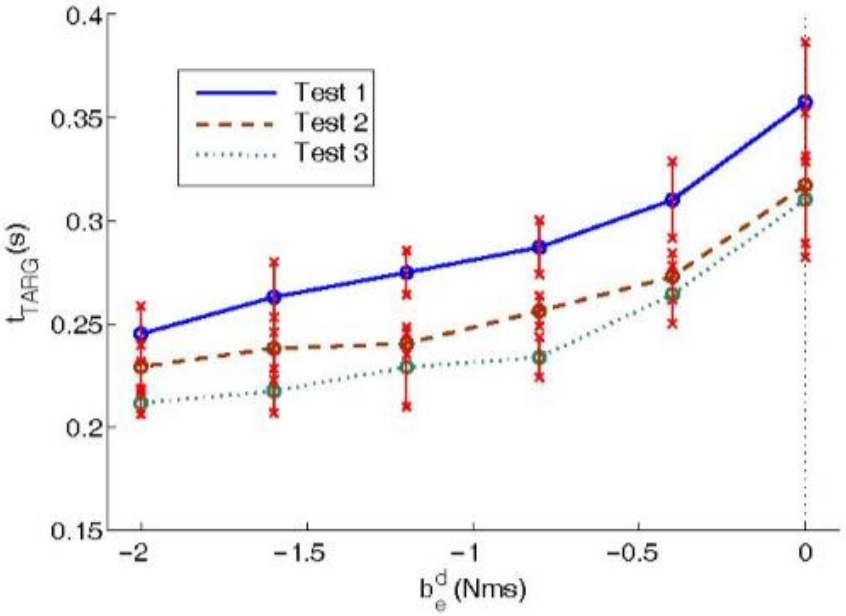
Στο παρακάτω πείραμα, ο χρήστης κουνάει το πόδι του καθώς φοράει τον εξωσκελετό από την εικόνα 8. Η γωνία του εξωσκελετού χρησιμοποιείται για την κίνηση ενός ψηφιακού αντικειμένου όπως φαίνεται στην εικόνα 10.

Το αντικείμενο προσομοιώνει την τροχιά του ποδιού έτσι ώστε το κέντρο της τροχιάς του να είναι το κέντρο της περιστροφής του γόνατου.

Για τα πειράματα έχουν χρησιμοποιηθεί διαφορετικές τιμές αρνητικής απόσβεσης και ο χρήστης πρέπει να κουνάει το αντικείμενο όσο πιο γρήγορα γίνεται και να το κρατάει στη τελική θέση για ένα συγκεκριμένο χρονικό διάστημα (κλάσματα δευτερολέπτου).



10 Η μπάλα (το αντικείμενο) φτάνει στον στόχο (a) καθώς ο χειριστής κουνάει το πόδι του (b)



11 Γράφημα μέσο χρόνου - αρνητικής απόσβεσης για τον χειριστή να φτάσει στο στόχο

Στην εικόνα 11 παρατηρούμε πως η αρνητική απόσβεση βοηθάει στη πιο γρήγορη κίνηση καθώς το αντικείμενο φτάνει στο στόχο του πιο γρήγορα με λιγότερη απόσβεση.

3.5 Συμπεράσματα

Καταλήξαμε στο πρώτο στάδιο για την ανάπτυξη μιας νέας μεθόδου για εξωσκελετούς επαύξησης των κάτω άκρων. Η προτεινόμενη μέθοδος ακολουθεί κυρίως δύο αρχές

1. Να ελέγχουμε την δυναμικότητα του εξωσκελετού και να απεικονίζουμε την ενεργή εμπέδηση.
2. Να χρησιμοποιούμε τις βοηθητικές δυνάμεις του εξωσκελετού για να βελτιώσουμε την κινηματική απόκριση των άκρων του χρήστη.

Το μεγαλύτερο μέρος του έργου επικεντρώθηκε στην δημιουργία ενός 1-DOF εξωσκελετού για να ελεγχθεί η συνεισφορά της ενεργής εμπέδησης. Η υλοποίηση του επικυρώθηκε με τις δοκιμές που πραγματοποιήθηκαν για την αντιστάθμιση βάρους των κινήσεων του ποδιού.

Στη συνέχεια πραγματοποιήθηκαν τα πειράματα για να μελετήσουμε την συνεισφορά της αρνητικής απόσβεσης στην ευκολότερη κίνηση του χρήστη. Παρατηρήσαμε από αυτά ότι όσο λιγότερη είναι η απόσβεση τόσο πιο γρήγορα φτάνει το πόδι που φοράει τον εξωσκελετό στον ψηφιακό στόχο.

4 Μοντελοποίηση Συστήματος Άρσης Βάρους Ανθρώπου Μηχανής

4.1 Εισαγωγή

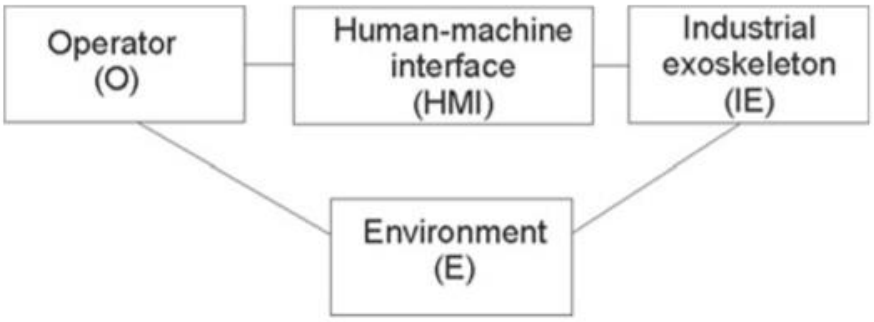
Το τρίτο άρθρο εισάγει την έννοια του HMS (Human-Machine System) δηλαδή την αρμονική συνύπαρξη του χρήστη και του εξωσκελετού μέσω ενός HMI (Human-Machine Interface).

Το άρθρο είναι αφιερωμένο στην ανάπτυξη αυτού του HMS κινηματικού μοντέλου και στην εκτίμηση της εξάρτησης του σε έναν αντισταθμιστή γραμμικού βάρους (**LGC**).

4.2 Biotechnical Walking System

- Ένα σύστημα ανθρώπου-μηχανής (**HMS**) αποτελείτε από
  - Τον χειριστή
  - Τον εξωσκελετό
  - Τη διεπαφή ανθρώπου-μηχανής (**HMI**)

Ένα τετοιο σύστημα ονομάζεται Biotechnical Walking System (**BTWS**)



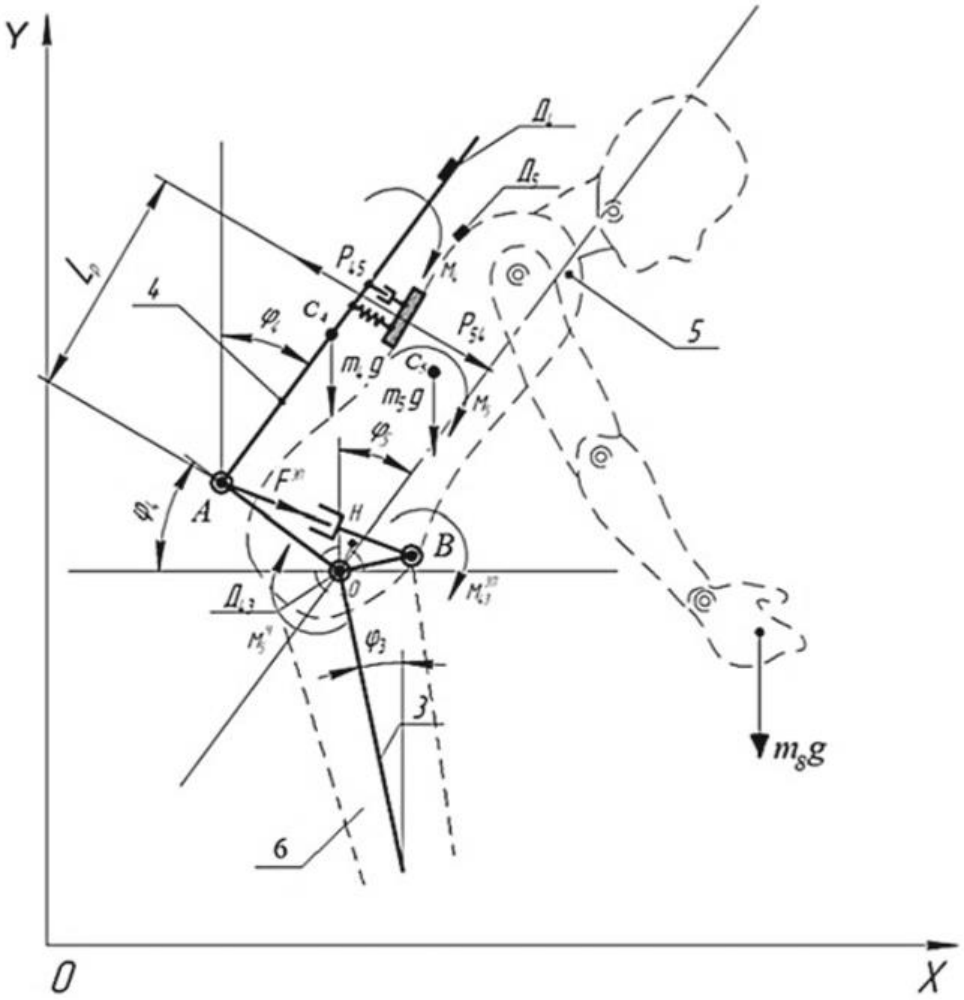
12 Δομικό σχήμα BTWS

Το κριτήριο για την αποτελεσματικότητα της αλληλεπίδρασης μεταξύ του χειριστή και του εξωσκελετού είναι η ακρίβεια και η ταχύτητα με την οποία ο εξωσκελετός αναπαράγει και βοηθάει την κίνηση του χειριστή. Το μέγιστο του κριτηρίου αυτού μπορεί να γίνει μέσω της διεπαφής ανθρώπου-μηχανής HMI. Το HMI επιτρέπει την χρήση αισθητήρων και επεξεργασίας σημάτων ώστε να αξιολογεί στιγμιαία την θέση των ποδιών και να χρησιμοποιήσει την πληροφορία αυτή για να δώσει κατάλληλο έναυσμα στον κινητήρα του εξωσκελετού ώστε να κινηθεί αντίστοιχα. Για να μπορέσει να αντισταθμίσει τις εξωτερικές δυνάμεις όπως την βαρύτητα, εξαρτάται και από τον LGC (γραμμικό αντισταθμιστή βάρους) που θα μελετήσουμε παρακάτω.

