Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Σηματων, Ελεγχου και Ρομποτικης

Τεχνητό Νευρωνικό Δίκτυο για ανίχνευση των ποδιών του ανθρώπου και εκτίμηση της κατάστασης βάδισης από ρομποτικό περιπατητήρα μέσω αισθητήρων 2D LiDAR

Διπλωματικη Εργασια

της

ΕΥΣΤΑΘΙΟΥ ΔΑΝΑΗΣ

Επιβλέπων Καθηγητής: Κωνσταντίνος Τζαφέστας
Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, 22 Σεπτεμβρίου 2021

NATION TECHNICAL UNIVERSITY OF ATHENS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF SYSTEMS, CONTROL AND AUTOMATION

Deep Neural Network for Human Gait Tracking and Analysis using a 2D LiDAR on a Robotic Rollator

SENIOR THESIS

of

EFSTATHIOU DANAI

Supervisor : Constantinos Tzafestas
Assoc. Professor NTUA

Athens, 22nd September 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου και Ρομποτικής

Τεχνητό Νευρωνικό Δίκτυο για ανίχνευση των ποδιών του ανθρώπου και εκτίμηση της κατάστασης βάδισης από ρομποτικό περιπατητήρα μέσω αισθητήρων 2D LiDAR

Διπλωματικη Εργασια

της

ΕΥΣΤΑΘΙΟΥ ΔΑΝΑΗΣ

Επιβλέπων Καθηγητής: Κωνσταντίνος Τζαφέστας
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Σεπτεμβρίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Τζαφέστας	Πέτρος Μαραγκός	Αλέξανδρος Ποταμιάνος
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, 22 Σεπτεμβρίου 2021

(Υπογραφή)

.....

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Κωνσταντίνο Τζαφέστα για την συνεργασία μας, την άριστη επικοινωνία και την ελευθερία που μου έδωσε να κάνω πράξη την ιδέα μου για την συγκεκριμένη εργασία. Τον δόκτορα Αθανάσιο Δομέτιο και την ερευνήτρια Γεωργία Χαλβατζάκη για όλη την βοήθεια που μου προσέφεραν στην εκπόνηση της εργασίας, αλλά και την καθοδήγησή τους σε θέματα σπουδών. Μαζί τους έμαθα πως να οργανώνω την έρευνα μου, να βάζω στόχους σε κάθε βήμα και τελικά πως πρέπει αυτή να την παρουσιάζω.

Το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, τον πατέρα μου Παναγιώτη, την μητέρα μου Αναστασία, την αδερφή μου Νεφέλη, την γιαγιά μου Γαρυφαλλιά και την αγαπημένη μου θεία Αγγελική, που πάντα με στηρίζουν σε κάθε μου βήμα και κάθε μου απόφαση και ζουν όλες τις αγωνίες μου σαν δικές τους.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου και συμφοιτητές μου Διονύση, Γραμματική, Δημήτρη, Γιάννη και Αναστάση για την στήριξή τους, την παρέα τους και τον χρόνο που μου χάρισαν όλα αυτά τα χρόνια. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Διονύση για την βοήθειά του σε αυτή την διπλωματική εργασία και τις αμέτρητες συζητήσεις που κάναμε επί του θέματος.

Δανάη Π. Ευσταθίου
22 Σεπτεμβρίου 2021

Περίληψη

Η ανίχνευση των ποδιών του ανθρώπου και η ανάλυση της βάδισής του αποτελούν σημαντικές λειτουργίες για ένα ρομπότ υποβοήθησης βάδισης, όπως οι ευφυείς περιπατητήρες. Τέτοιου είδους περιπατητήρες είναι εξοπλισμένοι με πλήθος αισθητήρων, προκειμένου να εξαγάγουν πληροφορίες που αφορούν στον άνθρωπο που τους χειρίζεται και να προσαρμόσουν την λειτουργία τους στις ανάγκες αυτού. Σε αυτή την διπλωματική εργασία παρουσιάζουμε την ανάπτυξη του τεχνητού νευρωνικού δικτύου LTGADnet, το οποίο ανιχνεύει και παρακολουθεί την θέση των ποδιών του ανθρώπου κατά την διάρκεια της βάδισης, ενώ επίσης εκτιμά την κατάσταση της βάδισής του σε πραγματικό χρόνο, χρησιμοποιώντας δεδομένα από αισθητήρα απόστασης στις δύο διαστάσεις, τοποθετημένο σε ρομποτικό περιπατητήρα. Το νευρωνικό δίκτυο χρησιμοποιεί ένα Convolutional Neural Network (CNN) για την εξαγωγή των κέντρων των ποδιών από τα δεδομένα του αισθητήρα απόστασης, τα οποία δεδομένα αναπαρίστανται ως occupancy maps. Το CNN συνδυάζεται με ένα Long Short Term Memory (LSTM) νευρωνικό δίκτυο για την εκμάθηση των δυναμικών χαρακτηριστικών των ποδιών κατά την βάδιση, βελτιώνοντας την ανίχνευση που παράγει το CNN, ακόμα και σε περιπτώσεις που τα πόδια δεν είναι ορατά για κάποιο χρονικό διάστημα. Ακόμα, ένα δεύτερο LSTM εκτιμά την κατάσταση της βάδισης, δεχόμενο ως είσοδο τα ανιχνευμένα κέντρα των ποδιών. Το LTGADnet εκπαιδεύτηκε και δοκιμάστηκε σε δεδομένα εξαγμένα από πραγματικούς ασθενείς χειριζόμενους ευφυείς περιπατητήρες, καθώς και σε τεχνητά δεδομένα προσομοίωσης της βάδισης. Η επίδοσή του είναι σημαντικά υψηλότερη σε σχέση με την τρέχουσα χρησιμοποιούμενη μέθοδο, βελτιώνοντας τόσο την ακρίβεια της ανίχνευσης, όσο και τον χρόνο παραγωγής αυτής, καθιστώντας το συγκεκριμένο νευρωνικό δίκτυο μια ανταγωνιστική μέθοδο ανίχνευσης ποδιών σε πραγματικό χρόνο.

Λέξεις Κλειδιά

Αλληλεπίδραση Ανθρώπου-Ρομπότ, Ρομπότ υποβοήθησης βάδισης, Ανίχνευση και Παρακολούθηση ποδιών, Ανάλυση Βάδισης, Νευρωνικά Δίκτυα, CNN, LSTM

Abstract

Online human leg tracking and gait analysis are crucial functionalities for mobility assistant robots, like intelligent walkers. Usually, such walkers are equipped with various sensors for the extraction of human-related features for adaptive human-robot interaction and assistance. On this thesis the novel deep neural network LTGADnet is proposed for detecting and recognizing gait features from 2D range data produced by a laser sensor mounted on a robotic walker. An effective Convolutional Neural Network (CNN) is proposed as a powerful feature extractor for detecting the user's leg centers in range data represented as occupancy grid maps. The CNN is coupled with a Long Short Term Memory (LSTM) network for learning the legs' motion temporal dynamics while walking, improving the prior detection, and providing better leg occlusion handling. A gait analysis is also performed by recognizing gait phases over both legs by feeding the leg tracking output to a subsequent LSTM. The LTGADnet framework has been trained and tested on data derived from real patients, as well as computer generated data. The presented experimental results show the network's efficiency in providing accurate detections compared to state-of-the-art and ability to be applied on an online system due to its high frequency, making it a competitive method for gait detection on robotic mobility assistants.

Keywords

Human-Robot Interaction, Walking Assistive Robots, Leg Detection and Tracking, Gait Analysis, Neural Networks, CNN, LSTM

Contents

Εκτενής Περίληψη στα Ελληνικά

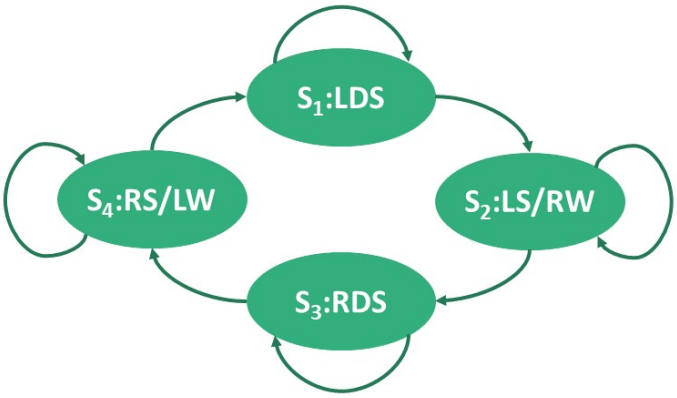
Σε αυτό το κεφάλαιο παρουσιάζεται η διπλωματική εργασία που εκπονήθηκε μέσω μιας εκτεταμένης περίληψης στα ελληνικά.

Συνεισφορά

Σε αυτή τη διπλωματική εργασία παρουσιάζουμε ένα καινοτόμο νευρωνικό δίκτυο βαθιάς εκμάθησης για την παρακολούθηση των ποδιών και την ανάλυση της βάδισης του ανθρώπου μέσω δεδομένων αισθητήρα απόστασης στις δύο διαστάσεις (LTGADnet). Η προτεινόμενη προσέγγιση χρησιμοποιεί ένα συνελικτικό νευρωνικό δίκτυο (CNN) για την ανίχνευση των ποδιών, ακολουθούμενο από ένα επαναλαμβανόμενο νευρωνικό δίκτυο (RNN) για την αξιοποίηση της χρονικής πληροφορίας που περικλείει η βάδιση και την αναθεώρηση της ανίχνευσης που παράγει το CNN σε απαιτητικές καταστάσεις που κάποιο πόδι αποκρύπτεται. Τέλος, χρησιμοποιείται ένα δεύτερο RNN για την εξαγωγή της χρονικής αλληλεπίδρασης των ποδιών, εκ της οποίας μπορεί να εκτιμηθεί η τρέχουσα κατάσταση βάδισης. Η συνεισφορά της εργασίας μας είναι η υψηλής ακρίβειας μέθοδος παρακολούθησης των ποδιών, χάρις στην βαθιά εξαγωγή χαρακτηριστικών, και η ευκολία χρήσης της μεθόδου αυτής σε κάθε ρομπότ υποβοήθησης της βάδισης, λόγω της αποτελεσματικότητας και ταχύτητας εκτέλεσής της. Πειραματικά δεδομένα χρήσης της μεθόδου εξήχθησαν με τη χρήση της βάσης δεδομένων LTGAD, που δημιουργήσαμε αποκλειστικά για τον έλεγχο και την εκπαίδευση του δικτύου, και περιλαμβάνει δεδομένα και ετικέτες από αληθινούς ασθενείς χρήστες ρομπότ υποβοήθησης βάδισης. Τα παραγόμενα αποτελέσματα επιδεικνύουν την βελτίωση της επίδοσης που προσφέρει η συγκεκριμένη μέθοδος σε σχέση με την τρέχουσα χρησιμοποιούμενη μέθοδο σε τέτοιου είδους εφαρμογές [1].

Πίνακας 1: Οι καταστάσεις βάδισης όπως ορίζονται στο [1] και η one-hot αναπαράστασή τους.

Κωδικός Κατάστασης	One-Hot	Κωδική Ονομασία	Ορισμός
s_1	[1, 0, 0, 0]	LDS	Αριστερή Διπλή Στήριξη
s_2	[0, 1, 0, 0]	LS/RW	Αριστερή Στάση/Δεξιά Αιώρηση
s_3	[0, 0, 1, 0]	RDS	Δεξιά Διπλή Στήριξη
s_4	[0, 0, 0, 1]	RS/LW	Δεξιά Στάση/Αριστερή Αιώρηση



Σχήμα 1: Διάγραμμα με τις πιθανές καταστάσεις βάδισεις και τις μεταβάσεις ανάμεσά τους.

Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο παρουσιάζουμε ορισμένες έννοιες των νευρωνικών δικτύων απαραίτητες για την κατανόηση αυτής της διπλωματικής εργασίας.

Τεχνητά Νευρωνικά Δίκτυα

Ένα τεχνητό νευρωνικό δίκτυο είναι ένα σύστημα που χρησιμοποιεί ένα σύνολο από κόμβους, ονομαζόμενους **νευρώνες**, και συνδέσεις μεταξύ τους, ονομαζόμενες **συνάψεις**, προκειμένου να μάθει πως να επιλύει ένα πρόβλημα τεχνητής νοημοσύνης. Οι νευρώνες αυτοί ανήκουν είτε στο **στρώμα εισόδου**, που αποτελεί το τμήμα του δικτύου που λαμβάνει την είσοδο, είτε σε κάποιο **κρυφό στρώμα**, τα οποία είναι προαιρετικά και συνεισφέρουν στην περαιτέρω επεξεργασία των δεδομένων, είτε στο **στρώμα εξόδου**, το οποίο αναπαριστά την τελική έξοδο του δικτύου. Τα σήματα που λαμβάνονται από τους νευρώνες του στρώματος εισόδου προωθούνται πρώτα στα κρυφά στρώματα και έπειτα στο στρώμα εξόδου. Το πιο απλό είδος στρώματος είναι το **fully-connected**, στο οποίο όλοι οι νευρώνες του συνδέονται με σύναψη με κάθε νευρώνα του προηγούμενου στρώματος.
Κάθε νευρώνας του δικτύου που λαμβάνει ένα σήμα το επεξεργάζεται και στη συνέχεια το μεταδίδει στους νευρώνες με τους οποίους συνδέεται μέσω των συνάψεων. Η επεξεργασία των σημάτων αποτελέι στην ουσία έναν γραμμικό συνδυασμό αυτών, με **βάρη** καθορισμένα από τις συνάψεις, ένα προεραιτικό προστιθέμενο **bias** και πέρασμα αυτού του αποτελέσματος από την **συνάρτηση activation** του νευρώνα. Όταν η συγκεκριμένη συνάρτηση είναι μη γκραμμική, επιτρέπει στο μοντέλο να μαθαίνει να επιλύει μη τετριμμένα προβλήματα, χρησιμοποιώντας λιγότερους νευρώνες.
Για την εκπαίδευση κάθε νευρωνικού δικτύου είναι απαραίτητη η δημιουργία ενός συνόλου **δεδομένων με ετικέτες** (labeled data). Οι ετικέτες των δεδομένων αποτελούν την επιθυμητή έξοδο του νευρωνικού δικτύου, όταν αυτό δεχτεί ως είσοδο τα συγκεκριμένα δεδομένα. Αυτές οι ετικέτες χρησιμοποιούνται για τον υπολογισμό του **κόστους**, το οποίο υπολογίζεται από την **συνάρτηση κόστους** του νευρωνικού και αναπαριστά το πόσο καλά προσεγγίζει η έξοδος του νευρωνικού την επιθυμητή έξοδο. Αυτό το σύνολο δεδομένων μπορεί να χωρίζεται

σε τρία μέρη: (1) το **training set**, που χρησιμοποιείται για την εκπαίδευση του δικτύου, (2) το **validation set**, που είναι προαιρετικό και χρησιμοποιείται κατά την διάρκεια της εκπαίδευσης για την θεώρηση της επίδοσης του μοντέλου και (3) το **test set**, που χρησιμοποιείται για τον τελικό έλεγχο του μοντέλου, όταν όλες οι αρχιτεκτονικές αποφάσεις έχουν ληφθεί και δεν αναμένεται να αναθεωρηθούν.
Η εκπαίδευση ενός νευρωνικού δικτύου γίνεται μέσω του **backpropagation**. Με αυτόν τον αλγόριθμο υπολογίζεται η κλίση στον χώρο των βαρών ενός νευρωνικού σε σχέση με την συνάρτηση κόστους.

Συνελικτικά Νευρωνικά Δίκτυα (CNN)

Τα συνελικτικά νευρωνικά δίκτυα (CNN) επιτελούν στα κρυφά τους στρώματα συνελίξεις και χρησιμοποιούνται κυρίως στην ανάλυση εικόνας. Κάθε συνελικτικό στρώμα που λαμβάνει την είσοδό του από το προηγούμενο στρώμα την συνελίσσει με ένα πλήθος από **φίλτρα** συγκεκριμένων διαστάσεων και παράγει πολλαπλά κανάλια εξόδου, όσα και τα φίλτρα. Η πράξη της συνέλιξης ανάμεσα σε ένα φίλτρο και μια εικόνα είναι ο γραμμικός συνδυασμός των κελιών της εικόνας με βάρη ορισμένα από το φίλτρο. Το πλεονέκτημα των συνελικτικών στρωμάτων έναντι των υπολοίπων είναι η ταχύτητα επεξεργασίας εισόδων μεγάλων διαστάσεων, όπως για παράδειγμα των εικόνων, αφού οι συνάψεις μεταξύ των νευρώνων σε αυτά τα στρώματα είναι πολύ λιγότερες από ό,τι για παράδειγμα στα fully-connected. Τα συνελικτικά στρώματα μπορεί να έχουν κι άλλες παραμέτρους, όπως είναι το **padding**, το **stride** και το **dilation**. Επίσης, τα CNN μπορεί να συνοδεύονται και από άλλα στρώματα πέραν των συνελικτικών, όπως στρώματα **pooling**, **normalization** ή **fully-connected**.

Επαναλαμβανόμενα Νευρωνικά Δίκτυα (RNN)

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) αποτελούν μια κλάση νευρωνικών δικτύων όπου οι συνάψεις μεταξύ των νευρώνων σχηματίζουν έναν ισχυρά συνεκτικό γράφο, που σημαίνει ότι υπάρχει μονοπάτι από κάθε νευρώνα σε κάθε άλλο. Αυτό σημαίνει ότι το RNN μπορεί να διατηρήσει κάποιας μορφής **μνήμη** και να τη χρήσιμοποιήσει προκειμένου να εξαγάγει χρονικές πληροφορίες πάνω στην είσοδο. Η είσοδος ενός RNN είναι συνήθως μία απροσδιόριστου μήκους ακολουθία. Τα RNN χρησιμοποιούνται κυρίως στην αναγνώριση ομιλίας ή γραφικού χαρακτήρα, αλλά μπορούν να φανούν χρήσιμα σε κάθε εφαρμογή με χρονικά συσχετιζόμενες εισόδους. Συχνοί τύποι RNN είναι το Long Short-Term Memory (LSTM) [2] και το Gated Recurrent Unit (GRU) [3], που αποτελεί απλουστευμένη μορφή του LSTM.

Χρονικά Συνελικτικά Δίκτυα (TCN)

Τα χρονικά συνελικτικά δίκτυα (TCN) αναπτύχθηκαν ως εναλλακτικά αντικατάστατα των RNN με σκοπό τη μείωση του χρόνου εκπαίδευσης και την αύξηση της διατήρησης της μνήμης. Αποτελούνται από dilated 1D συνελικτικά στρώματα με ίδιες διαστάσεις εισόδου και εξόδου. Η είσοδος ενός TCN είναι ένας 3D πίνακας μεγέθους (B, L, S_{in}), όπου B το batch size, L το μήκος της ακολουθίας και S_{in} το μήκος κάθε στοιχείου της ακολουθίας. Κάθε κρυφό στρώμα

ενός TCN έχει εκθετικά αυξημένο dilation σε σχέση με το προηγούμενο στρώμα. Προκειμένου η είσοδος και η έξοδος να έχουν τις ίδιες διαστάσεις χρησιμοποιείται zero-padding. Τα TCN έχουν αποδειχτεί ιδιαιτέρως αποδοτικά σε περιπτώσεις αναγνώρισης δράσης σε βίντεο [4] και γενικότερα μοντελοποίησης ακολουθιών [5], για αυτό και δοκιμάστηκε η χρήση τους στο LTGADnet έναντι του LSTM.

Η Βάση Δεδομένων για Παρακολούθηση Ποδιών και Ανάλυση Βάδισης

Σε αυτή την ενότητα παρουσιάζουμε την βάση LTGAD, μιά βάση δεδομένων που δημιουργήσαμε για την εκπαίδευση, την επαλήθευση και τον έλεγχο του δικτύου παρακολούθησης ποδιών και ανάλυσης βάδισης LTGADnet. Η βάση αυτή είναι δημοσιευμένη και προσβάσιμη μέσω αυτού του συνδέσμου. Περιλαμβάνει δεδομένα με τις ετικέτες τους εξαγμένα για το σκοπό προηγούμενης εργασίας [1], τα οποία συλλέχθηκαν και διαμορφώθηκαν έτσι ώστε να μπορούν να χρησιμοποιηθούν από την τρέχουσα εργασία. Ακόμα, περιλαμβάνει δεδομένα κατασκευασμένα από υπολογιστή που «μιμούνται» τα πραγματικά δεδομένα, με σκοπό τον εμπλουτισμό της ήδη υπάρχουσας βάσης.

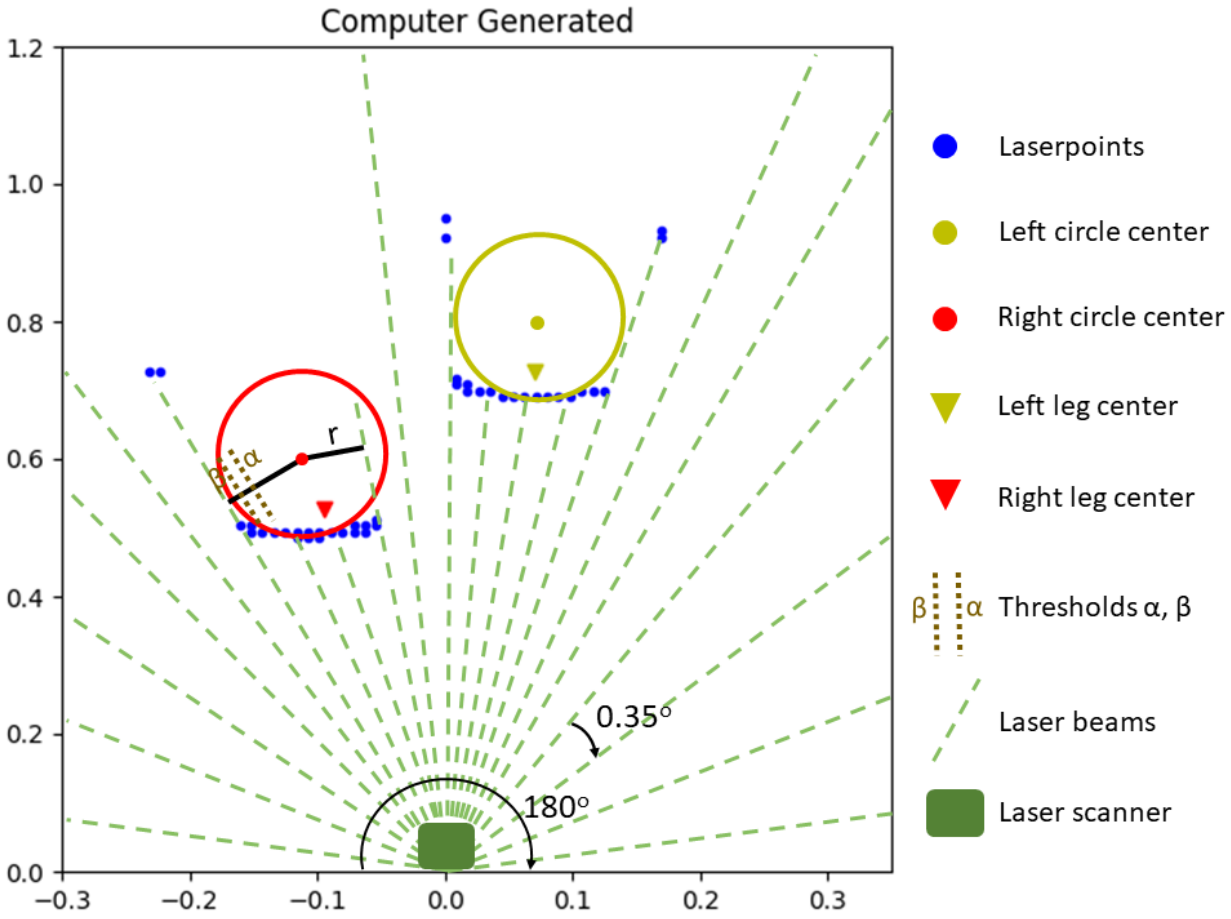
Δομή Βάσης

Η βάση LTGAD χωρίζεται σε δύο τμήματα: την βάση παρακολούθησης ποδιών και την βάση ανάλυσης βάδισης. Κάθε μία εξ' αυτών αποτελείται από δεδομένα εξαγμένα από 8 διαφορετικά πειράματα, καθώς και δεδομένα ενός υπολογιστικά προσομοιωμένου πειράματος, με τέτοιο μέγεθος ώστε το τελικό ποσοστό των τεχνητών δεδομένων στο σύνολο όλων των δεδομένων να προκύπτει περίπου 50%.

Πειραματική Διάταξη

Τα πειράματα που περιλαβάνονται στην βάση LTGAD προέρχονται από πραγματικούς ασθενείς και πραγματοποιήθηκαν στο πλαίσιο της εργασίας [1]. Οι συμμετέχοντες στα πειράματα ήταν άνω των 65 χρόνων και παρουσίαζαν από ήπια έως μέτρια κινητικά προβλήματα, έπειτα από κλινική αξιολόγηση. Για την τέλεση των πειραμάτων ζητήθηκε από τους ασθενείς να πραγματοποιήσουν σενάρια βάδισης και στροφής υποβοηθούμενοι από έναν ρομποτικό περιπατητήρα. Όλα τα πειράματα εξήχθησαν με απόλυτη ασφάλεια και υπό την επίβλεψη εξειδικευμένων νοσοκόμων.

Ο περιπατητήρας ήταν εξοπλισμένος με ένα αισθητήρα λέιζερ Hokuyo rapid UBG-04LX-F01 τοποθετημένο σε ύψος περίπου $40cm$ από το έδαφος, προκειμένου να συλλάβει την κίνηση των κνημών του ασθενούς. Για την εκτίμηση της κατάστασης βάδισης χρησιμοποιήθηκε ένα σύνολο από markers ενός συστήματος VICON Motion Capture τοποθετημένων στις φτέρνες και τα δάχτυλα των ποδιών του ασθενούς, όπως περιγράφεται στο [6].



Σχήμα 2: Αναλυτική παρουσίαση της μεθόδου που χρησιμοποιήθηκε για την παραγωγή των τεχνητών δεδομένων. Το μήκος των ακτίνων λέιζερ αναπαριστά την απόσταση από το ανιχνευμένο εμπόδιο, όπως αυτή υπολογίζεται από τον αισθητήρα.

Τεχνητά Δεδομένα

Για την εκπαίδευση του δικτύου LTGADnet είναι σημαντική η χρήση μιας εκτεταμένης και ακριβούς βάσης δεδομένων. Τα δεδομένα από τους πραγματικούς ασθενείς αφενός χάνουν σε ακρίβεια, λόγω των δυσκολιών που εμφανίζονται σε τέτοιου είδους πειράματα, αφετέρου αποδείχτηκαν περιορισμένα σε ποικιλία ασθενών για την εκτίμηση της κατάστασης βάδισης. Όταν τα δεδομένα για την εκπαίδευση του δικτύου περιλαμβάνουν πολλούς διαφορετικούς ασθενείς, τότε το δίκτυο μπορεί να μάθει να αναγνωρίζει μοτίβα στο περπάτημα αυτών, χωρίς να απομνημονεύει αυτά που εντόπισε κατά την εκπαίδευσή του. Για τους παραπάνω λόγους αποφασίσαμε να δοκιμάσουμε την κατασκευή τεχνητών δεδομένων.

Η κεντρική ιδέα είναι η προσομοίωση της διάταξης του περιπατητήρα και των ποδιών του ασθενούς. Θεωρείται ότι ένας αισθητήρας λέιζερ βρίσκεται στο κέντρο των αξόνων και βλέπει προς τα πόδια του ασθενούς. Ο συγκεκριμένος αισθητήρας θεωρείται ότι έχει 180° εύρος έκτασης σαρώματος και $\sim 0.35°$ ανάλυση γωνίας.

Τα πόδια του ασθενούς αναπαρίστανται από δύο κύκλους κινούμενους ημιτονοειδώς στους x, y άξονες. Τα ημίτονα κίνησης των δύο ποδιών στον y άξονα έχουν διαφορά φάσης π, προκειμένου να εκτελούν αντίθετες κινήσεις. Το ημίτονο κίνησης στον x άξονα είναι αρκετά μικρότερου πλάτους και ελαφρώς ακανόνιστης συχνότητας, προκειμένου να προσομοιωθεί η περιστασιακή μετατόπιση ολόκληρης της βάδισης προς τα δεξιά ή τα αριστερά. Ακόμα, η γωνία της βάδισης αλλάζει με το πέρασμα του χρόνου, προκειμένου να δημιουργηθούν εικονικές στροφές και να παραχθούν δεδομένα με occlusions, στα οποία το ένα πόδι του ασθενούς κρύβει το άλλο, όσον αφορά την οπτική του αισθητήρα.
Για την εικονική δειγματοληψία από τον αισθητήρα λέιζερ κάθε ακτίνα λέιζερ προσομοιώνεται ως μία ευθεία γραμμή με έναρξη την αρχή των αξόνων, όπου βρίσκεται και ο αισθητήρας (Σχήμα 2). Τα κοντινότερα στην έναρξη σημεία τομής της ευθείας με τους κύκλους θεωρούνται τα σημεία στα οποία ο αισθητήρας ανιχνεύει εμπόδια. Σε αυτά τα σημεία προστίθεται Γκαουσιανός θόρυβος για να προσεγγιστεί το ονομαστικό σφάλμα του αισθητήρα. Ακόμα, προσομοιώνονται τα λανθασμένα ίχνη που παρουσιάζονται συχνά σε δεδομένα 2D αισθητήρα. Αυτά τα ίχνη είναι ορατά στα δεδομένα όταν είναι μεγάλη η γωνία πρόσπτωσης των ακτίνων του λέιζερ σε ένα εμπόδιο και λόγω διάχυσης επιστρέφεται στον αισθητήρα σήμα μικρότερης έντασης, συμπεραίνοντας έτσι μεγαλύτερη απόσταση από το εμπόδιο από την πραγματική.
Τέλος, για τον υπολογισμό της κατάστασης βάδισης σε κάθε χρονική στιγμή, χρησιμοποιούμε μόνο την κίνηση του δεξιού ποδιού στον y άξονα. Οι φάσεις κατά τις οποίες ο άνθρωπος αλλάζει κατάσταση βάδισης χρησιμοποιήθηκαν όπως αυτές ορίζονται στα [7, 8]. Υποθέτουμε ότι κάθε πόδι παραμένει σε στάση για 60% ενός κύκλου βάδισης και για 40% σε αιώρηση, πράγμα που σημαίνει ότι κάθε κύκλος βάδισης αποτελείται από 10% LDS, 40% LS/RW, 10% RDS and 40% RS/LW (Πίνακας 1).

Το Νευρωνικό Δίκτυο LTGADnet

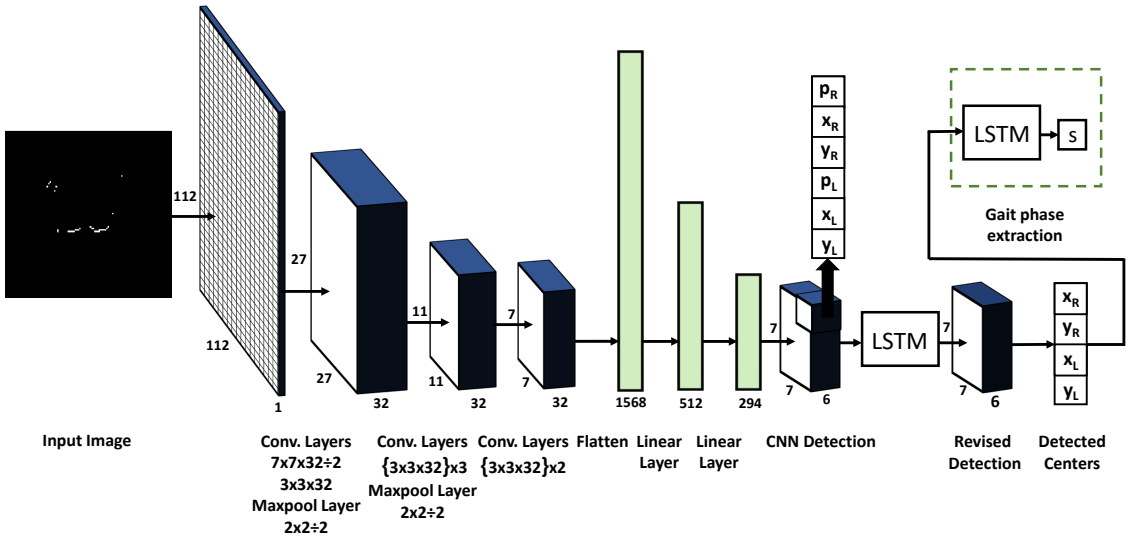
Σε αυτή την ενότητα παρουσιάζεται η αρχιτεκτονική του νευρωνικού δικτύου LTGADnet. Αναλύονται τα τμήματά του και οι αρχιτεκτονικές αποφάσεις που λήφθηκαν για καθένα εξ΄ αυτών.
Το LTGADnet αποτελείται από δύο δίκτυα:

1. ένα παρακολούθησης ποδιών, που είναι υπεύθυνο για την ανίχνευση και παρακολούθηση των ποδιών του ασθενούς που χειρίζεται των ρομποτικό περιπατητήρα και

2. ένα ανάλυσης βάδισης, το οποίο είναι υπεύθυνο για την εξαγωγή της κατάστασης βάδισης του ασθενούς, όπως αυτές ορίζονται στον Πίνακα 1.

Το Δίκτυο Παρακολούθησης Ποδιών

Το δίκτυο παρακολούθηση ποδιών αναλαμβάνει την λήψη των δεδομένων που παρέχει το λέιζερ και τον εντοπισμό των κέντρων των ποδιών του ασθενούς, ακόμα και σε περιπτώσεις occlusion. Θεωρείται ότι δεν υπάρχουν άλλα πόδια ή αντικείμενα που κρύβουν τα πόδια του



Σχήμα 3: Η αρχιτεκτονική του νευρωνικού δικτύου για παρακολούθηση ποδιών και ανάλυση βάδισης. Αποτελείται από ένα CNN ακολουθούμενο από ένα LSTM για την παρακολούθηση των ποδιών και ένα LSTM για την εκτίμηση της κατάστασης βάδισης.

ασθενούς και ότι τα τελευταία παραμένουν καθ' όλη την διάρκεια εντός του οπτικού πεδίου του λέιζερ.

Αρχικά, οι συντεταγμένες των εμποδίων που επιστρέφει το λέιζερ μετατρέπονται από πολικές σε καρτεσιανές. Από αυτές τις συντεταγμένες κρατάμε μόνο ορισμένες για την δημιουργία της τελικής εισόδου, με βάση ένα συγκεκριμένο $1m \times 1m$ πλαίσιο. Ο σκοπός του συγκεκριμένου πλαισίου είναι να περιορίσει την ύπαρξη πιθανών άλλων αντικειμένων πέραν των ποδιών στην είσοδο. Εμείς επιλέξαμε το πλαίσιο αυτό να έχει όρια $(-1.5m, 1.5m)$ και $(0.2m, 1.2m)$ στον x και y άξονα αντίστοιχα.

Στη συνέχεια δημιουργούμε την εικόνα εισόδου του δικτύου. Η εικόνα αυτή έχει μέγεθος 112×112 και αποτελεί ένα occupancy grid των δεδομένων του αισθητήρα. Συγκεκριμένα, ένα occupancy grid αναπαριστά τον χώρο μπροστά από το λέιζερ και περιέχει σε κάθε κελί του την τιμή 1, αν σε εκείνη τη θέση έχει εντοπισθεί εμπόδιο και την τιμή 0, αν σε εκείνη τη θέση δεν υπάρχει εμπόδιο ή δεν έχει εντοπισθεί.

Η έξοδος του δικτύου έχει διαστάσεις $6 \times 7 \times 7$. Το 7×7 τμήμα αναπαριστά ένα grid με το οποίο τέμνουμε την αρχική εικόνα. Για κάθε κελί αυτού του grid, το δίκτυο δίνει ως έξοδο 6 τιμές, 3 για κάθε πόδι:

1. Την πιθανότητα το κέντρο του ποδιού να υπάρχει σε αυτό το κελί.

2. Αν το κέντρο υπάρχει, την συντεταγμένη x αυτού, η οποία λαμβάνει τιμές στο $[0, 1)$ και αναπαριστά τη σχετική θέση στον x του κέντρου μέσα στο κελί.

3. Αν το κέντρο υπάρχει, την συντεταγμένη y αυτού, η οποία λαμβάνει τιμές στο $[0, 1)$ και αναπαριστά τη σχετική θέση στον y του κέντρου μέσα στο κελί.