4.3 Μαθηματικό Μοντέλο του BTWS

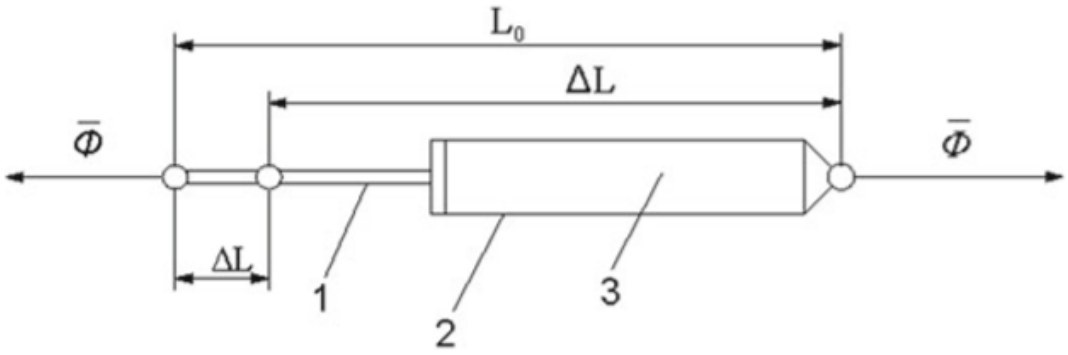
Στο παρών άρθρο, το BTWS παρέχει βοηθητική κίνηση στη μέση και πόδια του χειριστή. Αυτό επιτυγχάνεται με την εγκατάσταση ενός ενεργού Hip Joint (HJ, άρθρωση στο ισχίο) σε συνδυασμό με τον αντισταθμιστή βάρους. Το σύστημα μας επιτρέπει να εκφορτώσουμε το μυϊκό σύστημα της οσφυϊκής μοίρας της σπονδυλικής στήλης, ενώ το HMI μας επιτρέπει να ελέγξουμε τις παραμέτρους του συστήματος και αποτελείτε από απλούς αισθητήρες.

Το ανθρώπινο μοντέλο αποτελείτε από τον μηρό (3) , την πλάτη του εξωσκελετού (4) και την πλάτη του χειριστή (5). Η κίνηση του BTWS φαίνεται στην εικόνα 13.



13 Μοντέλο ανθρώπου-χειριστή εξωσκελετού

Υπάρχουν δύο είδη αντισταθμιστών: ενεργοί και παθητικοί, που ανάλογα αντισταθμίζουν ενεργά το drive του εξωσκελετού ή όχι. Για το LGC χρησιμοποιούμε το συνδυασμό τέτοιον αντισταθμιστών και αποτελείτε από (1) τη ράβδο, (2) το σώμα, (3) το ελαστικό στοιχείο και (4) έναν ηλεκτρικό κινητήρα (drive).



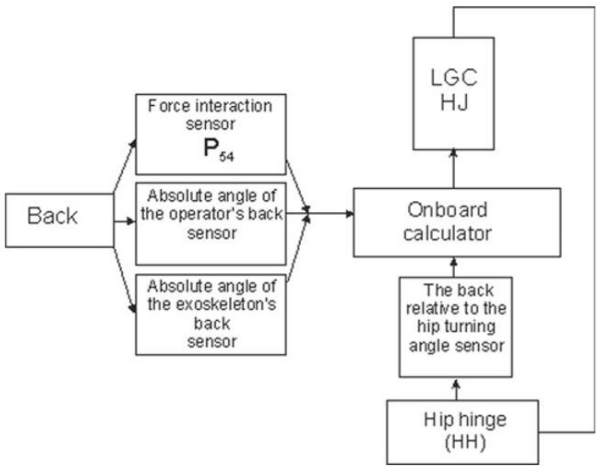
14 Σχήμα γραμμικού αντισταθμιστή βάρους.

Η διεπαφή ανθρώπου-μηχανής HMI αποτελείτε από τον μετρητή της κλίσης της πλάτης του χειριστή και του εξωσκελετού, ένα αισθητήρα που μετρά τη δύναμη της αλληλεπίδρασης, ένα hip joint εξοπλισμένο με έναν σχετικό μετρητή γωνίας, έναν ενσωματωμένο υπολογιστή και το LGC HJ που αναφέραμε παραπάνω. Αναλύοντας τα δεδομένα που παίρνουμε από τους αισθητήρες βάση του αλγορίθμου που έχουμε αναπτύξει (δεν αναφέρεται στο άρθρο ποιος είναι αλγόριθμος), ο ενσωματωμένος υπολογιστής παράγει κατάλληλες τάσεις που τις προμηθεύει στον κινητήρα του LGC.

Για την ανάπτυξη και περαιτέρω έρευνα του BTWS αναπτύχθηκε ένα δυναμικό και κινηματικό μοντέλο που μας επιτρέπει να επεξεργαστούμε λεπτομερώς τη διαδικασία αλληλεπίδρασης ενός βιολογικού αντικειμένου (άνθρωπο), τον ηλεκτρομηχανικό κομμάτι του εξωσκελετικού συστήματος και το HMI.

Για την μέτρηση της μηχανικής αλληλεπίδρασης των στοιχείων του συστήματος χρησιμοποιούμε την δύναμη $\mathbf{P_{54}}$ η οποία μας δίνεται από τους αισθητήρες.

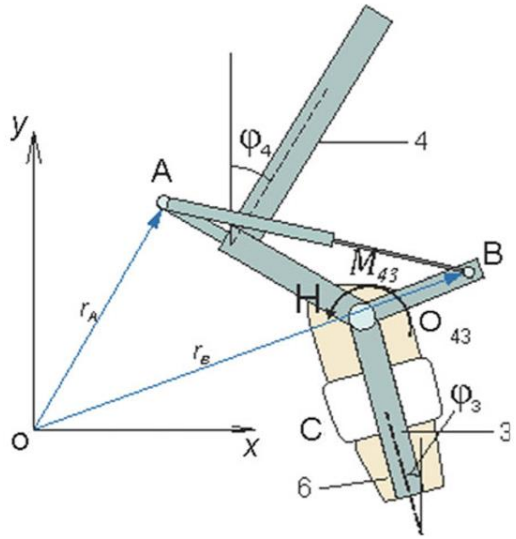
Η θέση στο οβελιαίο επίπεδο των συνδέσμων υπολογίζονται από τις σταθερές γωνίες $\boldsymbol{\varphi_4}$, $\boldsymbol{\varphi_5}$, $\boldsymbol{\varphi_3}$. Η σχετική γωνία $\boldsymbol{\varphi_{43}}$ ελέγχει την θέση της πλάτης του εξωσκελετού ανάλογα με τη θέση του ισχίου. Ο χρήστης θέτει την φύση της κίνησης του εξωσκελετού ρυθμίζοντας την μέγιστη κλίση της πλάτης του $\boldsymbol{\varphi_5}$.



15 Σχήμα διεπαφής Ανθρώπου-Μηχανής HMI

4.3.1 Κινηματικό Μοντέλο BTWS

Το μοντέλο έχει δύο γενικές συντεταγμένες φ4 και φ5. Η θέση της πλάτης του χειριστή αρχικοποιεί την γωνία φ5 ενώ η θέση της πλάτης του εξωσκελετού αρχικοποιεί την γωνία φ4. Είναι σημαντικό να ενώσουμε τις συντεταγμένες στα σημεία A και B που καθορίζουν την κινηματική του εξωσκελετού και την γενικευμένη συντεταγμένη φ4. Για τον σκοπό αυτό προτείνεται ένα κύκλωμα όπως στην παρακάτω εικόνα.



16 Σχετική διάταξη του LGC για τον χειριστή στο πόδι του εξωσκελετού (1. πόδι, 2. κάτω άκρο, 3. μηρός, 4. πίσω μέρος, 5. LGC)

Για να μπορέσουμε να καθορίσουμε τη σχέση μεταξύ του χαρακτηριστικού συστήματος που δίνει τις αντίστοιχες συντεταγμένες μπορούμε να περιγράψουμε τα διανύσματα σαν πίνακες με την ισότητα:

$$\bar{r}_B = \bar{r}_A + \bar{r}_{AB}.$$

Εξίσωση b

Από αυτή την ισότητα παίρνουμε την απόσταση μεταξύ των σημείων A και B που αντιστοιχούν στο μήκος L του LGC.

$$\bar{r}_{AB} = \bar{r}_B - \bar{r}_A$$

Εξίσωση c

Άρα από (b) και (c) έχουμε :

$$\bar{r}_{AB} = \bar{r}_{4A} - \bar{r}_{3B},$$

Εξίσωση d

Όπου

$$\bar{r}_{AB} = \begin{vmatrix} x_{AB} \\ y_{AB} \end{vmatrix}; \quad \bar{r}_{3B} = \begin{vmatrix} x_{O_{43B}} \\ y_{O_{43B}} \end{vmatrix}; \quad \bar{r}_{4A} = \begin{vmatrix} x_{O_{43A}} \\ y_{O_{43A}} \end{vmatrix}.$$

Σε συνέχεια, η μετατόπιση του LGC εξαρτάται μόνο από την θέση των διανυσμάτων $\mathbf{r_{3B}}$ και $\mathbf{r_{4A}}$. Το μήκος του LGC δίνεται από

$$L^2 = x_{AB}^2 + y_{AB}^2.$$

Χρησιμοποιώντας την εξίσωση b παίρνουμε:

$$x_{AB} = x_{O_{43B}} - x_{O_{43A}};$$
$$y_{AB} = y_{O_{43B}} - y_{O_{43A}};$$
$$x_{O_{43B}} = O_{43}B\cos\phi_3;$$
$$y_{O_{43A}} = O_{43}B\sin\phi_3, x_{O_{43A}} = O_{43}A\cos\phi_4;$$
$$y_{O_{43A}} = O_{43}A\sin\phi_4.$$

Άρα η σχέση μεταξύ του μήκους του LGC και της γωνίας περιστροφής εκφράζεται από τη σχέση:

$$L^2 = (O_{43}B\cos\phi_3 - O_{43}A\cos\phi_4)^2 + (O_{43}B\sin\phi_3 - O_{43}A\sin\phi_4)^2.$$

Για να βρούμε το ελάχιστο μήκος του L

$$\begin{aligned} \frac{dL}{d\phi_4} = \frac{1}{2}&((O_{43}B\cos\phi_3 - O_{43}A\cos\phi_4)^2 \\ &+ (O_{43}B\sin\phi_3 - O_{43}A\sin\phi_4)^2)^{-\frac{1}{2}} \\ &((O_{43}B\cos\phi_3 - O_{43}A\cos\phi_4)O_{43}A\sin\phi_4 \\ &+ (O_{43}B\sin\phi_3 - O_{43}A\sin\phi_4)O_{43}A\cos\phi_4) = 0 \end{aligned}$$

Λύνοντας την παραπάνω εξίσωση παίρνουμε την κατάλληλη τιμή της γωνίας φ4 τέτοια ώστε το L να είναι ελάχιστο.

Το LGC χαρακτηρίζεται από το γεγονός ότι η δύναμη που δημιουργείτε κατά την κίνηση καθορίζεται από την φόρμουλα $\mathbf{\Phi = C\ \Delta L}$ που είναι η δύναμη που παράγεται από την κίνηση των ελατηρίων στο LGC,

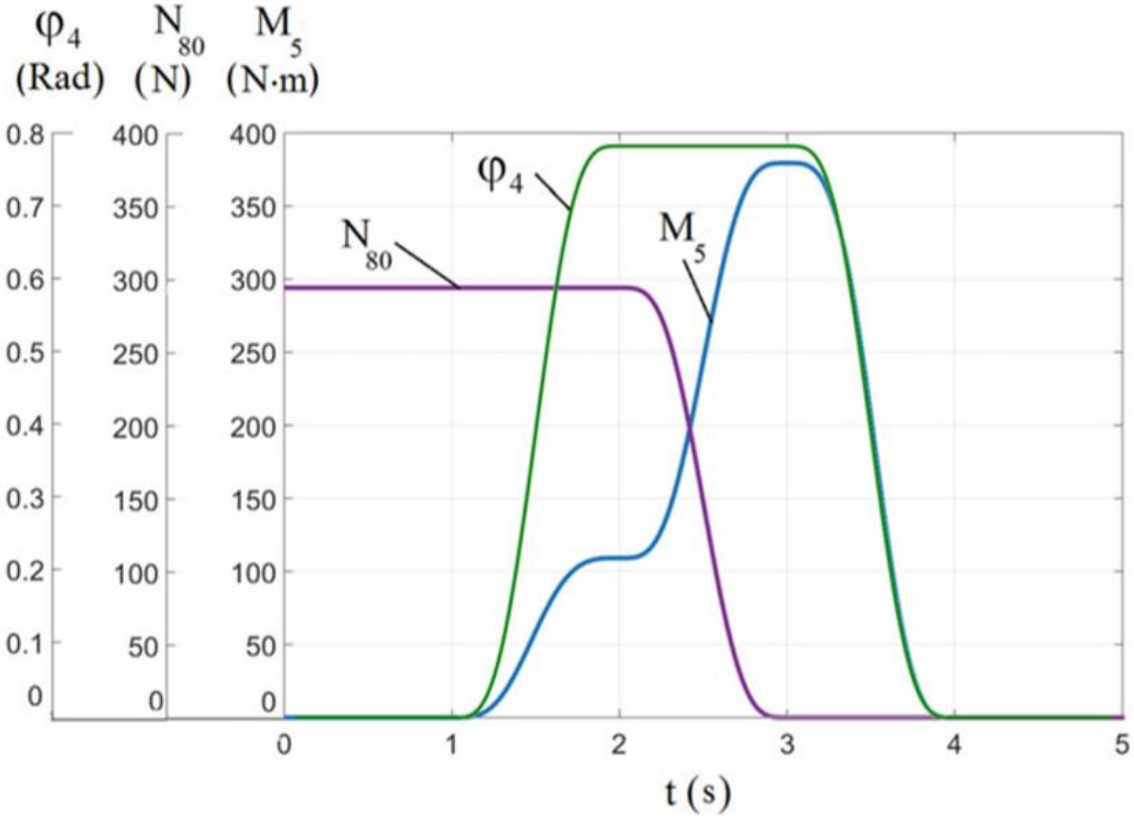
Όπου C – συντελεστής που περιγράφει τα χαρακτηριστικά των ελατηρίων. Η δύναμη που δημιουργείτε από το LGC σε σχέση με τον άξονα περιστροφής του HJ περιγράφεται από τη φόρμουλα

$$M_{O_{43}}(\Phi) = \Phi O_{43} H.$$

όπου $O_{43}H$ η δύναμη σε σχέση με το Φ στο κέντρο του HJ.

$$O_{43}H = \frac{2 \cdot \sqrt{(P \cdot (P - L_{AB}) \cdot (P - L_{O_{43}A}) \cdot (P - L_{O_{43}B}))}}{L_{AB}}. \quad (21.10)$$

$$P = \frac{1}{2} \cdot (L_{AB} + L_{O_{43}A} + L_{O_{43}B}). \quad (21.11)$$



17 Σχέση των παραμέτρων φ_4, N_{80} και M_5 αναλόγως του χρόνου

Η παραπάνω εικόνα περιγράφει την σχέση των παραμέτρων κατά τη διάρκεια που ο χειριστής σηκώνει κάποιο φορτίο.

4.4 Πειράματα

Το άρθρο δεν περιείχε κάποιο πείραμα.

4.5 Συμπεράσματα

Στο άρθρο προτείνονται δομικά σχέδια για το BTWS και το HMI. Αναπτύχθηκε το κινηματικό μοντέλο του BTWS. Καθορίζεται το μήκος του γραμμικού αντισταθμιστή βάρους που αλλάζει με βάση το μέγεθος της γωνίας περιστροφής του εξωσκελετού για διάφορες γεωμετρικές διαστάσεις. Αναλύθηκε η επιρροή του LGC κάθε φορά που υπάρχει κλίση κατά την κίνηση για την άρση κάποιο φορτίου. Προτάθηκε ένα μαθηματικό μοντέλο για τον υπολογισμό της δύναμης που δημιουργείτε από το LGC που περιέχει ενεργό και παθητικό αντισταθμιστή.

5 Φάση Υλοποίησης

5.1 Εισαγωγή

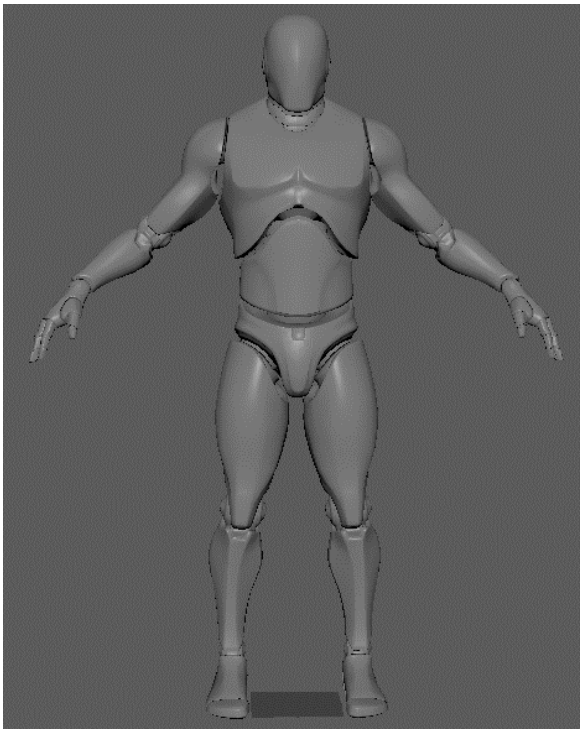
Σαν προσωπική υλοποίηση διάλεξα να δημιουργήσω ένα πόδι ενός βαθμού ελευθερίας που κινείτε με βάση αντίστροφης κινηματικής. Σαν σχηματικό μοντέλο διάλεξα το μοντέλο από το πρώτο άρθρο (εικόνα 2) και για την παρουσίαση έφτιαξα ένα ψηφιακό περιβάλλον που προσομοιώνει το περιβάλλον των πειραμάτων στο δεύτερο άρθρο όπως στην εικόνα 10.

Για την δημιουργία του ψηφιακού περιβάλλοντος χρησιμοποίησα τη μηχανή γραφικών Unity (Unity 2020.1.17f1 (64-bit)) και για την κίνηση με Inverse Kinematics έγραψα κώδικα σε C#.

Για την έρευνα μου στο πως να γράψω κίνηση με Inverse Kinematics συμβουλεύτηκα την αντίστοιχα διαφάνεια θεωρίας από το e-class του μαθήματος (6) και του άρθρου An Introduction to Procedural Animations (5)

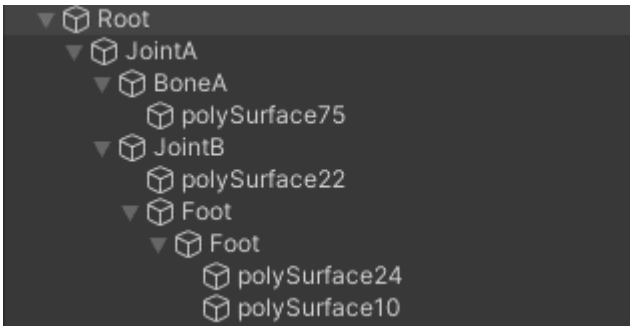
5.2 Υλοποίηση 3-DoF Ποδιού

Χρησιμοποιώντας ένα έτοιμο 3D μοντέλο που παρέχεται δωρεάν από την Unreal Engine και μέσω του προγράμματος Autodesk Maya (2018) τροποποίησα το μοντέλο έτσι ώστε να μπορώ να χρησιμοποιήσω ξεχωριστά τις αρθρώσεις στο ισχίο, μηρό και αστράγαλο.



18 Μοντέλο Mannequin

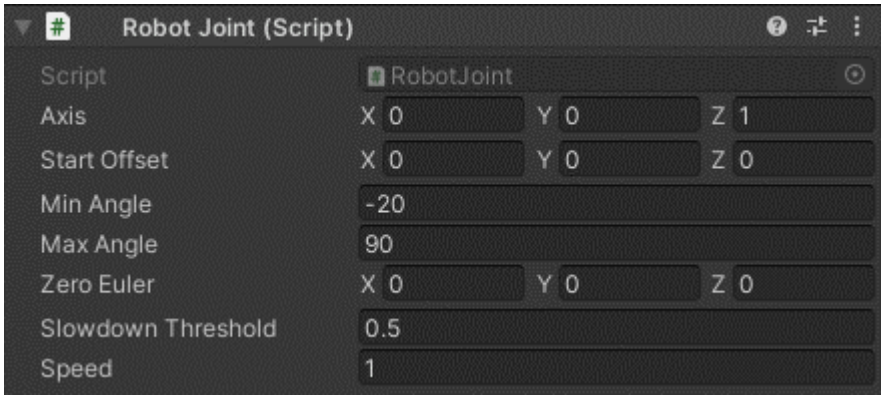
Έπειτα αφού έκανα κατάλληλα προσαρμογή το μοντέλο στην Unity έφτιαξα το μοντέλο με τέτοιο τρόπο ώστε να ακολουθεί κατάλληλη μορφή για τον κώδικα. Αυτό που φαίνεται παρακάτω στην



19 Μορφή στον Inspector της Unity

εικόνα 19 είναι η σχέση πατέρα παιδιού σε μια λίστα αντικειμένων της Unity που ονομάζεται Inspector. Τα Root, JointA, BoneA, JointB και Foot δεν είναι τίποτα παρά κενά σημεία στην ψηφιακή σκηνή μας. Αυτά τα κενά αντικείμενα όμως θα περιέχουν τα κατάλληλα scripts που θα εκτελούν την κίνηση. Αυτά θα είναι το **IKManager.cs** που θα βρίσκεται στο αντικείμενο Root και θα περιέχει τις βασικές μεθόδους για την πραγματοποίηση της κίνησης σύμφωνα με το μαθηματικό μοντέλο που θα μελετηθεί παρακάτω στην παράγραφο 5.3 . Τα Joint A, Joint B και ο πατέρας Foot περιέχουν το script **RobotJoint.cs** που είναι απλώς μια δομή που περιγράφει βασικά στοιχεία των αρθρώσεων. Τα υπόλοιπα παιδιά του Root είναι τα γραφικά που πήραμε από το Maya.