Το δίκτυο αποτελείται από δύο υποδίκτυα: (1) ένα CNN για την ανίχνευση ποδιών και (2) ένα LSTM για την διαχείριση περιπτώσεων occlusion. Τα στρώματα που απαρτίζουν το CNN είναι τα εξής: $7\times7\times32\div2$ Convolution (μέγεθος πυρήνα 7, αριθμός φίλτρων 32, stride 2), $3\times3\times32$ Convolution, $2\times2\div2$ Max Pooling, $3\times3\times32$ Convolution, $3\times3\times32$ Convolution, $3\times3\times32$ Convolution, $2 \times 2 \div 2$ Max Pooling, $3 \times 3 \times 32$ Convolution, $3 \times 3 \times 32$ Convolution. Στη συνέχεια ακολουθούν δύο στρώματα fully-connected διαστάσεων 1568×500 και 500×294, όπου $294 = 7 \times 7 \times 6$ το μέγεθος του grid. Αυτή είναι και η έξοδος του CNN. Τέλος, ένα απλό LSTM ενός στρώματος λαμβάνει ως είσοδος την έξοδο του CNN και δίνει ως έξοδο ένα grid ίδιων διαστάσεων. Κάθε στρώμα του δικτύου έχει ως συνάρτηση activation την ReLU συνάρτηση.
Για την εκπαίδευση του δικτύου, ως συνάρτηση κόστους χρησιμοποιήθηκε η παρακάτω:

$$\textit{Confidence Loss} = \sum_{i=1}^{7}\sum_{j=1}^{7}\sum_{k=1}^{2}(C_{ijk} - \hat{C}_{ijk})^2$$

$$\textit{Detection Loss} = \sum_{i=1}^{2}(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

$$\textit{Association Loss} = \sum_{i=1}^{2}(\hat{x}_i^t - \hat{x}_i^{t-1})^2 + (\hat{y}_i^t - \hat{y}_i^{t-1})^2$$

$$\textit{Loss} = \textit{Confidence Loss} + 5 \cdot \textit{Detection Loss} + 0.01 \cdot \textit{Association Loss}$$

όπου C_{ijk} είναι η επιθυμητή πιθανότητα εμφάνισης κέντρου ποδιού στο κελί ij για το πόδι k (1 αν το κελί περιέχει το κέντρο του ποδιού k και 0 διαφορετικά), $\hat{C}_{ijk}$ η πιθανότητα στην έξοδο του δικτύου για το κελί ij και το πόδι k, x_i, y_i η επιθυμητή θέση του κέντρου του ποδιού i στο grid, $\hat{x}_i, \hat{y}_i$ η θέση στην έξοδο του δικτύου του κέντρου του ποδιού i στο grid και $\hat{x}_i^t, \hat{y}_i^t, \hat{x}_i^{t-1}, \hat{y}_i^{t-1}$ οι συντεταγμένες x, y του ποδιού i στο grid τις χρονικές στιγμές t και $t-1$ αντίστοιχα.

Το Δίκτυο Ανάλυσης Βάδισης

Το δίκτυο ανάλυσης βάδισης λαμβάνει ως είσοδο τα κέντρα των ποδιών, όπως αυτά προκύπτουν από την έξοδο του πρώτου LSTM. Ως έξοδο δίνει τη one-hot κωδικοποίηση της εκτιμώμενης κατάστασης βάδισης, όπως αυτές ορίζονται στον Πίνακα 1. Το LSTM έχει 5 στρώματα, με καθένα εξ' αυτών να διατηρεί ανεξάρτητη μνήμη. Αυτά τα LSTM ακολουθούν το ένα το άλλο και η συνολική έξοδος είναι η έξοδος του τελευταίου στρώματος. Έτσι, κάθε στρώμα έχει διαστάσεις 4×4.

Για την εκπαίδευση του δικτύου χρησιμοποιήθηκε ως συνάρτηση κόστους η cross-entropy loss:

$$f_i(s) = \frac{e^{s_i}}{\sum_{j=1}^{C} e^{s_j}}$$
$$loss = -\sum_{i=1}^{C} t_i \log\left(f_i(s)\right)$$

όπου $f_i(s)$ το αποτέλεσμα της συνάρτησης Softmax για κάθε πιθανότητα κλάσης i, t_i η επιθυμητή πιθανότητα κλάσης και $loss$ το τελικό κόστος. Καθώς το πρόβλημα κατάστασης βάδισης, αν αντιμετωπισθεί ως πρόβλημα κατηγοριοποίησης, διακατέχεται από μια εγγενή ανισορροπία κλάσεων, με μόνο το 19% και το 18.6% των καταστάσεων να ανήκουν στην κατάσταση LDS και RDS αντίστοιχα, χρησιμοποίησαμε αναλογικά αντίστροφα βάρη στην συνεισφορά των κλάσεων στο cross entropy loss.

Το δίκτυο ανάλυσης βάδισης παρουσίασε μεγάλη αστάθεια κατά την εκπαίδευσή του. Ήταν συχνό το φαινόμενο το δίκτυο να μαθαίνει επιτυχώς στο training και στο validation set, αλλά η επίδοσή του να πέφτει σε μεγάλο βαθμό στο test set. Την συμπεριφορά αυτή την αποδίδουμε στην βάση δεδομένων, στον περιορισμένο αριθμό ασθενών και στην ανισορροπία των κλάσεων, η οποία δεν μπορεί να αντιμετωπισθεί πλήρως με την ρύθμιση των βαρών των κλάσεων στο cross entropy loss.

Πειραματικά Αποτελέσματα

Σε αυτό το κεφάλαιο παρουσιάζουμε τα πειραματικά αποτελέσματα του τελικού νευρωνικού δικτύου, τόσο στο κομμάτι της παρακολούθησης των ποδιών, όσο και σε αυτό της ανάλυσης βάδισης. Χρησιμοποιούμε ως μετρικές στον έλεγχο της παρακολούθησης ποδιών την μέση και την μέγιστη απόσταση, καθώς και την διάμεσο των αποστάσεων μεταξύ των πραγματικών και των εκτιμώμενων κέντρων των ποδιών. Για τον έλεγχο της ανάλυσης βάδισης εξάγουμε την ποσοστιαία ακρίβεια και τις μετρικές Precision, Recall και F1-score.

Στον Πίνακα 2 φαίνεται ότι το δίκτυό μας παρουσιάζει αύξηση κατά 52% στην ακρίβεια της εκτίμησης των κέντρων ων ποδιών σε σχέση με την τρέχουσα χρησιμοποιούμενη μέθοδο [1], η οποία, λόγω της χρήσης των particle filters είναι επιπλέον αρκετά χρονοβόρα και δεν μπορεί εύκολα να χρησιμοποιηθεί σε οποιονδήποτε ρομποτικό περιπατητήρα. Ακόμα, η ακρίβεια του δικτύου ανάλυσης βάδισης είναι ικανοποιήτικη, αλλά όχι αρκετή για την χρήση του δικτύου για εξαγωγή δεδομένων που αφορούν στην υγεία του ασθενούς. Πιστεύουμε πως ο εμπλουτισμός της βάσης με επιπλέον δεδομένα θα έπαιζε καθοριστικό ρόλο την βελτίωση της ακρίβειας του δικτύου.

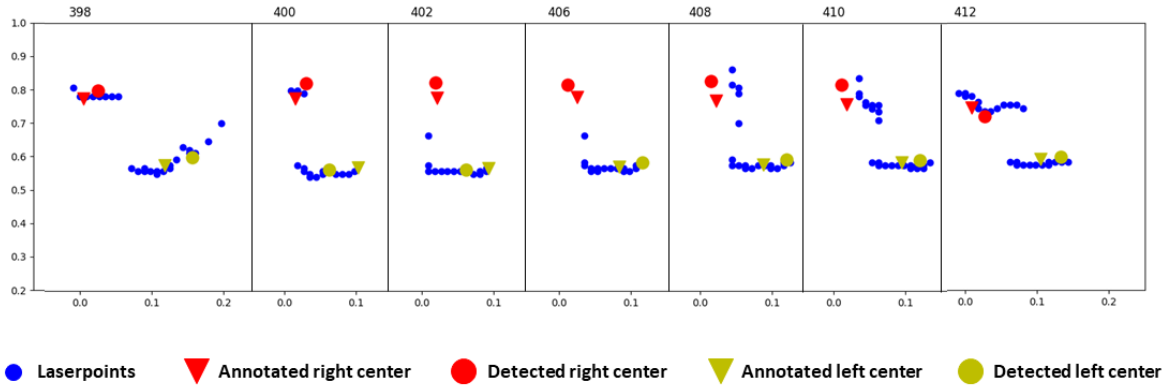
Ακόμα παρουσιάζουμε μερικά παραδείγματα εξόδου του νευρωνικού δικτύου.

Πίνακας 2: Αποτελέσματα θεώρησης Leave-one-out

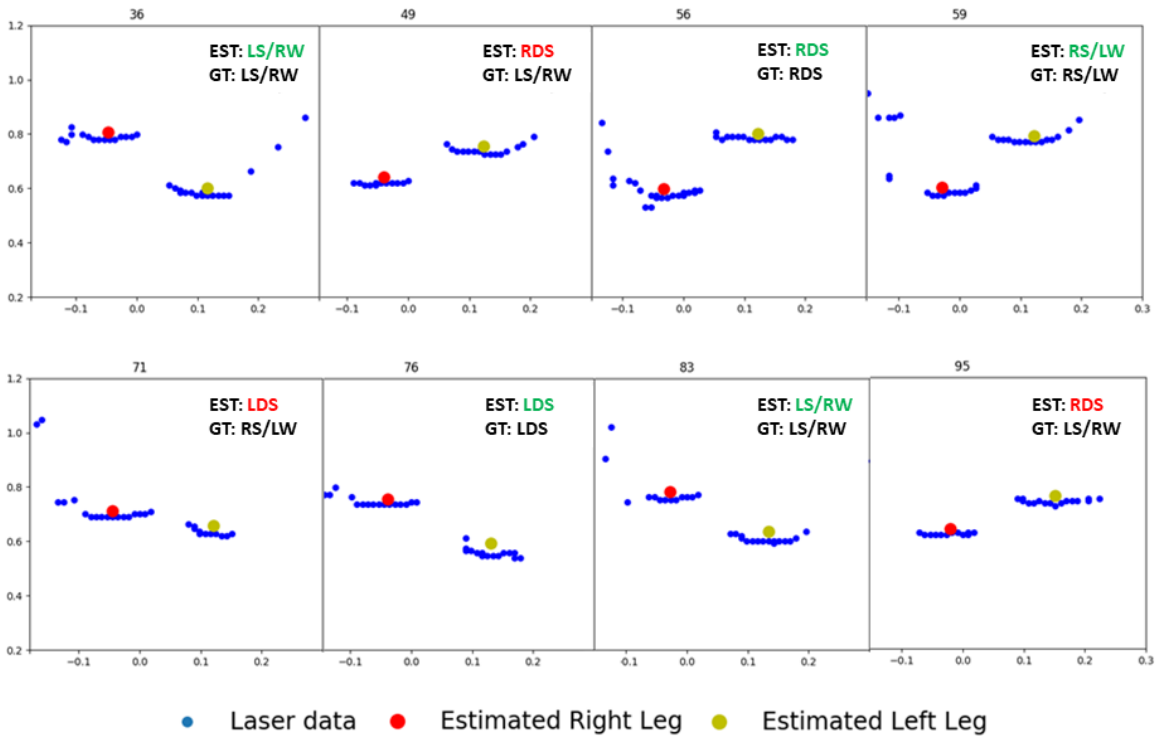
Παρακολούθηση Ποδιών									
Μετρική \ Πείραμα	1	2	3	4	5	6	7	8	μέση τιμή
Μέση τιμή (cm)	2.42	2.04	3.28	4.11	4.57	2.46	3.26	3.68	3.23
Μέγιστη τιμή (cm)	9.63	9.87	22.16	58.32	14.59	7.79	30.31	42.95	24.45
Διάμεσος (cm)	2.21	1.81	2.60	2.95	3.74	2.18	2.75	3.31	2.69
Ανάλυση Βάδισης									
Μετρική \ Πείραμα	1	2	3	4	5	6	7	8	μέση τιμή
Ακρίβεια (%)	75.60	71.56	68.85	72.74	69.27	63.17	69.36	75.89	70.805
Precision (%)	78.88	71.36	68.87	74.80	70.76	67.85	69.38	77.50	72.425
Recall (%)	75.60	71.57	68.85	72.74	69.27	63.17	69.37	75.89	70.8075
F1 score (%)	76.57	71.28	67.62	72.10	68.68	61.94	69.21	76.24	70.455

Στο Σχήμα 4 επιδεικνύεται η συμπεριφορά του δικτύου παρακολούθησης βάδισης σε περίπτωση occlusion. Παρατηρούμε ότι το δίκτυο καταφέρνει επιτυχώς να παρακολουθήσει το κέντρο του δεξιού ποδιού, ακόμα και όταν αυτό δεν είναι ορατό (διαγράμματα $\#400 - \#408$). Ακόμα, είναι εμφανής η ικανότητα του δικτύου να εξαγάγει με ακρίβεια τα κέντρα των ποδιών.

Στο Σχήμα 5 βλέπουμε ένα παράδειγμα εξόδου του δικτύου ανάλυσης βάδισης. Από τις 60 χρονικές στιγμές που αναφέρονται στο σχήμα, σε 13 παρατηρούμε λανθασμένη εκτίμηση κατάστασης βάδισης, με αποτέλεσμα να προκύπτει ακρίβεια περίπου 78.33%, ποσοστό ικανοποιητικό, με αρκετές όμως προοπτικές βελτίωσης σε περίπτωση χρήσης μιας ευρύτερης βάσης δεδομένων.



Σχήμα 4: Παράδειγμα εξόδου του νευρωνικού δικτύου παρακολούθησης ποδιών σε δεδομένα με occlusion.



Σχήμα 5: Παράδειγμα εξόδου του δικτύου ανάλυσης βάδισης. Στο σχήμα σημειώνονται τα εκτιμώμενα κέντρα των ποδιών, καθώς και η εκτιμώμενη κατάσταση βάδισης σε σχέση με την πραγματική. Απεικονίζονται μόνο οι χρονικές στιγμές στις οποίες υπάρχει αλλαγή στην εκτιμώμενη ή στην πραγματική κατάσταση βάδισης.

Chapter 1

Introduction

Functional walking is an integral part of every person's daily life. When mobility disabilities emerge, usually with age, depriving a human of a vital ability taken for granted, they affect the individual physically and emotionally [9]. With the constantly increasing life expectancy arises the need for finding solutions to mobility disabilities, which do not require invasive methods and are well suited to every individual. A lot of research has been conducted in this direction, involving conventional wheelchairs, walkers and canes, as well as intelligent assistive devices.

1.1 Motivation

A mobility-impaired person can be prone to fall incidents, which can easily cause injuries that could provoke further, often more severe, problems. The need for providing mobility assistants to people with such disabilities is critical. Conventional walkers and canes have played a significant role in assisting mobility-impaired humans, without unfortunately eliminating falling incidents, nor being easily adaptable to every patient's specific needs [10, 11]. However, their developmental potential under robotics and computer vision advancements is massive [12]. A smart, context-aware device can help every individual retain the normalcy in their everyday life and regain their independence and self-esteem [13, 14, 15, 16].

An **intelligent robotic mobility assistant** collects information from their environment, processes them and uses them, in order to help the user carry out everyday activities, or formally the Activities of Daily Living (ADLs), such as washing, toileting, dressing, feeding, transferring etc. As the type of assistance an individual needs depends a lot on the type of pathogenies they present, these assistants can learn these pathogenies while interacting with the user and adapt their behavior so to suit their specific characteristics and needs.

Among all different functionalities that can make a robotic assistant intelligent is a leg tracking and analysis mechanism. Leg tracking refers to the accurate estimation of human legs' position throughout time [17], while gait analysis refers to extracting information

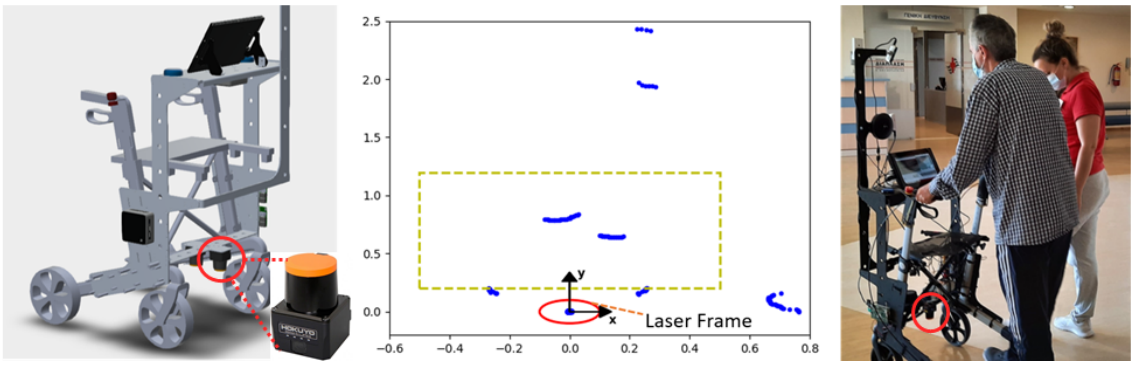


Figure 1.1: **Right:** A patient using an intelligent robotic walker. **Left:** Sketch of the walker design. A Hokuyo laser scans the walking area at a height $\sim$ 35cm above the ground. **Middle:** 2D range data (blue) from the laser scanner, whose origin is marked by a red circle. Due to the existence of obstacles other than the patient's legs, only the laser points lying in the bounding box (yellow) are considered for leg detection and gait analysis.

about the way a human walks. Insight into the position of a person's legs and their type of gait can suggest much about them, such as disabilities [18, 19], an inclination to fall [20], etc. Therefore, an intelligent robotic walker with efficient tracking and gait analysis system can also be employed for more sophisticated functionalities that correspond to the user's needs. However, the proximity of the user to the robot during the supportive actions of walking (Fig. 1.1), along with the pathological aspects that alter the gait patterns and walking frequencies, make the leg tracking very challenging, while the gait analysis task from partial 2D observations is increasingly demanding: in close proximity depth cameras are performing poorly, fast 3D range-scanners are prohibitively expensive and elderly patients are skeptical to wearable sensors. Notably, only a few works in the literature consider gait tracking using range data for robotic walkers [17, 1]. This work's motivation is to equip such a walker with an effective leg tracking and gait analysis mechanism in order to assess the individual's needs and adapt to assist as optimally as possible. A context-aware robotic walker could improve on patient-supporting, guiding, fall prevention, or even rehabilitation [21, 22].

1.2 Related Work

1.2.1 Robotic mobility assistants

The idea of equipping a mobility assistant with intelligent features already exists in numerous works in literature. Especially in the direction of smart walkers, many intelligent robotic assistants have been developed aiming to evolve conventional walkers, which can present numerous complications, such as difficulties in manoeuvring them through congested areas and around obstacles, and lack of stability and break control [23].
The PAM-AID [24] is a smart walker intended for people with visual or mobility impairments. This walker can control the orientation of its front wheel based on a user interface

on the handlebar, while also detecting obstacles via ultrasonic and laser sensors and providing voice messages to inform the user. ASBGo [25] uses information provided by sonar and force sensors in order to assist ataxic people by recognizing their intent and predict falling incidents. JARoW [26] uses a Kalman filter on laser sensor data to estimate and predict the locations of the user's legs and body in real time and adjust the walker's motions corresponding to the user's walking behavior. SYMBIOSIS [27] is equipped with force and ultrasonic sensors and extracts information about the user's gait and the human-walker interaction during gait.