Θέτουμε λοιπόν στο JointA, JointB και Foot το script RobotJoint, το οποίο πλέον από την Unity θεωρείτε στοιχείο του κάθε αντικειμένου και βάζουμε ανάλογα τιμές. Στο JointA επειδή είναι το ισχίο



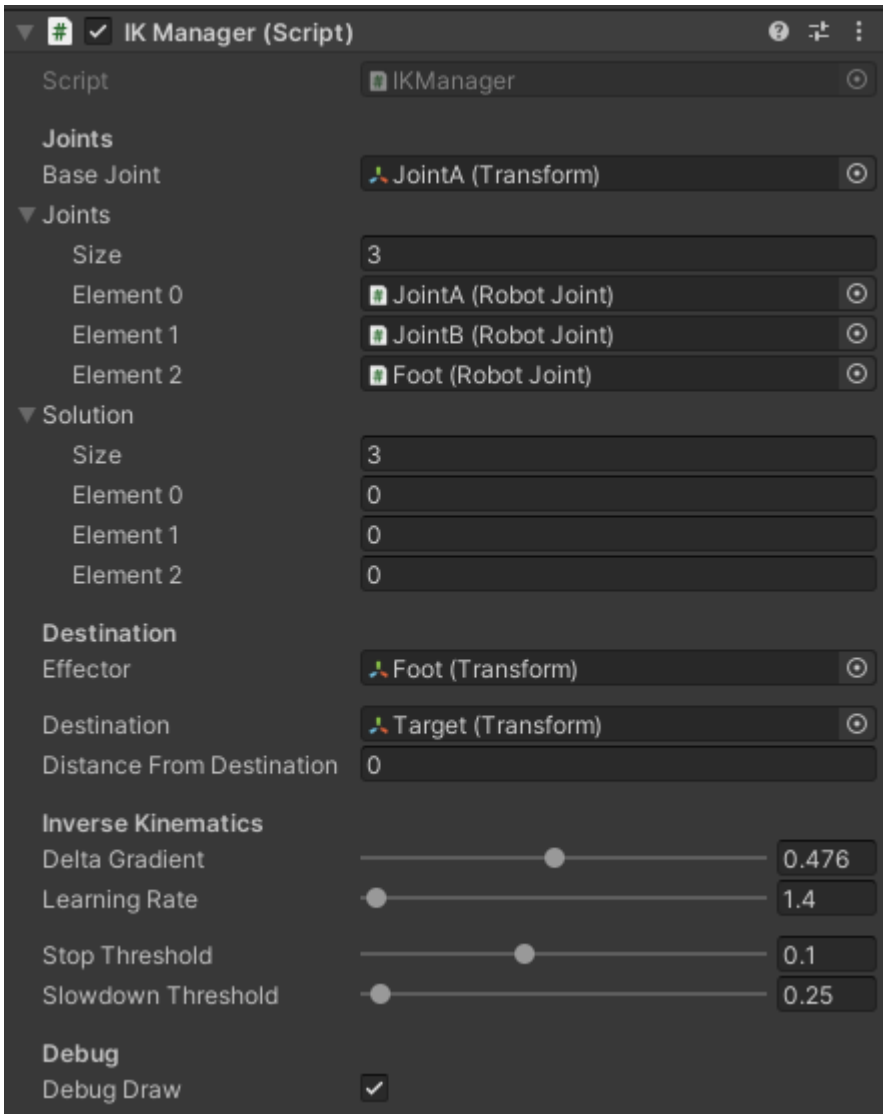
20 Παράδειγμα τιμών για το JointA

Είναι σημαντικό να επιτρέψουμε την περιστροφή του ποδιού στον άξονα Z [2]καθώς και θέτουμε την ελάχιστη και μέγιστη τιμή της περιστροφής αυτής ώστε να φαίνεται ρεαλιστικό το αποτέλεσμα της κίνησης του ανθρώπου.

[2] Η Unity έχει σύστημα αξόνων X, Y, Z όπου το X είναι δεξιά αριστερά το Y πάνω κάτω και το Z μπρος πίσω. Για περιστροφές χρησιμοποιούνται Quaternions.

Ομοίως θέτουμε τιμές για το JointB που είναι το γόνατο να περιστρέφεται στον Z άξονα με ελάχιστη και μέγιστη κλίση 4 και 90 αντίστοιχα. Στον αστράγαλο Foot βάλαμε RobotJoint αλλά δεν μας ενδιάφερε να δούμε κίνηση σε εκείνο το κομμάτι. (παρόλα αυτά είναι πιθανό αν μπουν κατάλληλες τιμές)

Τέλος στο Root βάζουμε το IKManager.cs όπου θα εκτελέσει τις βασικές λειτουργίες για την κίνηση.

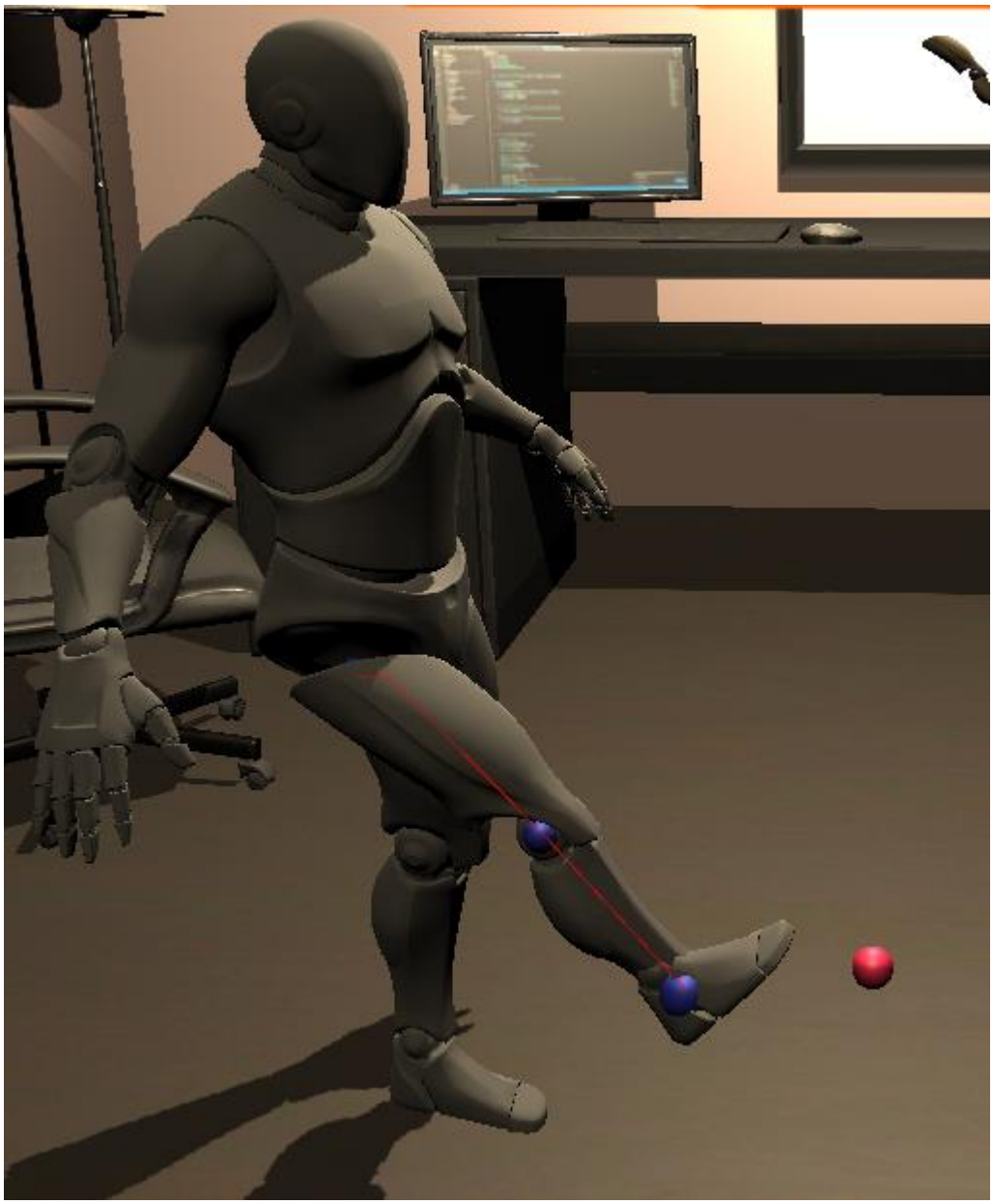


21 Τιμές για το IKManager.cs

Στο Base Joint βάζουμε το αντικείμενο που είναι η βάση της κίνησης. Θα μπορούσαμε να έχουμε ένα στερεό αντικείμενο όπως η λεκάνη του μοντέλου που δεν το έχουμε δηλώσει να κινηθεί αλλά δεν μας πειράζει να είναι το JointA. Στα Joints γεμίζουμε μια λίστα με τα αντικείμενα που περιέχουν το RobotJoints.cs σαν στοιχείο τους. Το Effector είναι το αντικείμενο το οποίο θα ακολουθήσει το target. Στο effector βάλαμε το συνονόματο παιδί του Foot. Οι υπόλοιπές τιμές είναι πειραματικές για

να πετύχουν την κίνηση που θέλουμε και θα τις περιγράψουμε λεπτομερώς στην επόμενη παράγραφο.

Σαν αποτέλεσμα λοιπόν έχουμε στην σκηνή την εξής εμφάνιση του μοντέλου.



Με κόκκινη γραμμή φαίνεται σύνδεση των Joints. Το target είναι η κόκκινη μπάλα. Ο λόγως που δεν φαίνεται σωστά η κίνηση είναι επειδή δεν τρέχει ακόμα η προσομοίωση, απλώς βλέπουμε την σκηνή.

5.3 Μαθηματικό Μοντέλο

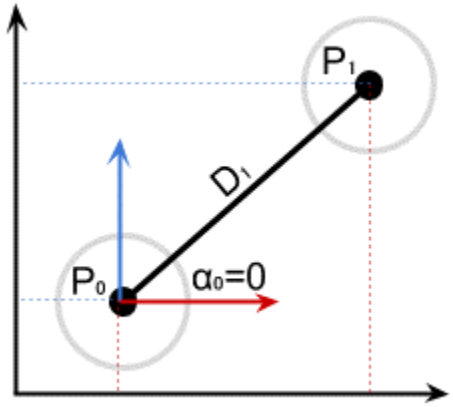
Σε αυτή την παράγραφο θα εξηγήσουμε τα βασικά μαθηματικά που ακολουθήσαμε για την δημιουργία του IKManager.

Αρχικά θα μιλήσουμε για τα Forward Kinematics. Για ευκολία θα τα παρουσιάσουμε σε 2D αλλά δεν αλλάζει κάτι δραματικά σε 3D.

Τα Forward Kinematics είναι μια λύση για την κίνηση ρομποτικών άκρων στον χώρο. Αυτό γίνεται με την περιστροφή του κάθε Joint/άρθρωσης. Όποτε χρειάζεται να κουνηθεί η κάθε άρθρωση, περιστρέφεται στον χώρο, μία κάθε φορά έτσι ώστε να φτάσει το τελικό σώμα (effector) στο σημείο που πρέπει.

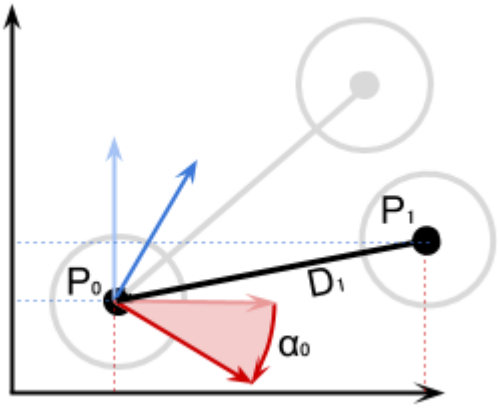
Οπότε είναι σημαντικό να μπορούμε να υπολογίσουμε την θέση των εμφωλευμένων αρθρώσεων λόγω της περιστροφής τους. Ας το λύσουμε για ευκολία για δύο αρθρώσεις.