MOBOT [28] is a robotic rollator equipped with LRF sensors used for environment mapping and obstacle detection, as well as user gait analysis, force/torque handle sensors, two Kinect sensors that record the user's movements and an array of microphones for audio capturing. The goal of this project is to provide adaptive user support and fall prevention exploiting multimodal information. i-Walk [10] is a current project on a smart walker aiming to utilize multi-sensory signals to monitor, analyse and predict human gait, while providing adaptive and autonomous navigation through human-robot vocal communication.

1.2.2 Leg Tracking Algorithms

Human leg tracking has been a popular subject on robotic applications, mostly for human detection, tracking, and following. Data derived from various sensors, including lasers, cameras, markers, etc., are used and often fused to estimate the position of human legs in sequential time frames. A fusion of RGB-marker and IMU data has been proposed for leg detection, combined with an extended Kalman filter for leg tracking [29]. Biometric data have also been used for human detection [30].

As most mobile robots are equipped with a 2D Laser Imaging Detection and Ranging (LIDAR) sensor, due to their reliability and affordability, there has been plenty of proposed work on learning algorithms for processing the laser sensor data for human detection and tracking.

Many of those methods, either used a leg pattern recognition scheme [31][32], boosted classifiers [33] or clustering methods [34][35] and Kalman filters for the tracking part. Leg tracking and gait analysis using two particle filters, one per leg, and probabilistic data association with an interactive multiple model scheme have also been proposed by [1]. Such implementations, however, have a high computational cost, and thus the high frequency of modern laser scanners cannot be fully exploited.

On the other hand, deep learning methods have also been considered in human tracking due to their scalability and fast inference. The use of CNN is presented in [36] for detecting people in crowds from range data. A U-Net architecture [37], commonly used for biomedical image segmentation, has also been proposed in [38]. These methods, though, perform person detection and do not consider learning the human legs' dynamic motion that can be exploited for tracking, which is crucial for further gait analysis.

1.2.3 Gait Analysis Mechanisms

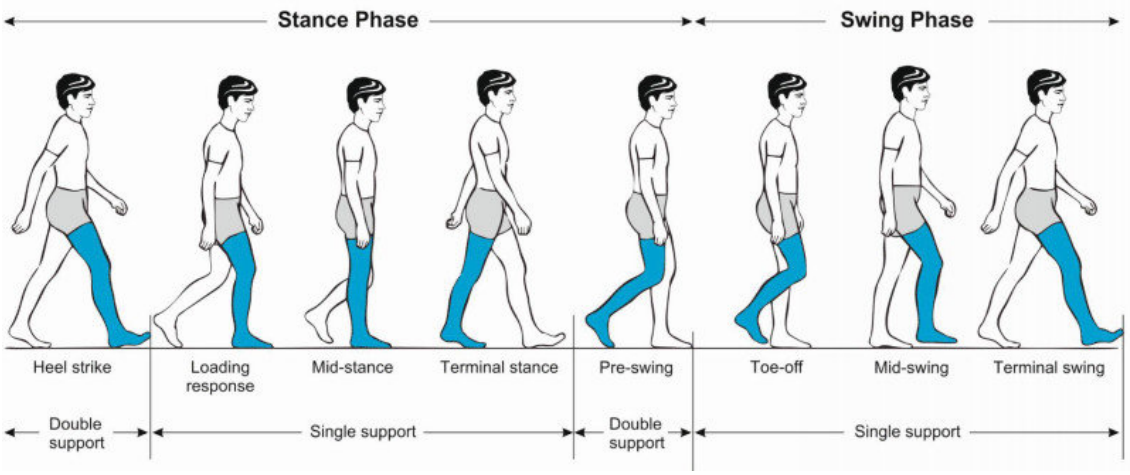


Figure 1.2: Fig. from [39] representing the phases of a single gait cycle, based on [7].

Human gait analysis is a field of research receiving increasing interest over the last decade and focuses in extracting the parameters involved in human walking. Gait analysis can be applied to various fields, such as sports, medicine, as well as human identification [40]. Especially in medicine, gait analysis can play a crucial role in monitoring patient health and deducing conclusions regarding their current health state.

The systems used for gait analysis can be categorized into two subgroups: (1) the wearable sensors, such as pressure and force sensors, inertial sensors that measure velocity, acceleration, orientation, etc. and (2) the non-wearable sensors. For smart walkers, which need to be appealing to the user and easy to use, it would be preferable to not involve any wearable sensors, but rather integrate all smart features in a normal rollator usage. For this reason, we focus on this section on the non-wearable sensors.

Non-wearable sensors for gait analysis can be divided into two categories: (1) floor sensors, including force platforms and pressure measurement systems, and (2) image processing methods. For smart walkers, only image processing methods can be applied. Such methods process data deriving from various sensors, such as cameras, laser range finder (LRF) sensors, Kinect or infrared cameras. Most of the developed smart walkers are equipped with an LRF sensor due to their simple scanning area representation and their affordability. The most recent and relevant work in literature [1] uses data from an LRF sensor and Hidden Markov Models (HMM) in order to estimate the patients gait state at each time frame. This method though has a complexity toll when its accuracy is high and is difficult to be implemented in an online system.

Scientifically, human gait is perceived as shown in Fig. 1.2, where a single gait cycle consists of 8 sub-phases: (1) heel strike or initial contact, (2) loading response, (3) mid-stance, (4) terminal stance, (5) pre-swing, (6) toe-off, (7) mid-swing and (8) terminal swing. This representation is of high complexity, as it categorizes the position of the legs' heels and toes across time. Such a representation seems impossible to be perceived from a simple

LRF sensor. For this reason, a simpler but still effective gait model is used in this thesis, where only four gait phases exist: (a) the left double support, which includes the explicit gait phase (1), (b) the left swing/right stance representing phases (2) - (4), (c) the right double support, representing phase (5) and (d) the right stance/left support, representing phases (6) - (8).

1.2.4 Deep Learning Detection

Deep neural networks have achieved exceptional results in object detection in general. The widely popular YOLO architecture [41] can detect multiple objects and suggest a bounding box for every one of them.

For this architecture to be used in our work, a specific and extensive dataset including bounding boxes would have to be created, which should not be mandatory for leg tracking. Also, this problem can not be solved by a single frame object detection neural network, as occlusions can temporarily make a leg invisible. Due to the necessity for an assistive walker to have constant awareness of the patient's leg position, a neural network architecture with temporal awareness is called for. For this purpose, two ways of extending a CNN were considered.

The first one was the use of a Long Short Term Memory (LSTM). In [42], for the exploitation of the temporal information that a video provides, a neural network with a Single Shot Multibox Detector (SSD), followed by two LSTM layers for object detection on video streams, was presented. Similarly, in [43], an LSTM layer was used after a YOLO for improving detection performance.

The second one was the use of a Temporal Convolutional Network (TCN). As TCNs have shown promising results and often a better performance than LSTMs, a Dilated TCN layer, similar to the one proposed in [44] for motion capture, was added after the CNN. This TCN uses exponentially increasing dilation between consecutive 1D convolutional layers for associating the features between successive data.

1.3 Contribution

This thesis presents a novel deep learning framework for Leg Tracking by detection and Gait Analysis from 2D range data (LTGADnet). The proposed approach uses a CNN for leg detection, followed by an LSTM network for exploiting temporal information in walking and revising the CNN's detections in challenging situations like leg occlusions. Finally, a second LSTM is used to extract the high-level temporal interaction of the legs, which can provide evidence about the occurring gait phase, resulting in a real-time gait analysis system. Our key contribution is a highly accurate leg tracking by detection method, thanks to the deep feature extraction, and an implementation that can be deployed as an off-the-shelf leg tracking method for any robotic mobility assistant, due to its effectiveness and high-frequency, without the need for extra calibrations or thresholds. Experimental evidence

Table 1.1: Gait State representation as in [1] and their one-hot key encoding.

State Code	One-Hot Key	Code Name	Definition
s_1	[1, 0, 0, 0]	LDS	Left Double Support
s_2	[0, 1, 0, 0]	LS/RW	Left Swing/Right Stance
s_3	[0, 0, 1, 0]	RDS	Right Double Support
s_4	[0, 0, 0, 1]	RS/LW	Right Swing/Left Stance

are provided for this deep learning solution using the Leg Tracking and Gait Analysis from 2D range data (LTGAD) database, which was created explicitly for the training, validation and testing of the LTGADnet and comprises data and annotations from real patients using mobility devices. The produced results are compared with the most recent algorithm that was developed for such applications [1], showcasing the performance increase of this work's method.

1.4 Problem Statement

The problem that this thesis aims to solve is to perform accurate, efficient, and robust leg tracking and gait analysis on a patient using a smart walker equipped with a 2D LIDAR. Given readings of the laser sensor at each time frame t, i.e., distances from the laser to detected objects, the goal is twofold: (1) to find the relative coordinates of the centers $(x_r[t], y_r[t])$ and $(x_l[t], y_l[t])$ of the patient's right and left leg with respect to the laser's position and (2) given these centers, to estimate the patient's gate state. These states represent specific phases of walking, as defined in gait analysis literature. The gait states we are considering in this work are shown in Table 1.1. These gait states are a part of a Markov chain 1.3, where every state at every time frame can either stay the same or transition to the next one in the chain. In order to understand what these states actually represent for a human walking, we can imagine a human that is in stance with both feet on the ground (DOUBLE SUPPORT) and starts walking. If he lifts the left leg first, then he was in LEFT DOUBLE SUPPORT before and is now in the LEFT SWING / RIGHT STANCE state. Then, he will stay for some milliseconds in this state and transition to the only possible to follow state, which is the RIGHT DOUBLE SUPPORT, as he lands his left leg, in order to lift at some point his right leg, find himself in the RIGHT SWING / LEFT STANCE state and continue hence the walking movement.

1.5 Organization of the thesis

In chapter 2 we explain some important topics of the neural network theory, necessary for achieving a full understanding of our work. In chapter 3 we present the database used for the training, validation and testing of the network. We describe in detail the data morphology and the database structure, the data extraction and creation process, as well as

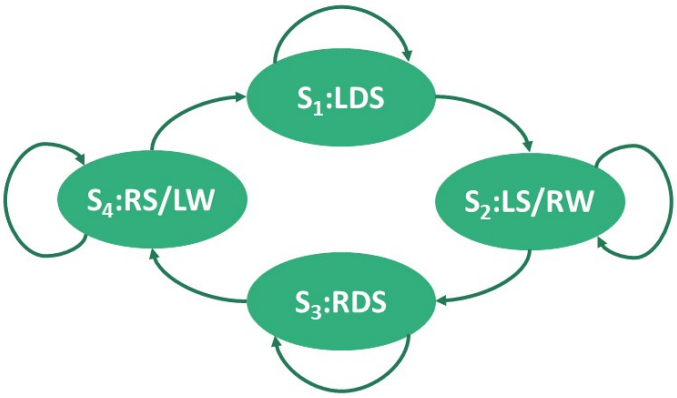


Figure 1.3: The possible gait states with all the possible transitions between them.

the method used for organizing the data and finally using them on the network. In chapter 4 we present the developed neural network, its components, the architectural decisions made on the network, but also the ones considered for increasing the network's efficiency and accuracy. In chapter 5 we show the performance of the network. We report an ablation study, showing the accuracy of all network architectures that were tested. A leave-one-out validation study is also exhibited, in order to prove the network's ability to learn the desideratum. We include diagrams indicating the ability of the network to successfully conduct the leg tracking and gait analysis. Moreover, we compare our framework to the state-of-the-art method for leg tracking and gait analysis. In chapter 6 we elaborate on the future work that could be performed to enhance the current thesis.

Chapter 2

Neural Network Preliminaries

In this section, some basic neural network concepts important for the full comprehension of this thesis's work are presented. We describe in detail some of the neural network architectures considered for the LTGADnet and explain the reasons for choosing these in the final network.

2.1 Artificial Neural Networks

In general, an artificial neural network is a system that uses a set of nodes, the **neurons**, and connections between them, the **synapses**, in order to learn how to solve an artificial intelligence problem. The neurons could belong to the **input layer**, which is the one receiving the network's input, the **hidden layers**, which are optional and contribute in the further processing of the input, or the **output layer**, which represents the final output of the network. The signals received from the input layer neurons are forwarded through the synapses first to the neurons of the hidden layers and finally to the output layer neurons. A layer where all its neurons are connected to all neurons of the previous layer is called **fully connected**.
Each of the neural network neurons that receives signals can process them and then transmit the produced signal to the neurons connected to it by synapses. The processing involves a linear combination of the received signals, with **weights** deriving from the synapses, an optional added **bias** and the final pass into the neuron's **activation function**.

2.1.1 Dataset

For the neural network training, a dataset of **labeled data** needs to be created. The labels of the data are the desired output of the neural network, if these data were given as input. These labels will further be used in calculating the **loss**, as described in 2.1.3. This dataset is split into three parts: the **training set**, the **validation set** and the **test set**. Each set plays a specific role in the neural network's design process:

- The training set is used for the training of the neural network.

- The validation set is used for an impartial examination of the network's performance during training and the tweaking of the model's **hyperparameters**, such as the learning rate, the number of neurons and layers, the number of epochs etc.
- The test set is used for the ultimate testing of the neural network, when no further changes to the network's architecture and hyperparameters are intended. The performance on the test set is the only one that can determine the network's true efficiency.

2.1.2 Activation Function

Every neuron in a neural network has an activation function, from which the neuron's input is passed and the result becomes the neuron's output. When this function is nonlinear, it allows the network to compute nontrivial problems using only a small number of nodes, and is then called a **nonlinearity**. Popular activation functions are the following:

- Linear: $f(x) = x$
- ReLU: $max(0, x)$
- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Tanh: $\tanh x$

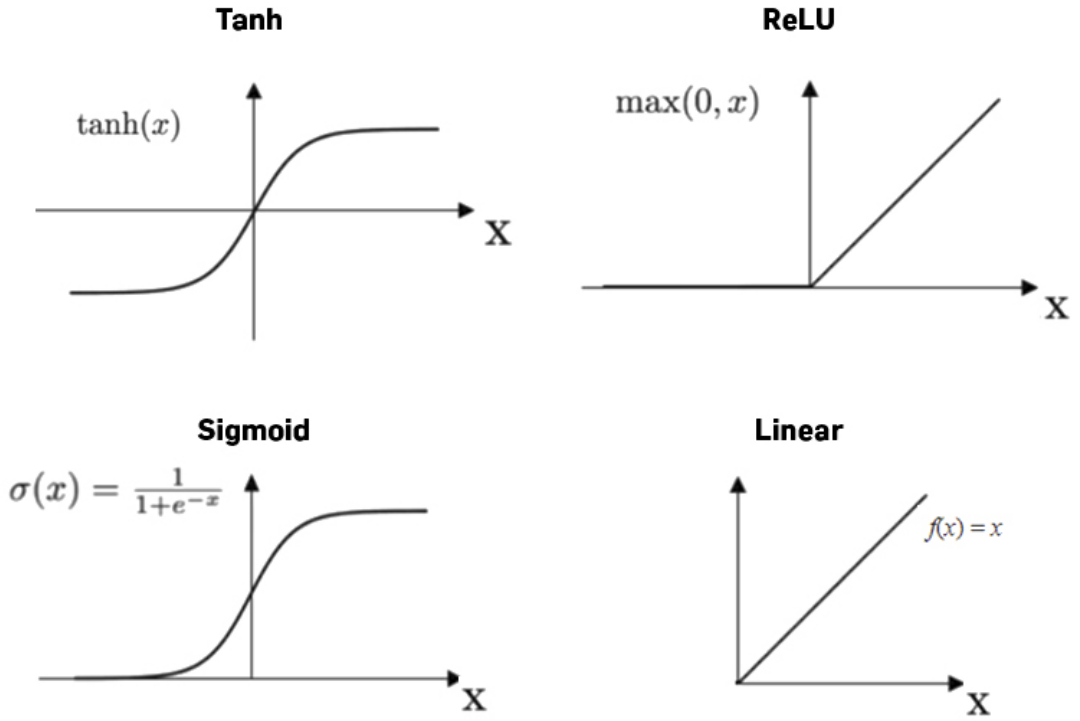


Figure 2.1: Common activation functions and their plots.

2.1.3 Loss Function

The **loss function** of the neural network is a function that, given the network's output, produces a real number, which represents some "loss". In the neural network's training, this loss represents how close the output is to the desired result. We treat the training as an optimization problem that seeks to minimize the loss function's result.

2.1.4 Backpropagation

The neural network's training is achieved through an algorithm called **backpropagation**. In this algorithm, the **gradient** in weight space of a neural network is computed, with respect to the loss function. Specifically, it creates an expression for the partial derivative $\partial C/\partial w$ of the cost function C with respect to any weight w (or bias b) in the network. This expression tells us how quickly the loss produced by the loss function changes when we change the weights and biases.

2.1.5 Optimizer

An **optimizer** is a method used during the neural network training to change the network's attributes, such as weights and learning rates, in order to reduce the loss. Usually, optimizers analyze the gradients produced by the backpropagation and decide how to tweak the network's parameters, based on these gradients. The most popular optimizers used are Stochastic Gradient Descent (SGD) and Adam.
The **learning rate** is a hyperparameter that controls how big a change to apply upon the model in response to the estimated loss, at each step that the model weights are updated. The choice of the right learning rate is significant for achieving the best possible result during the training. When the learning rate is too high, gradient descent can inadvertently increase rather than decrease the training error. On the other hand, as most loss functions are not convex, too low a learning rate can trap the parameters to a local minimum and lead to a permanently high training error.

2.1.6 Training

Usually, we begin the neural network training by initializing all weights and biases randomly (though it has been shown that determining them explicitly may improve results [45]). Then, the training set is passed through the network. This pass, namely the **forward pass**, can be made with one input at a time or a batch of inputs. The number of inputs that the batch contains is named the **batch size**. After every forward pass, a **backward pass** follows, which applies the backpropagation to calculate all partial derivatives. Then, the optimizer uses these to update the current weights and biases, a process called a **step** of the optimizer. The backward pass is generally the most time consuming process in the network's training. Thus, using a batch of inputs for the forward pass can help achieve

faster training, as backpropagation in batch training is only applied every few inputs, as opposed to single input training.

This process is realized a number of times called **epochs**. Each epoch is a single pass of the whole training set through the network. The number of epochs is another hyperparameter that needs to be optimized. A training for too many epochs can lead to the **overfitting** of the network, meaning that the network does not generalize well the problem, but rather **memorizes** the desired output for the training set. Training the network for too few epochs may cause the network to **underfit**, meaning that the network has not learned yet the desired task. A way to determine the number of training epochs is **early-stopping**. The idea is to stop the training process when the loss in the training set keeps decreasing, but the loss in the validation set does not. This is a sign for overfitting, as the neural network keeps learning, but unfortunately does not generalize.