Έστω ότι η πρώτη άρθρωση είναι στην αρχική μηδενική της θέση $\boldsymbol{\alpha_0} = 0$ αυτό σημαίνει πως:



P1 = P0 + D1

Αν το α_0 δεν είναι 0 τότε πρέπει απλώς να περιστρέψουμε το διάνυσμα της απόστασης κατά α_0 μοίρες.



Αυτό μπορούμε να το γράψουμε ως $P_1 = P_0 + \text{rotate}(D_1, P_0, \alpha_0)$.

Με αυτή τη λογική μπορούμε απλώς να γράψουμε και για τις επόμενες αρθρώσεις

$P_2 = P_1 + \text{rotate}(D_2, P_1, \alpha_0 + \alpha_1)$

Οπότε γενικά έχουμε:

$$P_i = P_{i-1} + rotate\left(D_i, P_{i-1}, \sum_{k=0}^{i-1} \alpha_k\right)$$

Αυτό στον κώδικα μας το μεταφράζουμε ως

```
    rotation *= Quaternion.AngleAxis(Solution[i - 1], Joints[i 1].Axis);
    Vector3 nextPoint = prevPoint + rotation * Joints[i].StartOffset;
```

Το AngleAxis(float angle, Vector3 axis) είναι μέθοδος της κλάσης Quaternion που προσφέρεται από την Unity. Δημιουργεί μια περιστροφή που περιστρέφει κατά γωνία angle στον άξονα axis. Το κομμάτι του κώδικα βρίσκεται σε μια for-loop για όλα τα joints που έχουμε θέσει ότι έχει το πόδι μας. Το Solution είναι ένα array που περιέχει τις γωνίες που θα πρέπει να κινηθεί το κάθε Joint. Το πως υπολογίζεται όμως θα το δούμε στη συνέχεια.

Τώρα λοιπόν έχοντας την μέθοδο για να βρούμε σε πιο σημείο βρισκόμαστε θα πρέπει να βρούμε πόσο απέχουμε από ένα σημείο που επιθυμούμε να φτάσουμε. Μια τέτοια συνάρτηση είναι απλή:

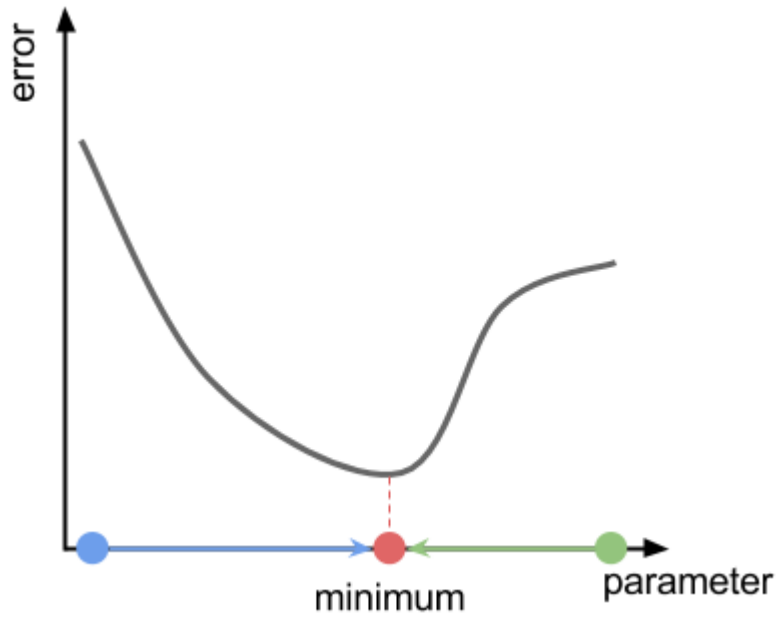
```
public float DistanceFromTarget(Vector3 target, float[] Solution) {
    Vector3 point = ForwardKinematics(Solution);
    return Vector3.Distance(target, point);
}
```

Όπου το Vector3.Distance(Vector3 pointA, Vector3 pointB) μας δίνει την απόσταση μεταξύ δύο σημείων και είναι μέθοδος που μας δίνεται από τη Unity της κλάσης Vector3.

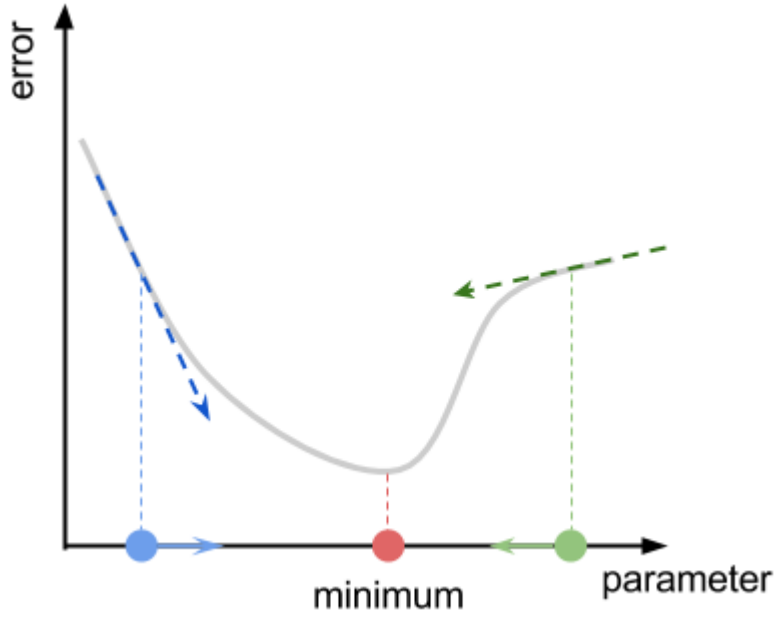
Για να λύσουμε όμως το πρόβλημα της αντίστροφης κινηματικής πρέπει να βρούμε πως θα ελαχιστοποιήσουμε την απόσταση που μας δίνεται από το DistanceFromTarget(). Η μέθοδος για την ελαχιστοποίηση που διαλέξαμε ονομάζεται Gradient Descent [3]. Παρόλο που δεν είναι η πιο αποτελεσματική μέθοδος, μας βοηθάει στο γεγονός ότι είναι problem-agnostic και αρκετά απλή.

Με το Gradient Descent έχουμε σκοπό να βρούμε το τοπικό ελάχιστο της απόστασης μεταξύ του effector και του στόχου. Η βασική ιδέα είναι ότι κάθε φορά ο αλγόριθμος θα πηγαίνει στην αντίθετη μεριά της κλίσης. Για να γίνει πιο κατανοητό ας δούμε το παρακάτω σχήμα

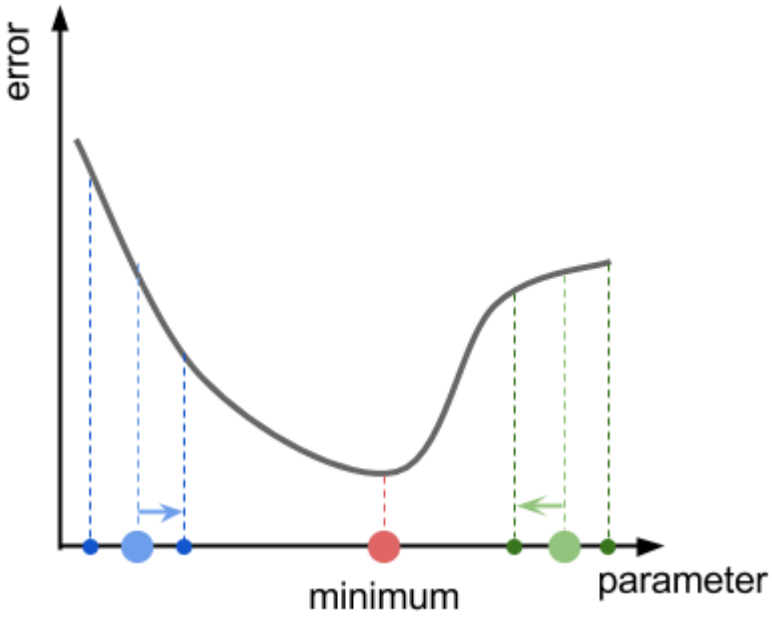
[3] https://en.wikipedia.org/wiki/Gradient_descent



Σε αυτή την περίπτωση έχουμε έναν αξονα Χ όπου είναι πιθανές τιμές της απόστασης και έναν άξονα Υ που είναι οι εσφαλμένες τιμές της συνάρτησης απόστασης. Σε κάθε σημείο το gradient descent θα πρέπει να βρεί την κλίση και να κουνηθεί προς την αντίθετη κατεύθυνση.



Για να βρούμε την κλίση αυτή όμως πρέπει να ξέρουμε την παράγωγο της συνάρτησης. Δυστυχώς δεν υπάρχει κάποιος τρόπος να βρούμε την πραγματική παράγωγο αλλά μπορούμε να βρούμε μια εκτίμηση της παραγώγου. Αυτό το πετυχαίνουμε κάνοντας δειγματοληψία στην συνάρτηση της απόστασης. Δειγματολιπτώντας τα κοντινά σημεία μπορούμε να ξέρουμε περίπου την κλίση της συνάρτησης. Αν το σφάλμα είναι μικρότερο στα αριστερά παμε αριστερά ή αν είναι στα δεξιά πάμε δεξιά.



Η απόσταση της δειγματοληψίας μεταξύ των σημείων είναι η **Δx**

Για να βρούμε μαθηματικά την κλίση της συνάρτησης όπως είπαμε πρέπει να βρούμε την παράγωγο της συνάρτησης. Αυτό μας επιτρέπει να μάθουμε και την κατεύθυνση. Αν η παράγωγος είναι θετική στο σημείο **p** τότε η συνάρτηση πάει προς τα πάνω, αν η παράγωγος είναι αρνητική πάει προς τα κάτω και αν είναι 0 τότε είναι ευθεία. Εμείς λοιπόν θα πάρουμε δείγμα της συνάρτησης σε δύο διαφορετικά σημεία για να πάρουμε την εκτίμηση της κλίσης.

$$\nabla f\left(p\right) = \frac{f\left(p + \Delta x\right) - f\left(p\right)}{\Delta x}$$

Μόλις έχουμε την παράγωγο μας λοιπόν πρέπει να κουνηθούμε στην αντίθετη κατεύθυνση της συνάρτησης. Άρα πρέπει κάθε φορά το σημείο μας p να το ανανεώνουμε ως:

$$p_{i+1} = p_i - L\nabla f\left(p_i\right)$$

Το **L** είναι μια σταθερά που λέγεται **Learning Rate** και εκφράζει πόσο γρήγορα κινούμαστε αντίθετα της κλίσης. Όσο πιο μεγάλη είναι η τιμή τόσο πιο γρήγορα θα φτάσουμε στο τοπικό ελάχιστο, αλλά και πιο πιθανό να ξεφύγει. Αυτά στον κώδικα μας φαίνονται από τις εξής γραμμές:

```
// Gradient descent
     float gradient = CalculateGradient(target, Solution, i, DeltaGradient);
     Solution[i] -= LearningRate * gradient * slowdown;
```

```
public float CalculateGradient(Vector3 target, float[] Solution, int i, float
delta) {
        // Saves the angle,
        // it will be restored later
        float solutionAngle = Solution[i];

        // Gradient : [F(x+h) - F(x)] / h
        // Update   : Solution -= LearningRate * Gradient
        float f_x = ErrorFunction(target, Solution);

        Solution[i] += delta;
        float f_x_plus_h = ErrorFunction(target, Solution);

        float gradient = (f_x_plus_h - f_x) / delta;

        // Restores
        Solution[i] = solutionAngle;

        return gradient;
    }
```

Αφού έχουμε πλέον το σημείο και την ελάχιστη απόσταση καλούμε την συνάρτηση InverseKinematics

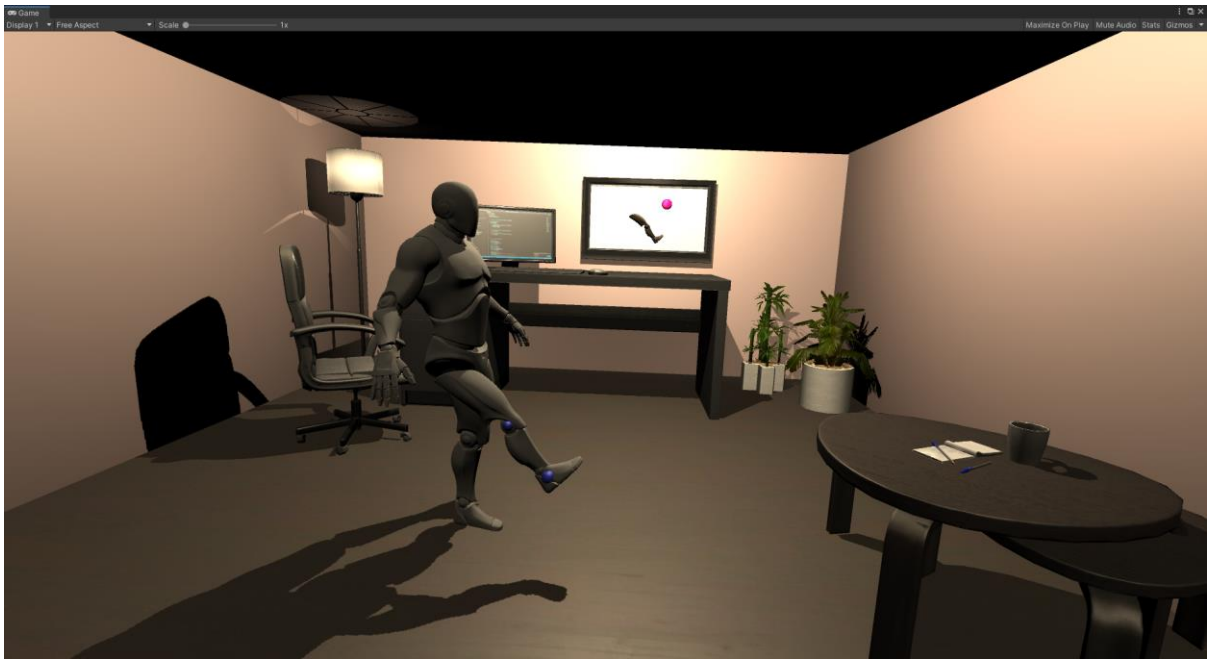
```
    public PositionRotation InverseKinematics(float[] Solution) {
        Vector3 prevPoint = Joints[0].transform.position;
        //Quaternion rotation = Quaternion.identity;

        // Takes object initial rotation into account
        Quaternion rotation = transform.rotation;
        for (int i = 1; i < Joints.Length; i++) {
            // Rotates around a new axis
            rotation *= Quaternion.AngleAxis(Solution[i - 1], Joints[i -
 1].Axis);
            Vector3 nextPoint = prevPoint + rotation * Joints[i].StartOffset;

            if (DebugDraw)
                Debug.DrawLine(prevPoint, nextPoint, Color.blue);

            prevPoint = nextPoint;
        }

        // The end of the effector
        return new PositionRotation(prevPoint, rotation);
    }
```

5.4 Δημιουργία Ψηφιακού Περιβάλλοντος

Αναφορά και credit σε assets που χρησιμοποιήθηκαν. Όλα τα assets κατεβήκανε από το Unity Asset Store (ψηφιακή αγορά assets της Unity) που ήταν δωρεάν και δεν είχαν κάποιο license που απέτρεπε την χρήση του για τον σκοπό της εργασίας.

Όνομα Asset	Credit
Mannequin(standard asset της Unreal Engine)	Epic Games
Wall TV set	3d.rina
Plants	Badri Bebua
Office Room Furniture	Elcanetay

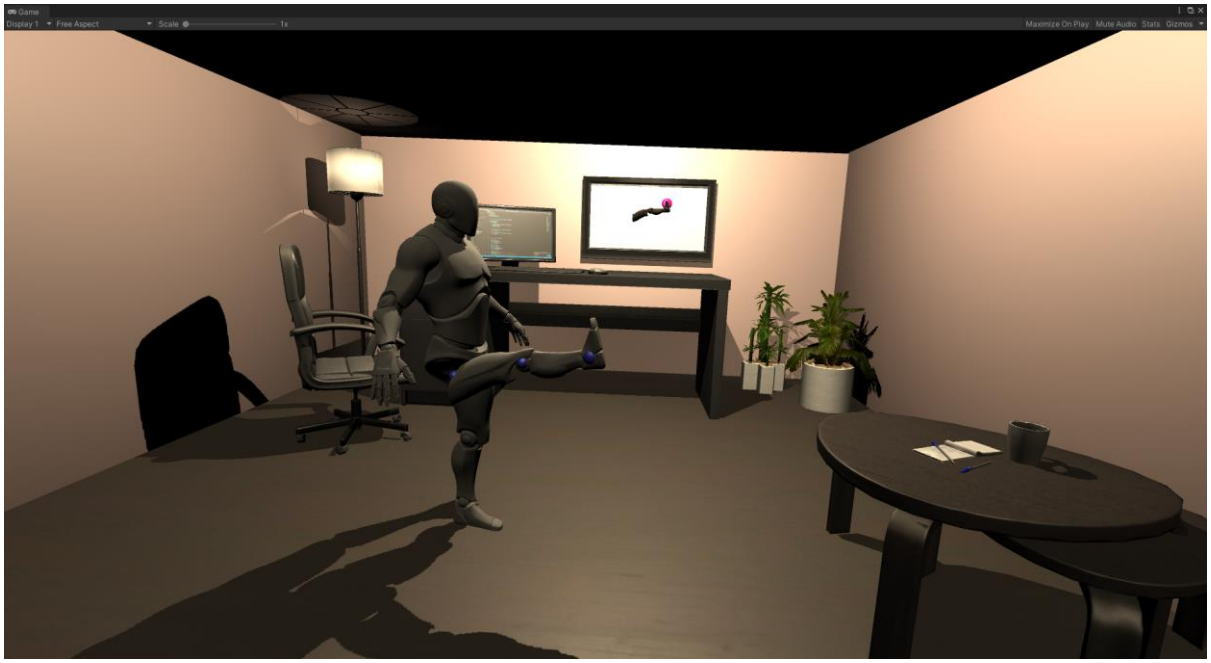
Έχοντας στη διάθεση μου αυτά τα assets τα τοποθέτησα κατάλληλα στην σκηνή, έβαλα κατάλληλο φωτισμό και έφτιαξα ένα επίπεδο το οποίο περιέχει αντικείμενο τύπου Render Texture το οποίο μου δίνει τη δυνατότητα να πάρω το view μιας ψηφιακής κάμερας και να το προβάλλω στο πεδίο αυτό. Λόγω αυτού στη σκηνή μου έχω δύο κάμερες, μία για την παρουσίαση και μία που βλέπει μόνο το πόδι και έναν ψηφιακό στόχο που βρίσκεται στο βάθος, τα οποία μετά τα κάνω render μέσω του επιπέδου αυτό στην τηλεόραση. Το τελικό αποτέλεσμα είναι ως εξής.