Except from defining the number of epochs, another way of preventing overfitting is the use of a **dropout layer**. With a dropout layer some of the neurons from a layer's output are ignored, meaning that they are temporarily removed from the network, along with all the synapses attached to them. Batch training has also been shown to help avoid overfitting.

2.2 Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a class of neural networks mostly for image analysis. It consists of an input layer, hidden layers and an output layer, where the hidden layers perform **convolutions**. The common activation function for these layers is the ReLU (2.1.2). More layers of a CNN could be **pooling layers**, **fully connected layers** and **normalization layers**.

2.2.1 Convolutional Layers

A convolutional layer takes its input from the previous layer, convolves it with a number of convolutional **filters** of specific width and height and passes it to the next layer. Each convolution with a filter produces a different output channel. The advantage of these layers, as opposed to the fully connected layers, is the practicality in large input data, such as images, due to the fact that a fully connected layer would need to have the same size as the image, requiring increased complexity. Convolutional layers may have additional hyperparameters except from the kernel width and height, such as the convolution **padding**, **stride** and **dilation**.

2.2.2 Convolution

The operation of convolution between a filter and an image is the linear combination of the image's cells, with weights determined by the filter. An example of convolution with a 3×3 filter is shown in Fig. 2.2.

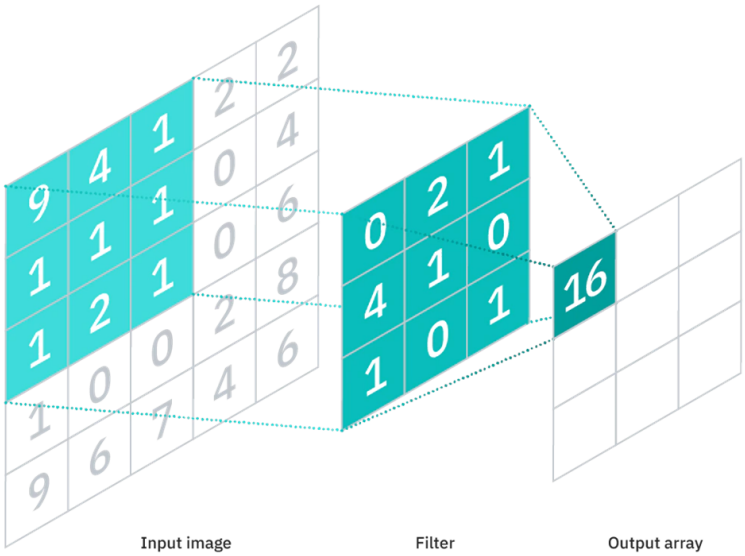


Figure 2.2: Example of convolution of an image by a 3×3 filter. The algebraic representation of this convolution is $9 \cdot 0 + 4 \cdot 2 + 1 \cdot 1 + 1 \cdot 4 + 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + 2 \cdot 0 + 1 \cdot 1 = 16$

We can see that, with this convolution some image pixels will not be multiplied by some filter values. In this way, the convolution is not applied equally to every image cell. When this is unwanted, padding of any value, mostly zero, is used (Fig. 2.3).
Also, a filter can move a number of image pixels before it is applied. This number is called the stride (Fig. 2.4). As the stride increases, the forward pass takes less time to finish, but also less image information is processed.
It is understood that, if the image is of size $n \times n$ and the kernel, padding and stride have size $f \times f$, p and s respectively, then the output of the convolutional layer will be of size:

$$\frac{n + 2p - f}{s} + 1$$

Another convolution parameter, commonly used in Temporal Convolutional Networks

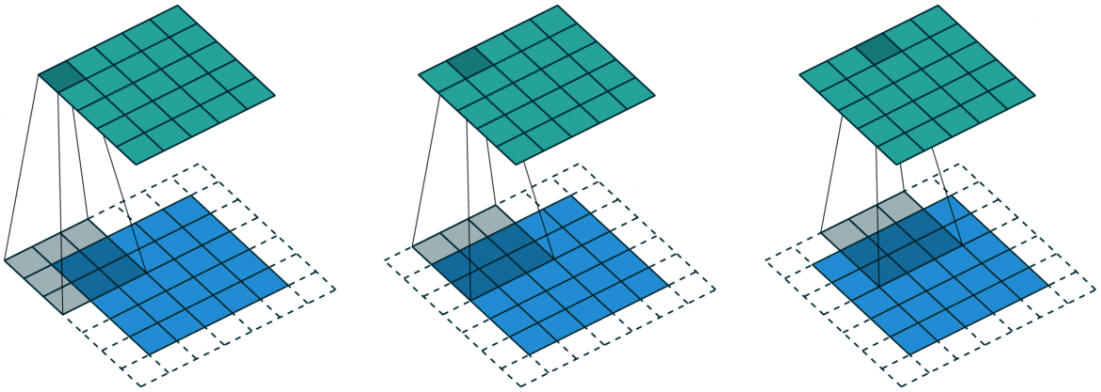


Figure 2.3: Example of convolution with kernel size 3 and padding 1

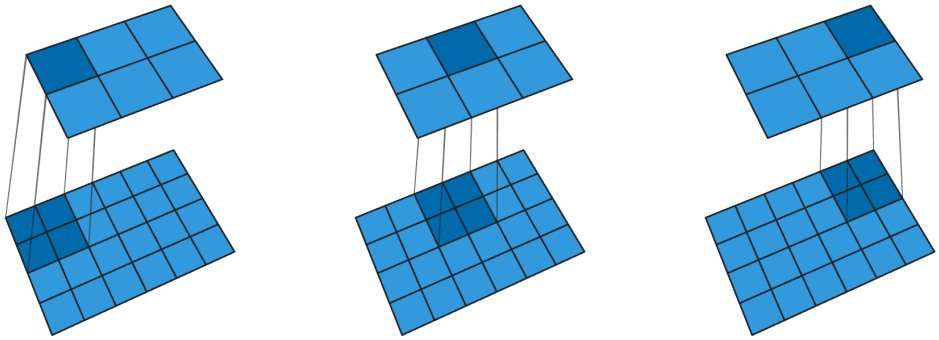


Figure 2.4: Example of convolution with kernel size 2 and stride 2

(TCN) as described in section 2.4, is the dilation. When a convolution includes dilation $d > 1$, it means that the filter will be convolved with a grid from the image created by skipping d pixels at all directions. In Fig. 2.5 we can see an example of a dilated convolution.

2.2.3 Pooling Layers

Convolutional networks may include pooling layers to reduce the data dimensions. They combine the outputs of neuron clusters at one layer into a single neuron in the next layer. Two are the popular types of pooling: max and average pooling. **Max pooling** returns the maximum value of each cluster of neurons in the feature map, while **average pooling** returns the average value.

2.2.4 Normalization Layers

In layer normalization, inputs belonging to a minibatch are transformed, so that elements in a single input have zero mean and unit variance. After this transformation, there is also a scaling and shifting step by learnable parameters. In convolutional layers, normalization is not applied to every input, but rather to every channel of an input. A common normalization method is **batch normalization** [46], where the normalization is done so that every element in a batch has zero mean and unit variance. With this method, the normalization of every input depends on the other inputs in the same batch.

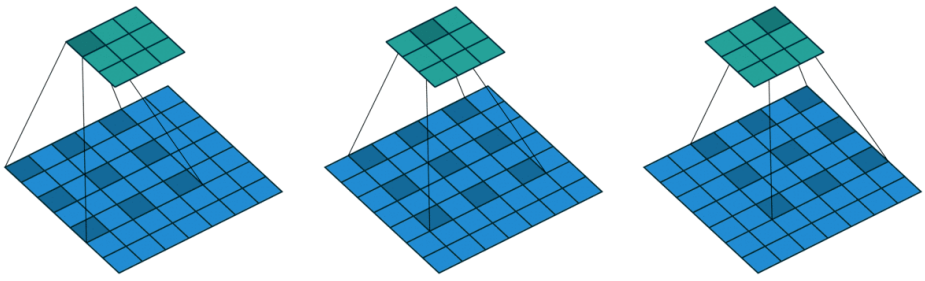


Figure 2.5: Example of convolution with kernel size 3 and dilation 2

2.2.5 CNN Example: You Only Look Once (YOLO)

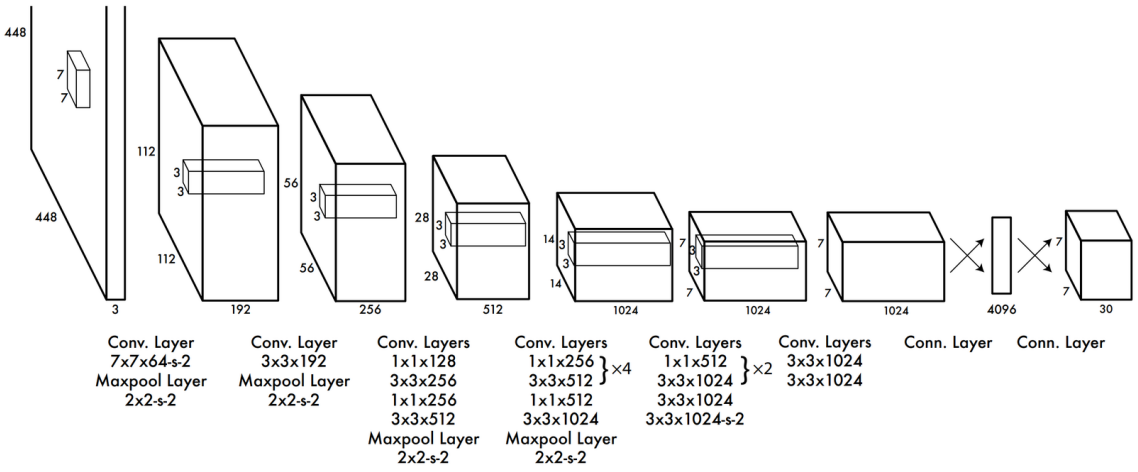


Figure 2.6: Architecture of the YOLO neural network

The YOLO neural network [41] performs detection of multiple objects of multiple classes in an image. Object detection refers to the estimation of both the object class and the object position in an image. This network consists of convolutional, maxpooling and fully connected layers. For every object in the input image, YOLO predicts a class and a bounding box with its center, width and height.

It divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, this cell is responsible for detecting the object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate the box is. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The x, y correspond to the coordinates of the center of the bounding box and w, h to its width and height, relative to the whole image.

For the YOLO training, the loss function is the following:

$$
\begin{aligned}
Loss_{yolo} = \lambda_{coord} & \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] && (2.1) \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(\sqrt{C_i} - \sqrt{\hat{C}_i} \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left[\left(\sqrt{C_i} - \sqrt{\hat{C}_i} \right)^2 \right] && (2.2) \\
& + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{noobj} \sum_{C \in classes} \left(\sqrt{p_i(C)} - \sqrt{\hat{p}_i(C)} \right)^2 && (2.3)
\end{aligned}
$$

where Eq. 2.1 is the bounding box coordinates error, Eq. 2.2 is the confidence error and Eq. 2.3 is the class estimation error. The $\mathbb{1}_{i}^{obj}$ denotes if object appears in cell i, $\mathbb{1}_{ij}^{obj}$

denotes that the jth bounding box predictor in cell i is responsible for that prediction and $\mathbb{1}_{ij}^{noobj}$ denotes the opposite of the latter. The λ_{coord}, λ_{noobj} parameters are used to stabilize the model, as the loss of the cells not containing any object tends to overpower the gradient of the cells that do.
As the YOLO network can be trained to detect objects of any class, it seems possible to use it for leg detection too. Like all deep neural networks, it would require an extensive dataset, that would need to contain not only the legs' centers, but also their bounding boxes. This is though a very time-consuming task, that could probably be avoided, as leg detection itself does not require bounding box predictions. That said, the idea was to take inspiration from the YOLO network and simplify it as much as possible.

2.3 Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) is a class of neural networks where connections between neurons form a strongly connected graph, meaning that all nodes in the graph can be reached from all other nodes. This means that the RNN can retain a form of **memory** and use it in order to extract temporal information about the input. The input of an RNN is usually an arbitrarily long **sequence**. RNNs are mostly used in speech and handwriting recognition, but can be found useful in any application involving temporal relations between inputs. Common types of RNNs are the Long Short-Term Memory (LSTM) [2] and the Gated Recurrent Unit (GRU) [3], which is a simplified variant of the LSTM.
A problem that RNNs face is that, while training with gradient descent, error gradients vanish exponentially quickly with the size of the time lag between important events. LSTMs were developed to deal with the **vanishing gradient problem** and their architecture will be explained in the next section.

2.3.1 RNN Example: Long Short-Term Memory (LSTM)

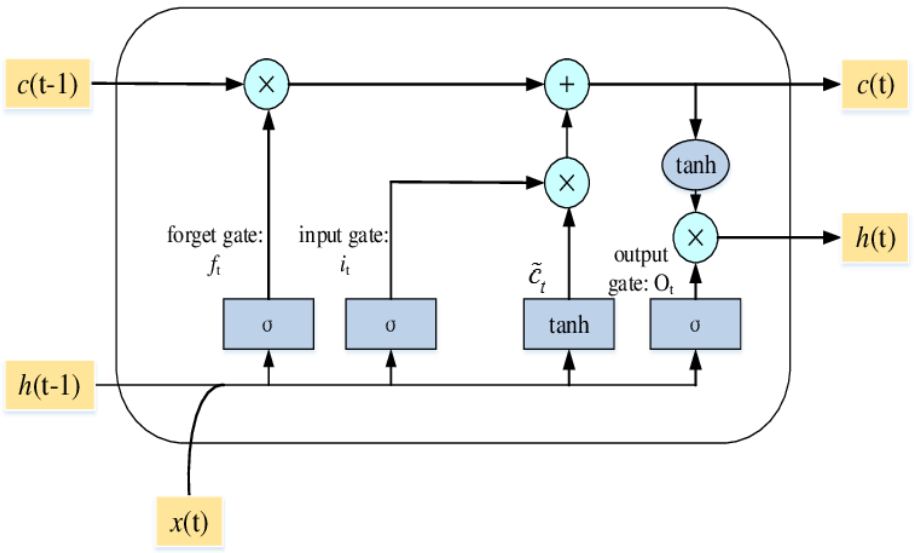


Figure 2.7: Architecture of the LSTM unit

The Long Short-Term Memory (LSTM) is a type of RNN and consists of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. A network can be composed of many consequent LSTM units, forming a multilayered LSTM. For every element in the input sequence, each layer computes the following function:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if}W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, h_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively, σ is the **sigmoid function**, and $\odot$ is the **Hadamard product**. The hidden and cell state during the RNN's training need to be reinitialized every time that a new sequence of data is feeded to the network.
LSTMs can also be **bidirectional**, meaning that the output of an input belonging to a sequence depends on both previous and future inputs. It is obvious that a bidirectional LSTM cannot be used in online applications, where future inputs are unknown.
So, in order to still be able in our task to track human legs in cases of occlusions, we used an LSTM after a CNN to memorize previous leg positions and correct mistakes that the CNN could make when a leg is invisible.

2.4 Temporal Convolutional Networks (TCN)

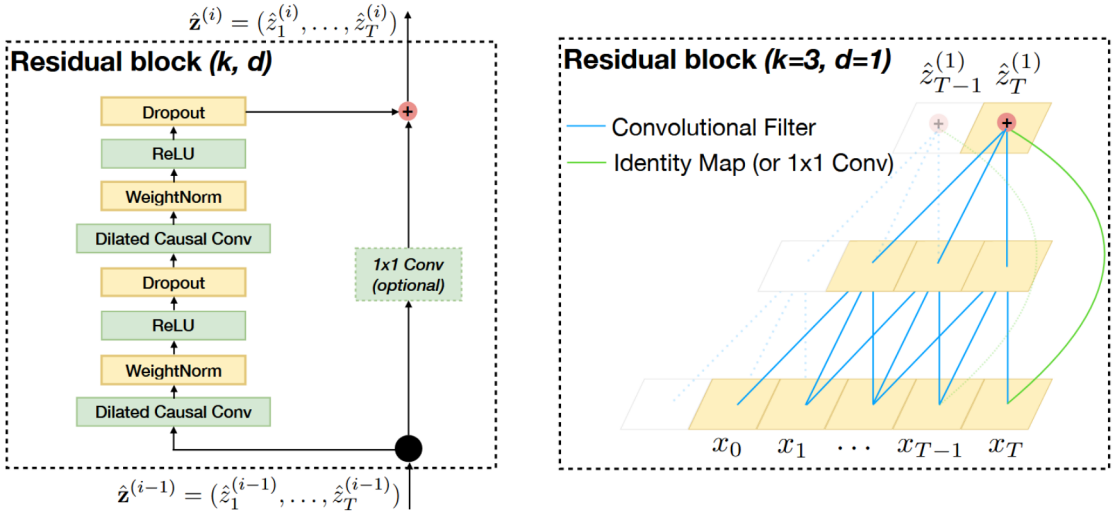


Figure 2.8: Architecture of the TCN residual block, as in paper [44].

Temporal Convolutional Networks (TCN) were developed as an alternative to RNNs, in order to decrease training time and increase memory retention. They consist of dilated causal 1D convolutional layers with the same input and output size. The input of a TCN is a 3D vector of size (B, L, S_{in}), where B the batch size, L the sequence length and S_{in} the input sequence element length, and produces as output a 3D vector of size (B, L, S_{out}). The dilated convolution operates as described in section 2.2.2, where each hidden layer has exponentially increased dilation to the layer preceding. Zero-padding is also applied, to ensure that every layer input and output sequence are of the same length. Each TCN layer is in fact a residual block, as in Fig. 2.8. TCNs have been proven to be really effective for video action segmentation [4] and other sequence modeling tasks [5], so we attempted to replace the leg tracking LSTM of our network with this architecture.

Chapter 3

The Leg Tracking and Gait Analysis Database

In this section we present the LTGAD database, a database created and used for the LTGADnet training, validation and testing. The database is publicly available and can be accessed online[1]. It consists of labeled data extracted from previous work [1], which were collected and cleaned, in order to be usable in our work. The real data extraction will be further explained in section 3.2. It also consists of computer generated data, created to "imitate" the real dataset and enrich the existing dataset, as an object detection network needs a large set of data for training.

3.1 Database Structure

3.1.1 General Structure

The LTGAD database is separated into two sub databases: the Leg Tracking database and the Gait Analysis database. Each one comprises of data from 8 different experiments, as explained in section 3.1.2, along with their annotations. Overall, these experiments consist of approximately 33000 frames. Additionally to the real experiments exists one computer generated experiment, used only for training the neural network and with such size that the final percentage of the computer generated frames to the real ones is 58%.

3.1.2 Experiment Structure

Each experiment, whether it is a real or an artificial one, consists of three files:

1. *laserpoints.csv*, which includes per frame (rows) the position of the obstacles provided by the laser sensor (columns). The position is relative to the position of the laser sensor, which is assumed to lie on the origin, and is represented as x, y coordinates.

[1] https://robotics.ntua.gr/ltgad/

So, every row has columns $x_1, y_1, x_2, y_2, ..., x_n, y_n$, where n the number of angles that the laser scans.

2. *centers.csv*, which exists only in the experiments of the Leg Tracking database and includes per frame (rows) the actual position of the center of both legs (columns). Similarly to laserpoints.csv, each position is relative to the position of the laser sensor, which is assumed to lie on the origin, and is represented as x, y coordinates. So, every row has 4 columns x_r, y_r, x_l, y_l, which represent the coordinates of the centers of the right and left leg respectively.

3. *gait_ states.csv*, which exists only in the experiments of the Gait Analysis database and includes per frame (rows) the annotated gait state. As explained in Table 1.1, the possible gait states are represented with a number between $1-4$ and correspond to LEFT DOUBLE SUPPORT, LEFT SWING / RIGHT STANCE, RIGHT DOUBLE SUPPORT and RIGHT SWING / LEFT STANCE.

4. *valid.txt*, which includes the frames that where selected from the initial data. Some data frames included not only the patient's legs, but also a nurse's or parts of other objects, which are not a usage case for our task. For this reason, the initial laserpoints where cleaned where possible, by removing by hand only the laserpoints that did not correspond to a patient's legs. But when this was not possible, due to their complete invisibility, these frames where omitted. We did not just delete these frames for two reasons: 1) these frames might not be suitable for our network training, but could be found useful by someone else for another task and 2) by creating the *valid.txt* file we could know when some frames are omitted, thus creating a discontinuity to the frame sequence, making it necessary for a reinitialization of the RNN's memory (section 2.3). So, in *valid.txt* every row has the starting and ending frame, both inclusive, of a chosen sequence.

3.1.3 Data Handler

The data handler is the package used for loading the data needed for the neural network and splitting them into the training, validation and test set. For the LTGADnet we needed to design explicitly the data handler, as the dataset has some peculiarities, with one being the *valid.txt* files, which do not allow a simple usage of a common data handler. So, this data handler needed to perform the following tasks:

- read the experiment data files while taking into consideration the *valid.txt* file. We needed to only keep the chosen data frames and save them as sequences.

- create the bounding box, which is the $1m \times 1m$ area assumed to contain the patient's legs, as shown in Fig. 1.1. Only the laser points lying in this bounding box are kept and transformed into the input image.

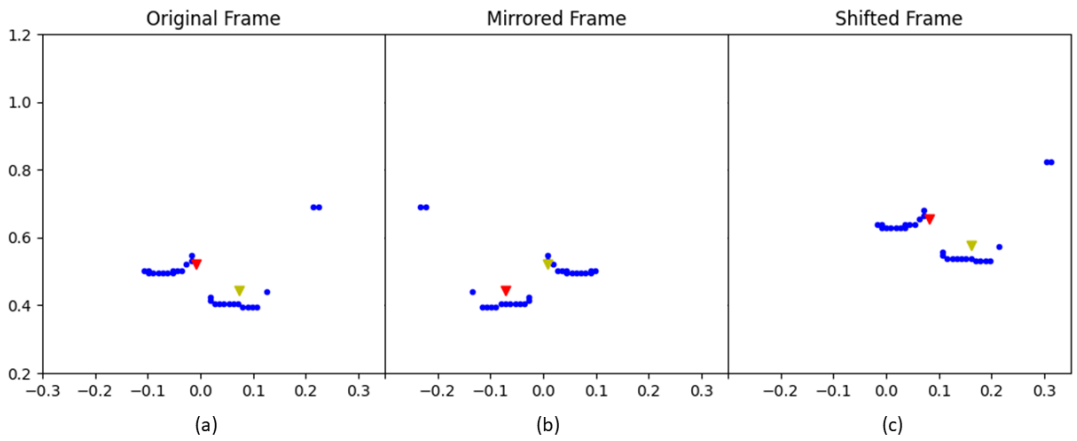


Figure 3.1: Three frames of the dataset, where (a) is the original frame and (b), (c) are the frames resulting after the data augmentation. With blue points are annotated the points detected by the laser (laserpoints), while with red and yellow triangles the annotated leg centers of the right and left leg respectively. In this figure, (b) is the original frame mirrored and (c) is the original frame shifted, with one of the 6 shifts that were performed.

- prepare one batch of data at a time. Both laser and annotated data are transformed, in order to be given as input to the network, as explained in section 4.1.1. The data are being loaded in the RAM one batch at a time for memory saving reasons, at the cost of a slower training.
- apply data augmentation. The experimental data are augmented, in order to increase the size of the dataset. This augmentation includes mirroring and extensive shifting of the data, resulting to a new dataset of size 8 times greater than the previous one. An example of the data augmentation is presented in Fig. 3.1. The shifts were carefully picked so that no information of the legs gets past the bounding box.

3.2 Experimental Setup

The experiments included in our dataset are of real patients and were conducted for the work of [1], where MOBOT [28] was used for the rollator. The participants to the experiments where over 65 years old and presented moderate to mild mobility impairment, according to clinical evaluation. The patients were wearing their normal clothes. They were asked to realize a walking scenario having physical support of a rollator and some turning manoeuvres to avoid obstacles. All patients performed the experimental scenarios under appropriate carer's supervision.
The rollator was equipped with a Hokuyo rapid laser sensor UBG-04LX-F01, with mean sampling period of about 28msec/scan, scanning range of 20 to 5600mm, angle range -120^o to 120^o and angular resolution 0.36^o, placed at a height of about 40cm from the ground, in order to capture the motion of the subject's tibia.

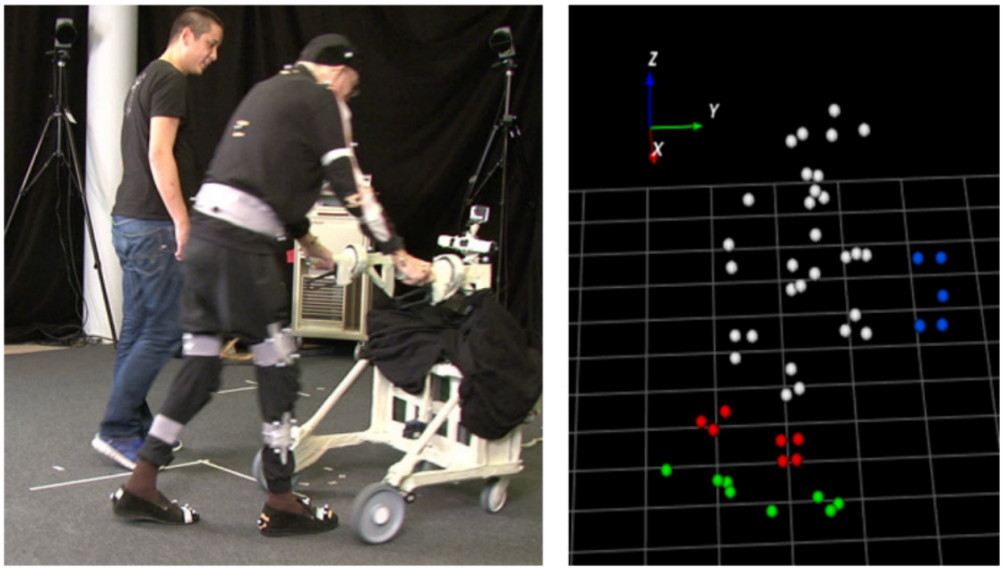


Figure 3.2: Figure from [1] showing a snapshot from the experimentation scene and a representation of the visual markers from MOKKA visualization system.

For the gait state extraction, a set of markers from a VICON Motion Capture system was placed on the heels and toes of the patients' body, as described in [6]. In this work, an automatic gait phases detection system was developed, based on the works of [47, 48]. This system uses 1 Heel marker and 3 Toe markers for each foot and considers the impact of those markers only on the z (vertical) axis. After linear interpolation, resampling and filtering with a lowpass Butterworth filter, the heel trajectory is extracted, as well as the toe trajectory by calculating the median of the three toe markers.
With these trajectories, as well as the calculated tow velocity in the vertical direction, certain gait states can now be detected, specifically the heel strike, loading response, terminal stance and toe off, as they have already been explained in section 1.2.3.

3.3 Computer Generated Data

For the training of both the convolutional and recurrent neural networks, a fairly large, but also highly accurate dataset of leg centers and gait states was necessary. As the experimental dataset was retrieved from real patients, there were many difficulties in extracting with high accuracy their leg centers and gait states. This can be seen in Fig. 3.1, where one leg's center in the original frame lacks in accuracy (the right one in this figure), and which is the case in many other frames. Also, the existing real dataset was of around 33000 frames, which is acceptable for the leg detection training, but seems insufficient for the gait analysis, not so much because of the number of frames, but rather the number of different patients realizing the scenarios. A variety of patients can lead to a more accurate gait analysis, as the transitions between the gait states can appear to follow certain patterns in an individual's walk. So, we would need the neural network to learn how to detect these

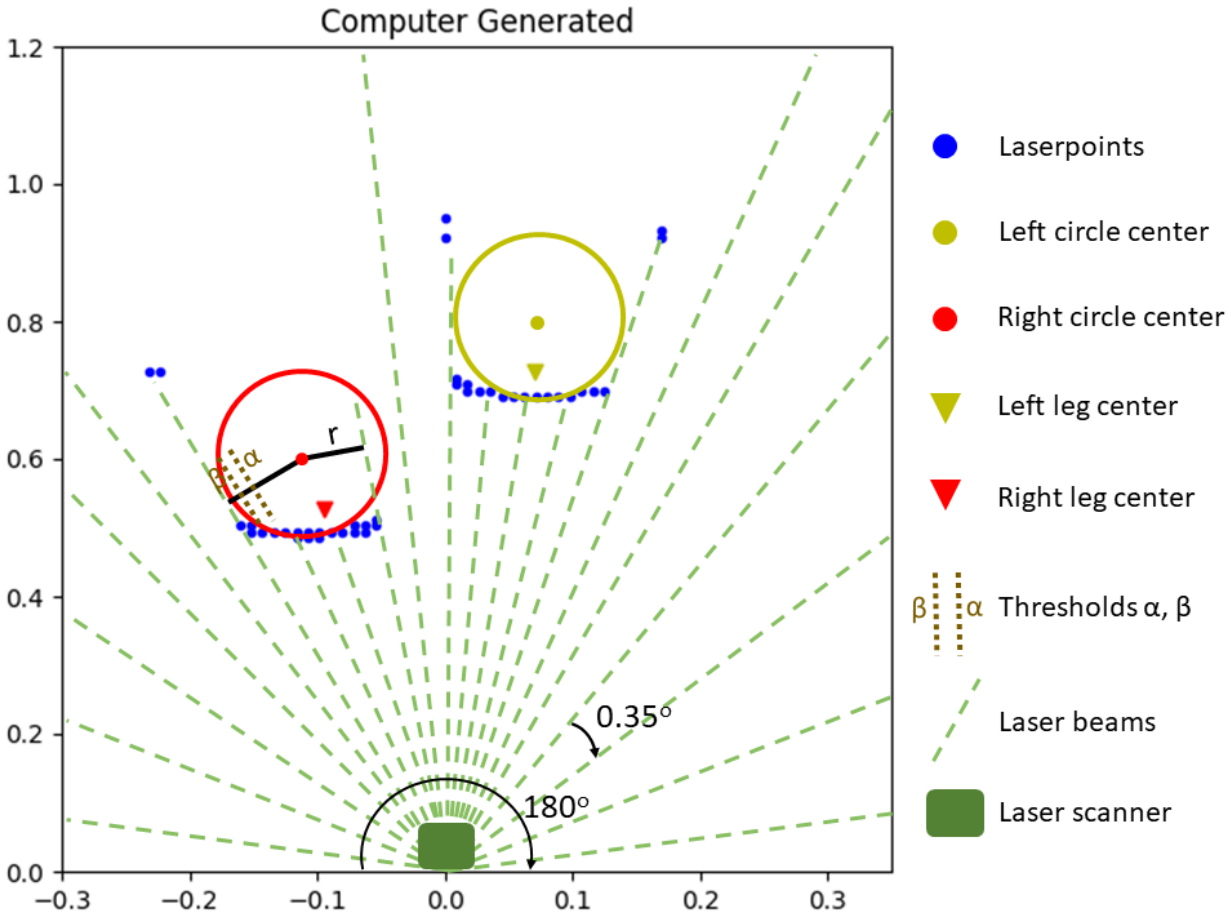


Figure 3.3: A detailed presentation of the method used for the data generation. The length of the laser beams represents the distance of the detected objects, as calculated by the laser sensor.

patterns, rather than memorize the ones given in the training input. All the above were the motivation for creating computer generated data.

The idea was to imitate the walker setup and simulate hypothetical patient's legs. It is assumed that there is a laser lying at the origin and has vision of the patient's legs, which are facing the rollator. This laser has 180° area scanning range and $\sim 0.35°$ angular resolution.

First, we designed two circles representing the legs. We assumed that these circles would move sinusoidally, both on x and y axes. This assumption was made due to the nature of leg movement, an example of which is demonstrated in Fig. 3.4, where we show the y coordinate of the center of a patient's leg in time. In this figure, it is obvious that a sine is able to represent well the leg movement on the y axis.

We define the center of each leg's movement as the point with coordinates the equilibrium points of the sinuses on the x and y axes. We also define the gait center as the midpoint between the movement centers of the two legs. The amplitudes of the y sinuses of both legs are almost the same, but have π phase difference, so that when the one moves forward, the

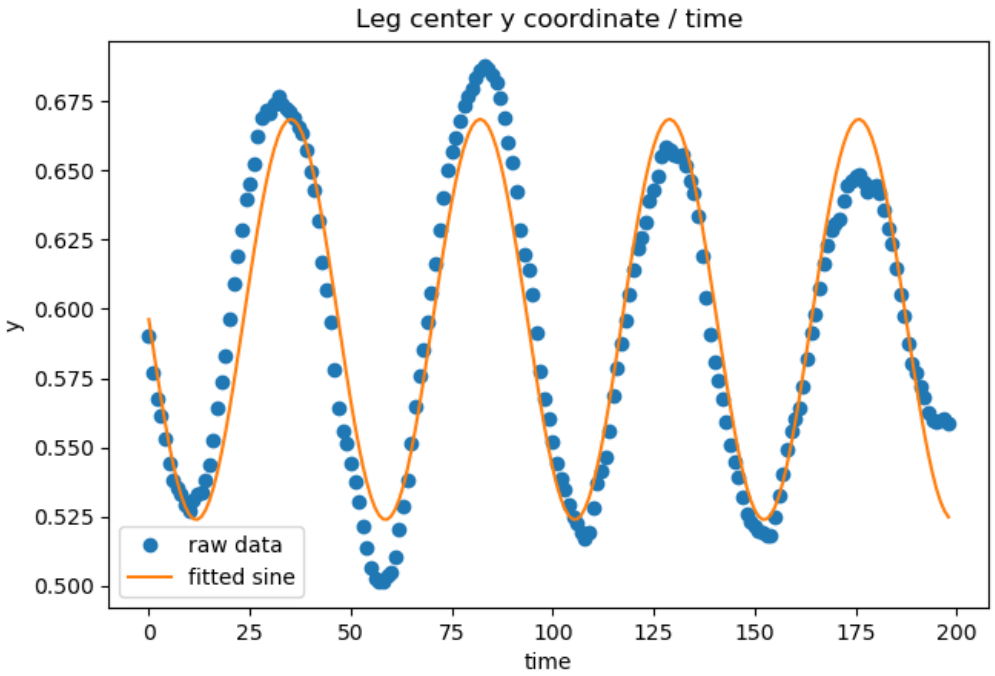


Figure 3.4: Visual indication of the sinusoid behaviour of the legs' motion. An experiment is presented including a sine fitted using least square error.

other moves backwards. Notice that the backwards moving leg is in reality stationary, but the rollator frame is moving away from it. Thus the leg seems to be moving backwards, as all positions are calculated relatively to the rollator frame. The sinusoid on the x axis is of a much smaller amplitude and frequency than the one on the y axis, as well as of a slightly irregular phase increase at each time step, in order to simulate the occasional slight shift of the whole walking movement to the left or the right.

With the aforementioned setup, the patient has a direction parallel to the y axis, something that has to change over the course of time. To imitate different gait directions, before we virtually sample from the laser sensor, we rotate the leg centers around the gait center by an angle we consider the **direction**. This direction changes through time smoothly by picking regularly a new target direction and increasingly approaching it in each time step. If the direction is above a threshold angle, leg occlusions can occur as one leg completely blocks the other with respect to the laser sensor. This phenomenon is a rare occasion in the experimental data and the most difficult situation faced by any leg tracking model. Therefore we implemented an option in which the direction remains in such high angles so that occlusions occur in every step. This way the RNN part of the network will have many instances of occlusions to train on to.

To virtually sample from the laser sensor, we emulate each laser beam as a straight line passing through the origin, where the sensor is lying (Fig. 3.3). To find the points that the laser sensor would produce when detecting the legs, we use the distance r of each

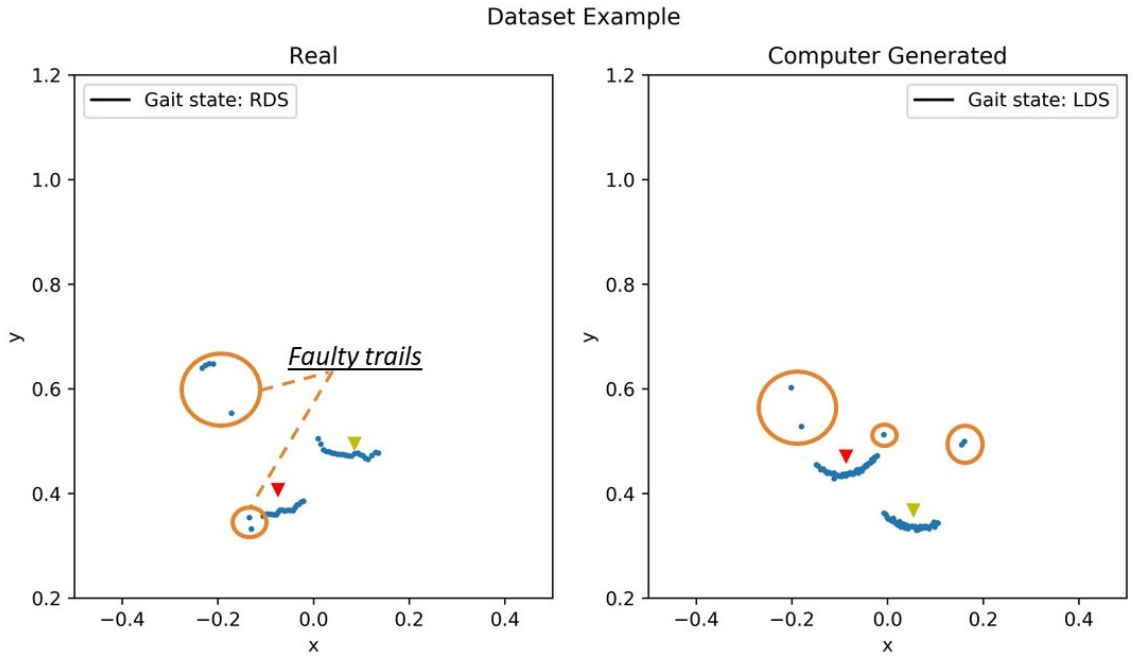


Figure 3.5: Two frames of the dataset, one real and one computer generated. On both frames are annotated with blue the leg points, with a red triangle the centers of the right legs and with a yellow one the centers of the left legs. The faulty trails are also marked with orange on both frames.

laser beam from the leg center and the intersection points of the laser beam and the leg circle. If this distance is lower than a threshold α, dependent on the leg circle radius, then the intersection point that lies closer to the sensor is used. In order to create more realistic data, those intersection points are shifted in the direction of the laser beam using a Gaussian noise with zero mean and standard deviation 2mm, suitable to simulate the nominal error of the real sensor.