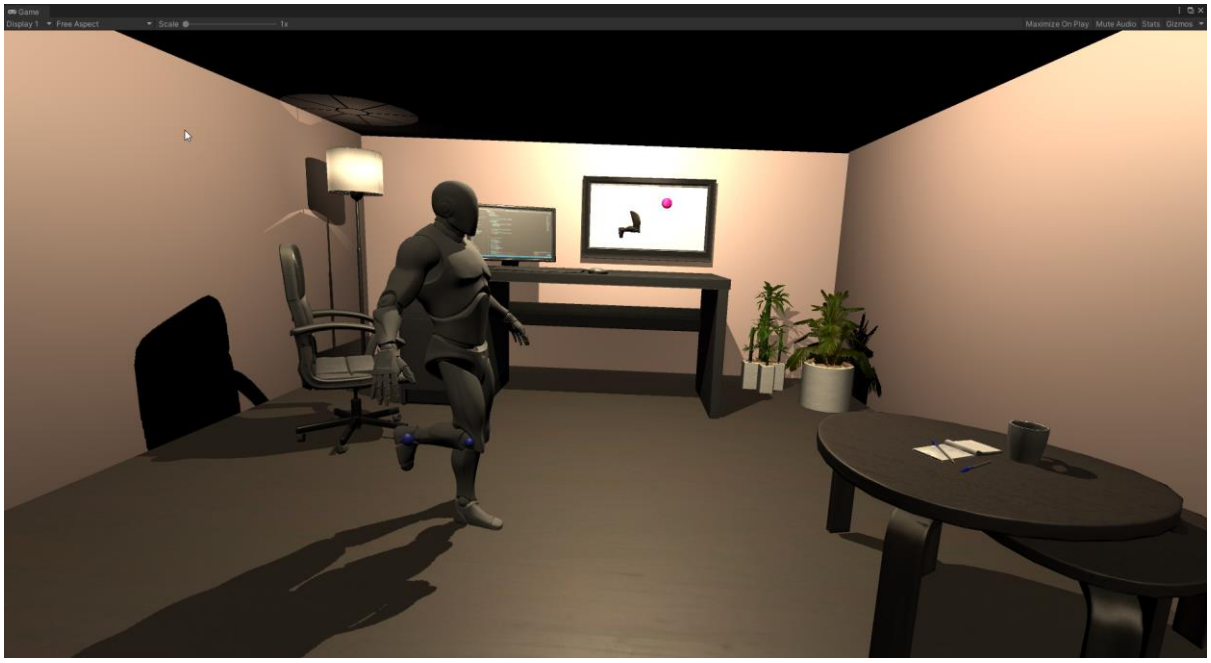
22 Τελική όψη της παρουσίασης

5.5 Τελική Παρουσίαση

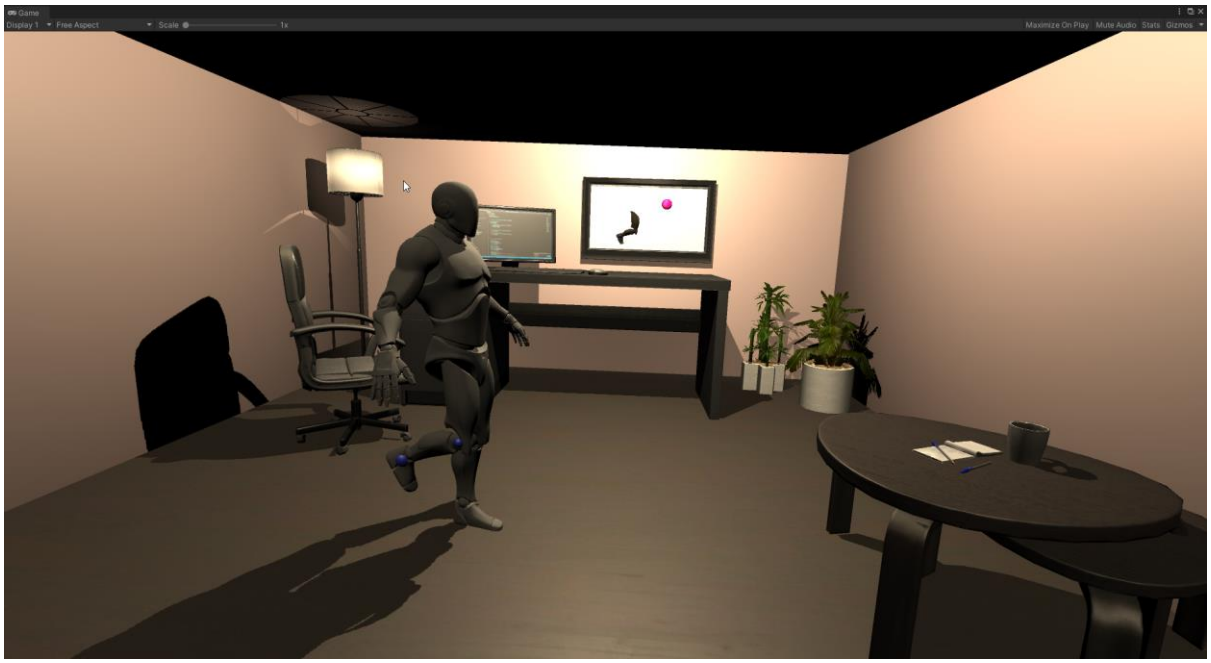
Στην τελική παρουσίαση θα δείξω σε εικόνες την κίνηση του ποδιού σε μέσες και ακραίες περιπτώσεις.



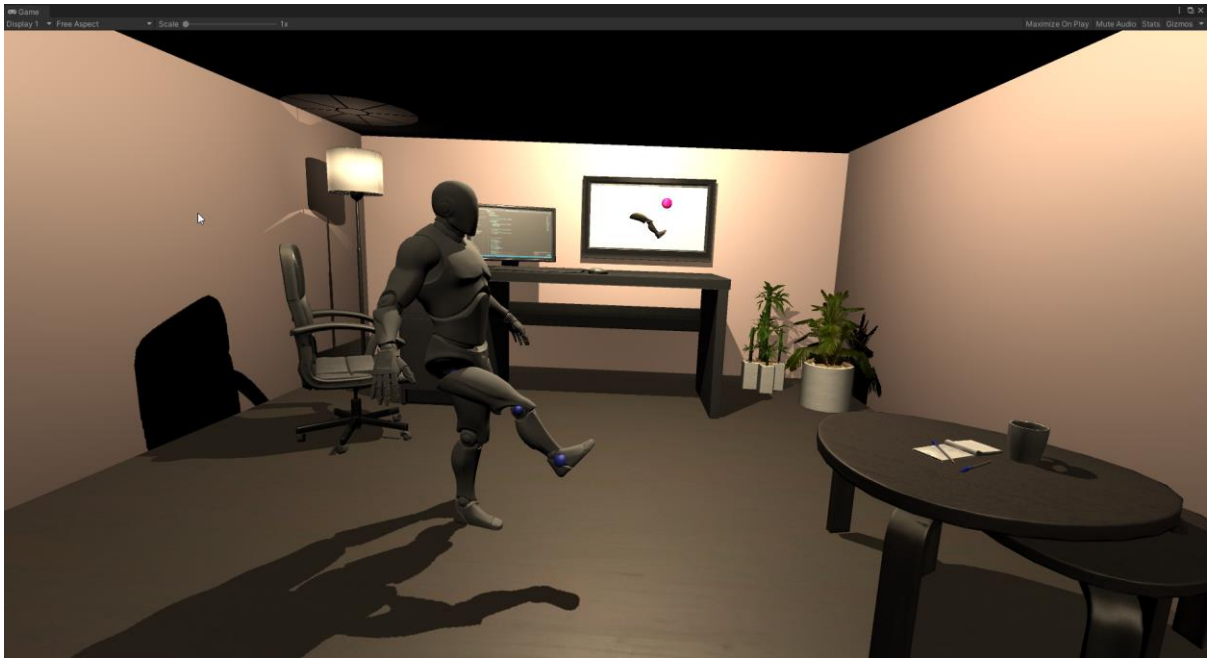
23 Όταν το target είναι κοντά στο ψεύτικο target ή πιο μακριά η μέγιστη κίνηση του ποδιού είναι όπως φαίνεται στη φωτογραφία.



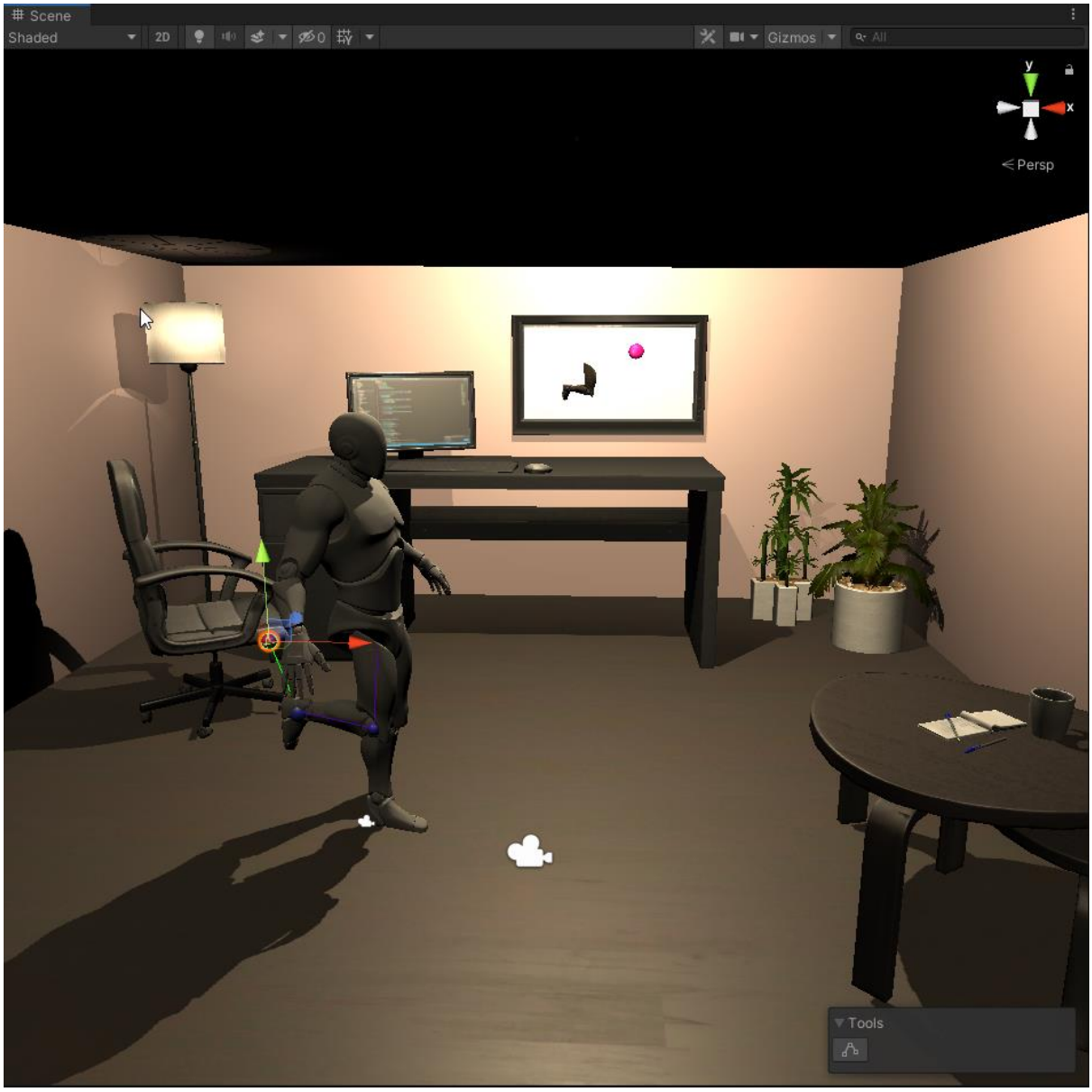
24 Το target είναι πίσω από το μπούτι. Ως target αναφέρομαι στην κόκκινη μπάλα, που δεν φαίνεται στη σκηνή.



25 Το target είναι πίσω πολύ μακριά από το πόδι.



26 Μια απλή περίπτωση κατά τη διάρκεια της κίνησης.



27 Εμφάνιση για το πως φαίνεται παράλληλα στη σκηνή του Editor της Unity. Εδώ φαίνεται το target και πια είναι η ελάχιστη απόσταση έγινε sampled από τον ο effector

6 Bibliography

1. **Ferris, Aaron J. Young and Daniel P.** *State-of-the-art and Future Directions for Lower Limb Robotic Exoskeletons.* 2016.

2. *Modeling of Human-Machine Interaction in an Industrial Exoskeleton Control System.* **Jatsun, .** Springer, Cham : s.n., 2020. International Conference on Interactive Collaborative Robotics.

3. *Modeling of the Exoskeletal Human-Machine System Movement Lifting a Load.* **Karlov, A., Saveleva, E., Yatsun, A., & Postolny, A.** Springer, Singapore : Zavalishin's Readings, 2020, September. Proceedings of 15th International Conference on Electromechanics and Robotics.

4. *Active-impedance control of a lower-limb assistive exoskeleton.* **Aguirre-Ollinger, G., Colgate, J. E., Peshkin, M. A., & Goswami, A.** s.l. : IEEE, 2007, June. IEEE 10th international conference on rehabilitation robotics.

5. **Zucconi, .** Introduction to Procedural Animations. *alanzucconi.com.* [Online] https://www.alanzucconi.com/2017/04/17/procedural-animations/.

6. **Καβαλλιεράτου, and Καραμπίδης, .** Inverse Kinematics. [Online] https://eclass.icsd.aegean.gr/modules/document/file.php/ICSD425/Lectures/9InverseKinematics.pdf.

7 Επισύναψη Κώδικα

7.1 RobotJoint.cs

```
using UnityEngine;

namespace RobotJoints {
    public class RobotJoint: MonoBehaviour {
        public Vector3 Axis;
        public Vector3 StartOffset;

        public float MinAngle;
        public float MaxAngle;

        // The initial one
        public Vector3 ZeroEuler;

        void Awake() {
            ZeroEuler = transform.localEulerAngles;
            StartOffset = transform.localPosition;
        }

        // Try to move the angle by delta.
        // Returns the new angle.
        public float ClampAngle(float angle, float delta = 0) {
            return Mathf.Clamp(angle + delta, MinAngle, MaxAngle);
        }

        // Get the current angle
        public float GetAngle() {
            float angle = 0;
            if (Axis.x == 1) angle = transform.localEulerAngles.x;
            else
            if (Axis.y == 1) angle = transform.localEulerAngles.y;
            else
            if (Axis.z == 1) angle = transform.localEulerAngles.z;

            return ClampAngle(angle);
        }
```

```
public float SetAngle(float angle) {
           angle = ClampAngle(angle);
      if (Axis.x == 1) transform.localEulerAngles = new Vector3(angle, 0, 0);
           else
      if (Axis.y == 1) transform.localEulerAngles = new Vector3(0, angle, 0);
           else
      if (Axis.z == 1) transform.localEulerAngles = new Vector3(0, 0, angle);

           return angle;
       }



       // Moves the angle to reach
       public float MoveArm(float angle) {
           return SetAngle(angle);
       }

       private void OnDrawGizmos() {
    Debug.DrawLine(transform.position, transform.parent.position, Color.red);
       }
   }


}
```

7.1.1 Επεξήγηση κώδικα

Namespace RobotJoints: για την χρήση της δομής με όνομα RobotJoints στο IKManager

Vector3: Κλάση της Unity που περιγράφει ένα διάνυσμα στον χώρο.