Next we simulate the faulty trailing patterns, which are often present in 2D laser data. These patterns are mostly noticeable when laser beams have great incidence angles with the legs and due to scattering, the laser sensor receives signals of lower intensity, assuming a greater distance to the obstacle (Fig. 3.5). To calculate those, we select the intersection points of the laser beams with r greater than α but lower than a second threshold β and shift them on the laser beam direction by an amount analogous to the tangent of the incidence angle (Fig. 3.3). As this tangent grows to infinity when the laser beam becomes tangent to the circle, we limit this shift and apply a Gaussian noise in order to simulate the empirically noticed clustering of points at a certain distance.

To calculate the gait state of each time frame, we only used the phase of the right leg's sinusoidal motion on the y axis. The phases at which the person changes state were deduced using the frequency analysis of normal human gait found in [7, 8]. We assume that every leg stays on stance 60% of each gait cycle and 40% on swing, meaning that every gait cycle consists of 10% LDS, 40% LS/RW, 10% RDS and 40% RS/LW.

After a selective choice of the real data, the application of data augmentation and the creation of the computer generated data, the final dataset consists of around 210000 frames used for training, validating and testing the neural network.

Chapter 4

The Leg Tracking and Gait Analysis Network

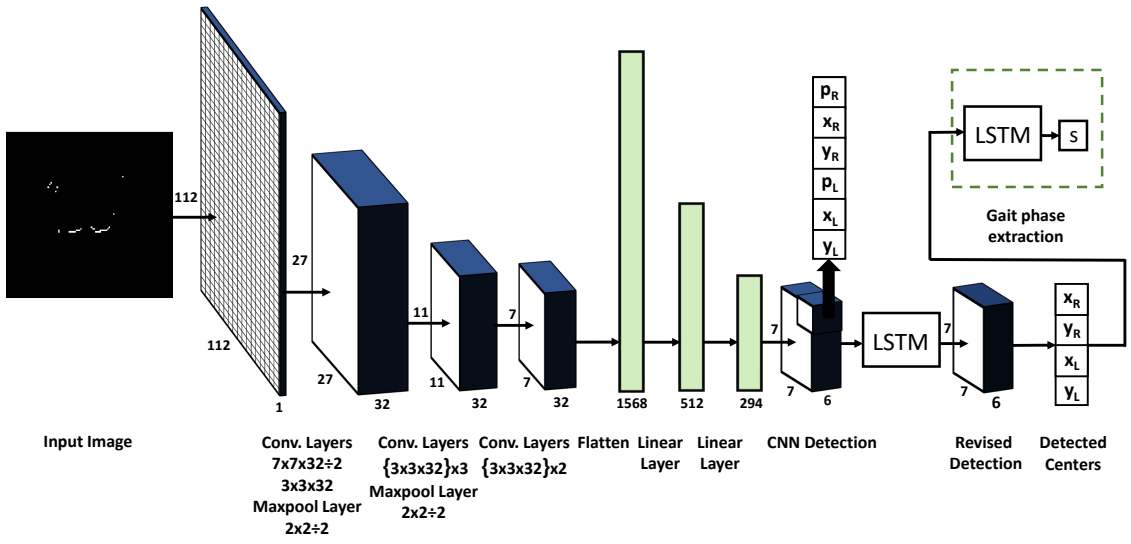


Figure 4.1: The architecture of our proposed framework for leg tracking by detection composed of Convolutional and Linear Layers followed by an LSTM. The output of the leg detection is fed into an LSTM that estimates the gait state.

In this section we present the neural network architecture. We explain its compartments, the architectural decisions made for each one of them, but also the attempts made on other architectures, which were not implemented in the end.
The LTGADnet is a combination of two models:

1. the Leg Tracking one, which is responsible for detecting and tracking the legs of the patient using a robotic walker and

2. the Gait Analysis one, which is responsible for extracting the gait state of the patient, as shown in Table 1.1.

4.1 Leg Tracking Network

The Leg Tracking network is the responsible one for the detection and tracking of the patient's legs. Its task is to get as input the data provided by the laser and find the centers of the legs, while handling the occasions of leg occlusions.

4.1.1 Input

The input of the Leg Tracking network are the laser data. We assume that the legs lie in the laser angle range and that no other objects can obscure the legs. It is also assumed that no other legs exist in the network input, which is the case of normal usage of a robotic rollator.
The data provided by the laser sensor are distances and angles of the detected obstacles to the laser position. These distances are transformed into coordinates of the position of the obstacles, using equations:

$$\begin{aligned} x &= dist \cdot \sin a \\ y &= dist \cdot \cos a \end{aligned} \tag{4.1}$$

where $dist$, a the distance measured by the laser sensor and the beam angle in the laser data. The laser frame axes system is assumed as in Fig. 1.1.
The neural network is designed to get as input a square image of size 112×112. This size is significantly smaller than the one in YOLO and was found to be more than sufficient for our task (section 4.1.5). This image accounts for an occupancy grid of the laser data. Such a grid represents the space in front of the laser and contains a 1 in the cells that contain an obstacle and a 0 in the cells that no obstacle exists or no obstacle was detected.
As the laser data contain much more information than is actually needed, since the legs cover only a certain portion of the scanned area, we create a bounding box, in which we assume that the legs lie on. This bounding box needs to be of size $1m \times 1m$, in order for the image to be well proportioned, but its coordinates can be tuned according to the needs of any rollator setup. So, from the points produced by Eq. 4.1 only the ones lying in the bounding box are kept. For the LTGAD database, the bounding box was chosen to have limits $(-0.5m, 0.5m)$ and $(0.2m, 1.2m)$ on the x and y axis respectively.
In order to create the image, we need to map the real coordinates into image cells. For this purpose, the new coordinates become:

$$\begin{aligned} x_{new} &= s - 1 - s \cdot \frac{y - h_{min}}{h_{max} - h_{min}} \\ y_{new} &= s \cdot \frac{x - w_{min}}{w_{max} - w_{min}} \end{aligned}$$

where $s \times s$ the image size and $h_{min}, h_{max}, w_{min}, w_{max}$ the min and max height and width of the bounding box respectively. These new coordinates belong to the cells that will contain a 1 in the image. In this way, we have created the input image of the leg tracking network.

4.1.2 Output

The output of the leg tracking network is of size $6 \times 7 \times 7$. The 7×7 grid is similar to the one in YOLO and represents the grid with which we segment the initial image. For each cell of this grid, the neural network provides 6 values, 3 for each leg:

1. The probability that the leg's center exists in that grid cell
2. If the center exists, the x coordinate of the leg center. This coordinate can take a value in $[0, 1)$ and represents the relative position of on the x the center in the cell.
3. If the center exists, the y coordinate of the leg center. Same as with x, this coordinate can take a value in $[0, 1)$ and represents the relative position on the y of the center in the cell.

The difference with the YOLO architecture is that in our task we know that no more than two legs will be visible at all times, so we exploit that by forcing the output of each grid cell to predict two leg centers. For each one of the two legs, the grid cell that outputs the highest probability will be considered the one containing that leg's center. From these cells we calculate the detected leg centers. We first add the grid cell's coordinates with the x, y coordinates it contains, and then we map them back to meters.

4.1.3 Architecture

The Leg Tracking network consists of two sub-networks: (1) a CNN for leg detection and (2) an LSTM for handling occlusions. The CNN takes as input an 112×112 image, which is created as explained in 4.1.1, and gives as output a $6 \times 7 \times 7$ grid, as explained in 4.1.2. This output is then forwarded to a simple one-layered LSTM, which outputs a revised version of it.
The CNN has 7 convolutional layers. Each one is followed by a ReLU. Max-pooling layers intervene, in order to further reduce the data dimensions. More specifically, the layers of the CNN are: $7 \times 7 \times 32 \div 2$ Convolution (e.g. kernel size 7, number of channels 32, stride 2), $3 \times 3 \times 32$ Convolution, $2 \times 2 \div 2$ Max Pooling, $3 \times 3 \times 32$ Convolution, $3 \times 3 \times 32$ Convolution, $3 \times 3 \times 32$ Convolution, $2 \times 2 \div 2$ Max Pooling, $3 \times 3 \times 32$ Convolution, $3 \times 3 \times 32$ Convolution.
Then, two fully-connected layers follow, of size 1568×500 and 500×294, where $294 = 7 \times 7 \times 6$ the grid size. This is the detection output of the CNN. All layers are followed by the ReLU activation function. Batch normalization is also used after every convolutional layer, which significantly increases the network's accuracy. For training, we applied dropout 0.5 after the first linear layer.

```
self.cnn_layers = Sequential(
    Conv2d(1, 32, kernel_size=7, stride=2),
    BatchNorm2d(32),
    ReLU(inplace=True),
    Conv2d(32, 32, kernel_size=3, padding = 2),
    BatchNorm2d(32),
    ReLU(inplace=True),
    MaxPool2d(kernel_size=2, stride=2),
    Conv2d(32, 32, kernel_size=3, padding = 1),
    BatchNorm2d(32),
    ReLU(inplace=True),
    Conv2d(32, 32, kernel_size=3),
    BatchNorm2d(32),
    ReLU(inplace=True),
    Conv2d(32, 32, kernel_size=3),
    BatchNorm2d(32),
    ReLU(inplace=True),
    MaxPool2d(kernel_size=2, stride=2),
    Conv2d(32, 32, kernel_size=3),
    BatchNorm2d(32),
    ReLU(inplace=True),
    Conv2d(32, 32, kernel_size=3),
    BatchNorm2d(32),
    ReLU(inplace=True),
)

self.linear_layers = Sequential(
    Linear(1568, 512),
    Dropout(0.5),
    ReLU(inplace=True),
    Linear(512, 294),
    ReLU(inplace=True)
)
```

Figure 4.2: A snippet of the code written in PyTorch for the CNN, for a clear representation of the layer content and succession. The input image flows from the CNN layers to the linear layers and so the detection output is produced.

4.1.4 Training

The CNN and LSTM were trained separately. This was necessary for the LSTM training, so that it would learn how to revise the detections of a well trained CNN. The loss function used for training is the following:

$$\begin{aligned} Confidence\ Loss &= \sum_{i=1}^{7}\sum_{j=1}^{7}\sum_{k=1}^{2}(C_{ijk} - \hat{C}_{ijk})^2 \\ Detection\ Loss &= \sum_{i=1}^{2}(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ Loss &= Confidence\ Loss + \alpha \cdot Detection\ Loss \end{aligned} \tag{4.2}$$

where C_{ijk} is the expected confidence of the grid cell ij (1 if the cell contains leg k and 0 for the rest), $\hat{C}_{ijk}$ the output confidence of the grid cell ij for leg k, x_i, y_i the expected position of leg i in the grid and $\hat{x}_i, \hat{y}_i$ the position of leg i in the grid as a result of the output.
Note that this loss function is different from the one used in YOLO. Not only is the bounding boxes' loss absent, but also the detection loss is unusually defined; in YOLO the detection loss is calculated between the grid cell actually containing the object and its corresponding grid cell in the output, whereas we calculate it between the grid cell actually containing the object and the grid cell that the network considers as the object container. We tested both losses, with our loss achieving higher accuracy.
In training, $\alpha = 5$ proved to achieve the best results for both the CNN and LSTM training. After some experimentation on the batch size with no obvious impact, batch size 32 was used.
For the CNN, Adam was used as an optimizer, with a learning rate starting at 10^{-4} and decaying at 10^{-5}, 10^{-6} and 10^{-7} at epochs 15, 30 and 50 respectively. The training ran for 100 epochs. Also, in order to avoid overfitting, a technique similar to early stopping was applied. Every time the loss at the validation set decreased, we saved the current model. The last model to be saved was the one to be finally used. In this way, we know that what the neural network is learning is not a result of memorizing the dataset.
For the LSTM, Adam was used as an optimizer, with a learning rate starting at 10^{-4} and decaying at 10^{-5} and 10^{-6} at epochs 25 and 50 respectively. The training ran for 50 epochs. The same technique used for the CNN for early stopping was applied.

4.1.5 Considered Approaches

Many different architectures and architectural parameters were considered for the leg tracking network towards increasing the neural network's accuracy. The results of these architectures will be thoroughly presented in section 5.1. In this section we explain all the considered architectures and the idea behind choosing these.

- Sigmoid activation function after the last fully-connected layer. The first idea was to apply batch normalization after every convolutional layer. We discovered that, although the batch normalization helped the training process and the model to learn more accurately, on the validation or the testing of the model, where we set model.eval() on PyTorch, batch normalization significantly decreased the accuracy and the model behaved as if it had not learned. After excessively trying to fix this problem, we decided to either use batch normalization without model.eval() on testing or replace batch normalization with a Sigmoid as activation function on the last fully-connected layer, in order to map the output on the $[0, 1]$. The batch normalization still produced better results than the sigmoid, so we finally chose the first option.

- Input image size 224×224 with the corresponding extra CNN layers needed. This

change though had no effect on the network's accuracy, which was expected, considering that image size of 112 already has enough high accuracy, with each image cell representing a $\sim 9mm^2$ of the scanned space.

- More or less channels on every convolutional layer with the corresponding changes on the linear layer size. The number of channels determines the complexity of the CNN and thus the model's ability to learn or its tendency to overfit, if the complexity is too high. The combinations tested were:

  1. 16 channels on every convolutional layer with one fully-connected layer 784×294 (CNN.16.294)
  2. 16 channels on every convolutional layer with the same fully-connected layers (CNN.16.500)
  3. 32 channels on every convolutional layer with one fully-connected layer 1568×294 (CNN.32.294)
  4. 64 channels on every convolutional layer with the first fully-connected layer being 3136×500 (CNN.64.500)
  5. 64 channels on every convolutional layer with the first fully-connected layer being 3136×1500 (CNN.64.1500)

  All these different approaches suggested that even 16 channels per layer were enough on a specific experiment testing, but we decided to choose 32 channels, as the results could change when validating with leave-one-out. Also, no significant effect on the frequency was noticed when the channels are doubled.

- GRU instead of LSTM. This change was applied in order to check if the model with the LSTM overfitted and would stop learning slower with the GRU. We discovered that the GRU did not have any notable impact on the model's accuracy.

- TCN instead of the LSTM. The TCN tested was a Dilated TCN with max dilation 8. The TCN though had lower accuracy than the LSTM and was thus not used in the final model.

- Two layer LSTM. This change did not significantly increase the network's accuracy, meaning that the one layer LSTM is already capable of dealing with our problem.

- Different loss function for the LSTM training. We considered adding on the CNN's loss function another term, the association loss. With this term our goal is for the LSTM to correlate successive detection frames and produce a revised detection close to the one on the previous frame. The association loss is defined as:

$$Association\ Loss = \sum_{i=1}^{2} (\hat{x}_i^t - \hat{x}_i^{t-1})^2 + (\hat{y}_i^t - \hat{y}_i^{t-1})^2 \tag{4.3}$$

where $\hat{x}_i^t, \hat{y}_i^t$ the estimated x, y coordinates of the center of leg i in the grid for time frame t and x_i^{t-1}, y_i^{t-1} the estimated $\hat{x}$, $\hat{y}$ coordinates of the center of leg i in the grid for time frame $t-1$. This loss did indeed help the LSTM learn to handle leg occlusions, as a correlative loss function helps towards keeping an estimate of the center of the temporarily invisible leg. Thus it was finally used in the training.

4.2 Gait Analysis Network

The gait analysis network is the one responsible for estimating the gait state of the patient, using the leg center prediction from the leg tracking network.

4.2.1 Architecture

The input for this network are the leg centers extracted from the revised detection grid output of the leg tracking LSTM. The output of the network is the one-hot key encoding of the estimated gait state, as shown in Table 1.1. The LSTM has five layers, each having an independent memory. These LSTMs are stacked, with the whole LSTM's output being the output of the last LSTM. All LSTM layers have input and hidden size 4.

4.2.2 Training

For the training of the gait analysis network we used cross-entropy loss. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. In our case, this loss function takes as input 4 probability values, one for each gait state. It first applies the Softmax activation function on them and then it calculates the loss. This process is described below:

$$f_i(s) = \frac{e^{s_i}}{\sum_{j=1}^{4} e^{s_j}}$$

$$Loss = -\sum_{i=1}^{4} t_i \log\left(f_i(s)\right)$$

where $f_i(s)$ the result of the Softmax for every class probability i, t_i the true class probability and $loss$ the final cross-entropy loss.

As the gait state estimation, if treated as a classification problem, has an inherent class imbalance, with only 19% and 18.6% of the states belonging to LDS and RDS respectively, we used weighted cross entropy loss for the training, with inversely proportional weights for every class. Batch size 32 was used for the training. We used Adam as an optimizer, with a learning rate starting at 10^{-4} and decaying at 10^{-5} and 10^{-6} at epochs 25 and 50 respectively. The training ran for 50 epochs with dropout 0.3 applied between LSTM layers. The same technique used for the leg tracking network's training for early stopping was applied.

4.2.3 Considered Approaches

Many different approaches were tested on the gait analysis network. A single-layered LSTM, a bigger hidden size on every LSTM layer with a fully-connected layer in the end mapping the hidden size to the 4 values of the one-hot encoding, a multi-layered LSTM with output the sum of every layer's output, or a bigger input, which could be the whole detection grid, not only the leg centers. Overall, we concluded that the 5-layered was the most capable in tackling the gait analysis problem, showing sufficient but still not optimal accuracy, which will be further discussed in section 5.2.