Axis: σε πιο άξονα περιστρεφόμαστε.

StartOffset: αρχικές τιμές της θέσης του Joint.

MinAngle, MaxAngle : ελάχιστες και μέγιστες τιμές γωνίας που μπορεί να περιστραφεί το κάθε Joint.

ZeroEuler: οι αρχικές γωνίες στις οποίες βρίσκονται τα Joints

Awake(): Μέθοδος της Unity. Καλείτε πρώτη όταν πατήσεις Play.

Σε αυτή αρχικοποιούμε κάθε φορά τις μεταβλητές ZeroEuler και StartOffset.

ClampAngle(float angle, float delta = 0)

Επιστρέφουμε την γωνία περιστροφής η οποία θα έχει εύρος τιμών μεταξύ min και max angle.

Το Clamp είναι συνάρτηση της κλάσης Mathf της Unity που επιστρέφει μεταβλητές σε εύρος τιμών που ορίζουμε.

GetAngle()

Σε αυτή τη συνάρτηση ανάλογα τι έχουμε θέσει το Axis παίρνουμε την ανάλογη γωνία.

SetAngle()

Σε αυτή τη συνάρτηση ανάλογα τι έχουμε θέσει το Axis θέτουμε την ανάλογη γωνία για να κουνήσουμε το άκρο.

MoveArm() απλώς καλεί την SetAngle().

OnDrawGizmos() μέθοδος της Unity, τη χρησιμοποιούμε για να εμφανίσουμε πληροφορίες στον Editor της Unity. Σε αυτή τη περίπτωση με το Debug.DrawLine() ζωγραφίζουμε μια κόκκινη γραμμή μεταξύ των Joints

7.2 IKManager.cs

```
using UnityEngine;
using RobotJoints;
using System.Collections;
using System.Collections.Generic;


public delegate float ErrorFunction(Vector3 target, float[] solution);

public struct PositionRotation {
    Vector3 position;
    Quaternion rotation;

    public PositionRotation(Vector3 position, Quaternion rotation) {
        this.position = position;
        this.rotation = rotation;
    }

    // PositionRotation to Vector3
    public static implicit operator Vector3(PositionRotation pr) {
        return pr.position;
    }
    // PositionRotation to Quaternion
    public static implicit operator Quaternion(PositionRotation pr) {
        return pr.rotation;
    }
}
```

```
public class IKManager: MonoBehaviour {

    [Header("Joints")]
    public Transform BaseJoint;
    public RobotJoint[] Joints = null;
    // The current angles
    public float[] Solution = null;

    [Header("Destination")]
    public Transform Effector;
    [Space]
    public Transform Destination;
    public float DistanceFromDestination;
    private Vector3 target;

    [Header("Inverse Kinematics")]
    [Range(0, 1f)]
    public float DeltaGradient = 0.1f; // Used to simulate gradient (degrees)
    [Range(0, 100f)]
    public float LearningRate = 0.1f; // How much we move depending on the
gradient

    [Space()]
    [Range(0, 0.25f)]
    public float StopThreshold = 0.1f; // If closer than this, it stops
    [Range(0, 10f)]
    public float SlowdownThreshold = 0.25f; // If closer than this,
      it linearly slows down


    public ErrorFunction ErrorFunction;

    [Header("Debug")]
    public bool DebugDraw = true;
  void Start() {
        GetJoints();
        // The error function to minimise
        ErrorFunction = DistanceFromTarget;
    }

    public void GetJoints() {
        Joints = BaseJoint.GetComponentsInChildren < RobotJoint > ();
        Solution = new float[Joints.Length];
    }
```

```
// Update is called once per frame
    void Update() {
        // Do we have to approach the target?
        Vector3 direction = (Destination.position -
transform.position).normalized;
        target = Destination.position - direction * DistanceFromDestination;
        //if (Vector3.Distance(Effector.position, target) > Threshold)
        if (ErrorFunction(target, Solution) > StopThreshold)
            ApprochTarget(target);

        if (DebugDraw) {
            Debug.DrawLine(Effector.transform.position, target, Color.green);
        }
    }


    public void ApprochTarget(Vector3 target) {
        // Starts from the end, up to the base
        // Starts from joints[end-2]
        for (int i = Joints.Length - 1; i >= 0; i--)
        //for (int i = 0; i < Joints.Length - 1 - 1; i++)
        {
            // FROM:    error:      [0, StopThreshold,  SlowdownThreshold]
            // TO:      slowdown:   [0, 0,              1                 ]
            float error = ErrorFunction(target, Solution);
            float slowdown = Mathf.Clamp01((error -
 StopThreshold) / (SlowdownThreshold - StopThreshold));

            // Gradient descent
            float gradient = CalculateGradient(target, Solution, i, DeltaGradi
ent);
            Solution[i] -= LearningRate * gradient * slowdown;
            // Clamp
            Solution[i] = Joints[i].ClampAngle(Solution[i]);

            // Early termination
            if (ErrorFunction(target, Solution) <= StopThreshold)
                break;
        }


        for (int i = 0; i < Joints.Length - 1; i++) {
            Joints[i].MoveArm(Solution[i]);
        }
    }
```

```
   /* Calculates the gradient for the inverse kinematic.
     * It simulates the forward kinematics the i-th joint,
     * by moving it +delta and -delta.
     * It then sees which one gets closer to the target.
     * It returns the gradient (suggested changes for the i-th joint)
     * to approach the target. In range (-1,+1)
     */
    public float CalculateGradient(Vector3 target, float[] Solution, int i, fl
oat delta) {
        // Saves the angle,
        // it will be restored later
        float solutionAngle = Solution[i];

        // Gradient : [F(x+h) - F(x)] / h
        // Update   : Solution -= LearningRate * Gradient
        float f_x = ErrorFunction(target, Solution);

        Solution[i] += delta;
        float f_x_plus_h = ErrorFunction(target, Solution);

        float gradient = (f_x_plus_h - f_x) / delta;

        // Restores
        Solution[i] = solutionAngle;

        return gradient;
    }

    // Returns the distance from the target, given a solution
    public float DistanceFromTarget(Vector3 target, float[] Solution) {
        Vector3 point = InverseKinematics(Solution);
        return Vector3.Distance(target, point);
    }









    public PositionRotation InverseKinematics(float[] Solution) {
```

```
        Vector3 prevPoint = Joints[0].transform.position;
        //Quaternion rotation = Quaternion.identity;

        // Takes object initial rotation into account
        Quaternion rotation = transform.rotation;
        for (int i = 1; i < Joints.Length; i++) {
            // Rotates around a new axis
       rotation *= Quaternion.AngleAxis(Solution[i - 1], Joints[i -1].Axis);
       Vector3 nextPoint = prevPoint + rotation * Joints[i].StartOffset;

            if (DebugDraw)
                Debug.DrawLine(prevPoint, nextPoint, Color.blue);

            prevPoint = nextPoint;
        }

        // The end of the effector
        return new PositionRotation(prevPoint, rotation);
    }
}
```

7.2.1 Επεξήγηση κώδικα

Using RobotJoints; Για να μπορέσουμε χρησιμοποιήσουμε τον τύπο RobotJoints που γράψαμε προηγουμένως.

Public delegate ErrorFunction(Vector3 Target, float[] Solution)

Η συνάρτηση σφάλματος μας όπως εξηγήσαμε στην παράγραφο 5.3

Public struct PositionRotation. Απλή δομή που περιέχει τη θέση και την γωνία του κάθε Joint. Στη συνέχεια γράφουμε τον απαραίτητο κώδικα για να μπορούμε να πάρουμε και να θέσουμε μόνο position ή μόνο rotation αν χρειαστεί.

BaseJoint : η βάση του ποδιού

RobotJoint[] Joints : πίνακας με τα RobotJoint

Solution[] οι τρέχων γωνίες

Effector: το αντικείμενο το οποίο προσπαθεί να φτάσει το Target.

Destination: το αντικείμενο Target.

DistanceFromDestination: αν θέλουμε να κάνουμε offset την απόσταση από το κέντρο του Target.

Target: το σημείο στο χώρο που θέλουμε να φτάσουμε, μας δίνεται από το Destination.

Delta Gradient: η απόσταση για την δειγματοληψία, το Δx από τη παράγραφο 5.3

Learning Rate: σταθερά που καθορίζει πόσο γρήγορα θα φτάσουμε στη λύση του gradient descent. Περιγράφεται στο 5.3

Stop Threshold: απλό κατώφλι, σε πόση απόσταση να σταματήσει ο effector.

Slowdown Threshold: ομοίως όσο φτάνει κοντά, χάνει ταχύτητα.

ErrorFunction: δημιουργία της συνάρτησης που δηλώσαμε παραπάνω.

DebugDraw: flag για το αν θέλουμε να φαίνονται τα Gizmos η όχι από τον κώδικα μας.

Void Start(): Συνάρτηση με προτεραιότητα αμέσως μετά την Awake(). Της Unity.

Σε αυτή αρχικοποιούμε τα Joints και δίνουμε στο Solution μήκος όσα Joints έχουμε.

Void Update(): Της Unity, βασικό βρόχος που εκτελείτε κάθε frame. Σαν το loop() του Arduino.

Εδώ βλέπουμε την κατεύθυνση μας από το Destination. Το target παίρνει τιμή ίση με τη θέση του destination – την κατεύθυνση μας (* DistanceFromDestination, αν έχουμε offset)

Αν δεν χρειάζεται να σταματήσουμε -> ApproachTarget()

Debug.DrawLine(): ζωγραφίζουμε μια γραμμή που μας δείχνει την απόσταση του effector και του Destination (η αλλιώς target, η τιμή θα είναι ίδια αν δεν έχουμε τιμή στο DistanceFromDestination).

Public void ApproachTarget()

Χρησιμοποιεί την κλίση για να δώσει γωνία στο Solution για κάθε Joint.Ύστερα χρησιμοποιεί την MoveArm() των RobotJoints για να τους αλλάξει τη γωνία.

Η CalculateGradient() είναι ακριβώς ότι εξηγήσαμε στην παράγραφο 5.3 για την εύρεση της παραγώγου της συνάρτησης error.

Float DistanceFromTarget() μας επιστρέφει την ευκλείδεια απόσταση μεταξύ δύο σημείων. Τα σημεία τα παίρνουμε από το InverseKinematics() στην οποία βρίσκουμε πιο είναι το επόμενο σημείο στο οποίο πρέπει να κουνηθούμε όπως είδαμε στην παράγραφο 5.3