Chapter 5

Experimental Results

For the leg tracking network, we use as metrics the mean, max, and median euclidean distance between the detected centers and the annotated ones. The max and median distances are provided, as rare occasions of one-frame faulty detections from the CNN can skew the mean distance upwards. For the gait analysis network, we calculated the overall accuracy of the gait state detection and the Recall, Precision, and F1 scores. For the latter three metrics, we use the weighted mean over all gait states, due to the inherent class imbalance in the data, with stance states (s_1, s_3) covering only $\sim 38\%$ of the dataset.
We compare our results with the ones in [1], which is the state-of-the-art leg tracking and gait analysis algorithm and the only one that tackles the same problem as this paper. We could not compare with other works in the literature, such as [36, 38], as they perform person tracking instead of individual leg tracking. At the same time, their need for specific thresholds and parameterization makes them inapplicable for our study.

5.1 Ablation Study

To justify the architectural design choices for our leg tracking and gait analysis network of Fig. 4.1, we present an ablation study in Table 5.1. We specifically focus on ablating the tracking by detection part of the framework, where various design choices had to be made. We do not refer to the gait analysis one, as its instability would not make an ablation study valid. The different architectures that we tested are compared based on the mean and the max euclidean distances between the ground truth leg centers and the produced ones in the validation set. The different designs that were tested are:

- (a)-(f) multiple variations of the CNN architecture, as described in section 4.1.5, including the final one `CNN.32.500`,
- (g) a full network (as in Fig. 4.1) with a one-layer LSTM trained with loss function (4.2) and $\alpha = 5$, noted as `LSTM1`,
- (h) a full network (as in Fig. 4.1), but with a two-layer LSTM trained with loss function (4.2) and $\alpha = 5$, noted as `LSTM2`,

Table 5.1: Ablation Study

Architectures	Mean distance (cm)	Max distance (cm)
CNN.16.294	4.03	60.64
CNN.16.500	3.97	60.61
CNN.32.294	4.17	60.33
CNN.32.500	**4.04**	**60.4**
CNN.64.500	4.08	60.93
CNN.64.1500	4.30	68.89
LSTM1	**3.16**	**15.26**
LSTM2	3.15	19.21
TCN	3.47	40.10
LSTM1assoc	3.21	13.51
TCNassoc	3.87	48.80
GRU	3.99	58.15

- (i) a full network (as in Fig. 4.1) with a Dilated TCN instead of the LSTM with max dilation 8, trained with loss function (4.2) and $a = 5$, noted as TCN,
- (j) a full network (as in Fig. 4.1) with a one-layer LSTM trained with loss function (4.2) with $\alpha = 5$ plus an additional leg association loss (4.3) with a weight $\beta = 0.1$, noted as LSTM1assoc,
- (k) a full network with TCN instead of the LSTM, with max dilation 8, trained with loss function (4.2) plus the leg association loss (4.3) and parameters $\alpha = 5, \beta = 0.1$, noted as TCNassoc and,
- (l) a full network (as in Fig. 4.1) with a GRU instead of the LSTM, trained with loss function (4.2) and $a = 5$, noted as TCN.

Inspecting the results in Table 5.1, we can see that all CNN architecture can learn to efficiently solve our problem, even the CNN.16.294 and CNN.16.500, which only have 16 filters per layer. In the end, we chose though the CNN.32.500, as it does not enormously increase the complexity and this way we can ensure that the model would have enough parameters if in the future was trained with more data. We also notice that overall an architecture based on LSTMs provides more accurate results than TCNs. TCNs' performance depends on the training batch size and has a high memory cost as it needs long sequences of data. Our dataset of real patient walking instances limits us to short batch training, in which the LSTM with its efficient memory handling beats TCN's performance. Moreover, the 2-layer LSTM (LSTM2) does not improve the detection performance considerably. Notably, we found that the addition of the association error in LSTMassoc, TCNassoc does not seem to have a great effect on the performance, but in cases of occlusions it has a visible improvement, as it keeps a better track of the occluded leg, something that we cannot see from

Table 5.2: Leave-one-out validation results

Leg Tracking									
Metric \ Experiment	1	2	3	4	5	6	7	8	mean
Mean (cm)	2.42	2.04	3.28	4.11	4.57	2.46	3.26	3.68	3.23
Max (cm)	9.63	9.87	22.16	58.32	14.59	7.79	30.31	42.95	-
Median (cm)	2.21	1.81	2.60	2.95	3.74	2.18	2.75	3.31	2.69
Gait Analysis									
Metric \ Experiment	1	2	3	4	5	6	7	8	mean
Accuracy (%)	75.60	71.56	68.85	72.74	69.27	63.17	69.36	75.89	70.805
Precision (%)	78.88	71.36	68.87	74.80	70.76	67.85	69.38	77.50	72.425
Recall (%)	75.60	71.57	68.85	72.74	69.27	63.17	69.37	75.89	70.8075
F1 score (%)	76.57	71.28	67.62	72.10	68.68	61.94	69.21	76.24	70.455

a simple mean and max metric. So, we conclude that the best architecture was the one with the 1-layer LSTM (`LSTM1assoc`), as depicted in Fig. 4.1, trained with loss function (4.2) plus the leg association loss (4.3). This shows that the combination of the superior feature extraction of CNNs, when combined with the ability of LSTMs to encode temporal dynamics and predict the evolution of the targets over time, is an effective method for challenging dynamic tasks, including the leg tracking by detection from 2D range data. Our lightweight architecture contains 1703958 parameters, making it very efficient for real-time performance on any mobile robotic assistant, like the one in Fig.1.1.

5.2 Leave-one-out validation

We further validated our proposed LTGADnet using a leave-one-out cross-validation strategy. In this case, for every training session, we exclude one experiment and use it for testing, while another one is used for validation. The results of the cross-validation are shown in Table 5.2. We performed tests both for tracking and gait analysis. Regarding the leg tracking, our results show the overall good performance of our method across different combinations of train/tests, indicating the generalization ability of LTGADnet in tracking legs of real patients, who suffer from various pathologies that affect their walking performance (hence, variable dynamics to be learned by our network).

Moreover, we report our results on the Gait Analysis problem. Here, our results are found sufficient yet not optimal. Our average accuracy over all gait phases and all tests is $\sim 71\%$ with an F1-score of $\sim 70\%$. This result stems from two major parameters. (i) Pathological walking comprises great variability in the different gait phases [7]. The representation of the gait phases as in Table 1.1, imposes the difficulty of recognizing the joint state of the legs, needing difficult dynamics to be extracted from 2D data. Note that, for example, a swing phase initiates when the toe leaves the ground [7], which is very difficult to be captured by the 2D representation of the leg movement (especially when detecting

points on the tibia). Moreover, the DS phases are very short subphases, though crucial for transitioning in walking, but tough to be detected. In our dataset, only 19% of the instances belong to the LDS and 18.6% to the RDS phase, making our dataset highly imbalanced, reflected in our results. (ii) Our dataset seems to be small for such a demanding task. To understand how variable gait dynamics are, we should consider that a normal gait cycle is usually composed of 60% stance and 40% swing approximately [7], while in our dataset, subject 1 has on average 62.60% stance and 37.40% swing, while subject 6 has 73.14% stance and 26.86% swing per gait cycle (subjects picked randomly). Note that we only trained/tested on instances of walking activity, as such detection is possible in our overall integrated intelligent system, showcased in [10]. The high variability in the duration of the phases, together with the high sensitivity over the potentially different pathological walking dynamics, demands a much broader dataset than the one we are experimenting with.

5.3 Network Output Examples

In this section we present some plots to demonstrate the results of our method. We exhibit the behavior of our method in leg tracking and gait analysis on normal cases, cases of occlusions, as well as on data deriving from a laser sensor with different characteristics.
In Fig. 5.1 we show an example of occlusion handling by the leg tracking network. We can see that, although the right leg stays almost invisible for 10 frames, the network is able to keep track of it and smoothly recover when the leg is visible again. In this figure are very well illustrated the problems existing in the real dataset. The annotated left centers in frames #400 and #402 seem to be misplaced, as well as the annotated right centers in frames #408 − #412. This is a result of the inherent difficulties in capturing data from real elderly patients, where it is important to make the process as easy and straightforward as possible, with minimal re-calibrations and retakes.

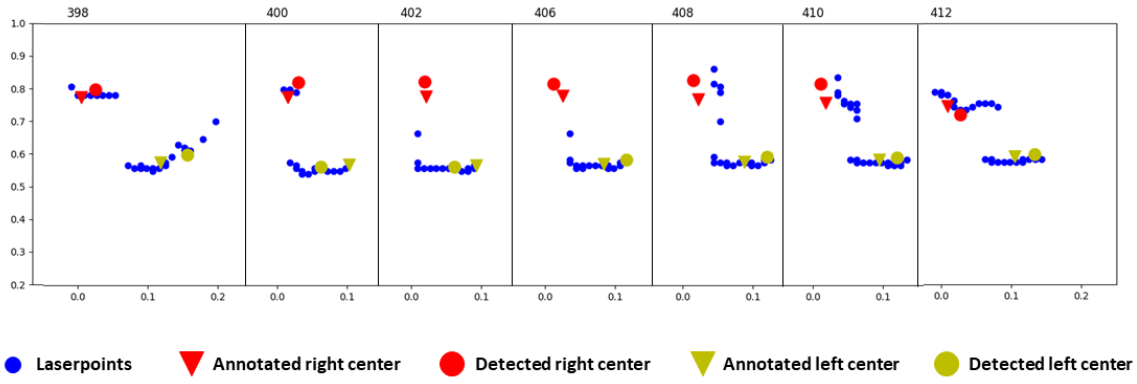


Figure 5.1: An example of input with occlusion and the output of the leg tracking network. In-between frames have been omitted for space-saving purposes.

We can though also see in the figure the importance of the computer generated data and their contribution in balancing the inaccurate annotations, as in frames $\#400, \#402$ the detected left center seems to be more accurate than the annotated one. Unfortunately, the computer generated data are only capable of improving the detection to some extent, with left leg center detections in frames $\#406 - \#412$ drifting from the actual leg centers, similarly to the annotated ones in frames $\#400, \#402$.

In Fig. 5.2 we demonstrate a normal input data case and the output of the leg tracking network. In this example, it can be seen that due to the use of the computer generated data, it is common for the output of the neural network to produce more accurate leg centers than the annotated ones (frames $\#250$, $\#261 - \#267$). Generally, the accuracy and efficiency of the leg tracking network is obvious, with the network always detecting both legs and keeping track of them, without mixing them up.

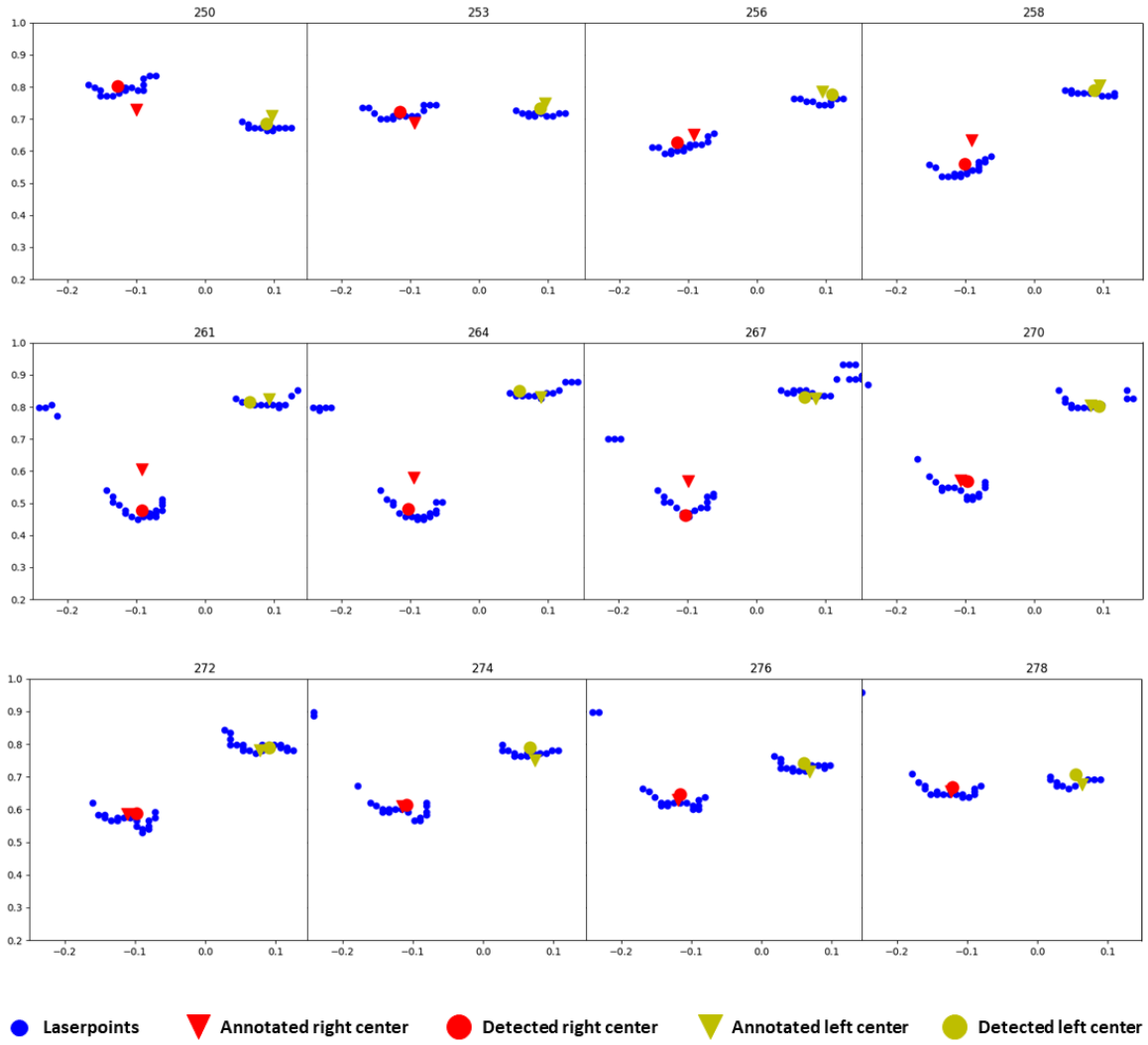


Figure 5.2: An example of the leg tracking output. In-between frames have been omitted for space-saving purposes.

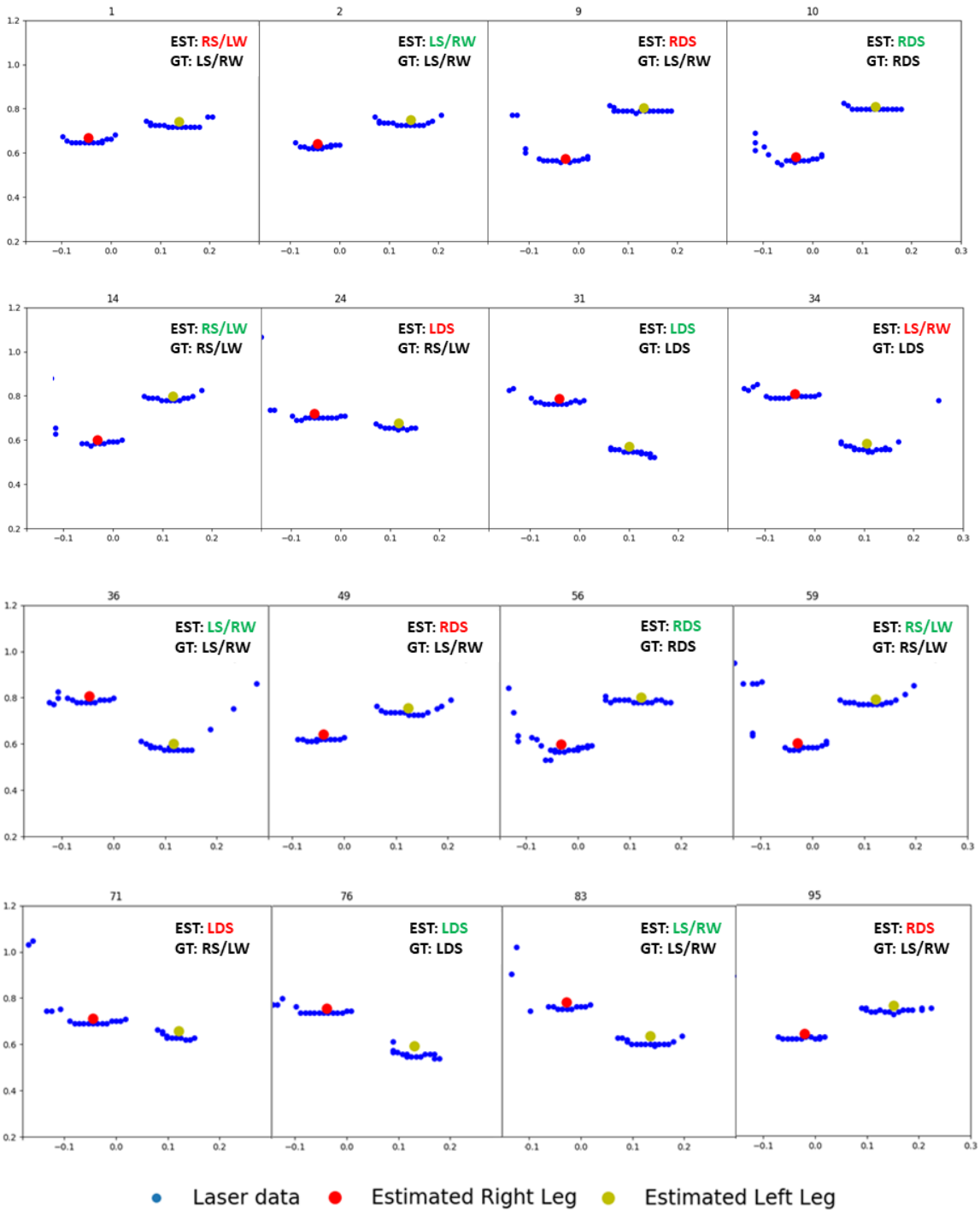


Figure 5.3: An example of the gait analysis output. On the figure are annotated the leg centers produced by the leg tracking network, as well as the estimated gait state versus the ground truth. Only the frames at which a change at either the estimated or the annotated gait state occurs are presented.

Faulty trails, as described in section 3.3 are not mistaken for legs, nor influencing the leg

center detection. Even though the shape of the laser points for each leg changes significantly through time, the network is capable of recognizing and keeping track of them.

In Fig. 5.3 we present an example of the output of the gait analysis network. The number of total frames in the example is 95 and the number of the frames with correct gait estimation are 71, resulting in accuracy $\simeq$ 74.74%. The class imbalance of the dataset is very well illustrated. We can also see how the network in the first frame returns a false estimation of the gait state, which is understandable, as it is impossible to know from a single frame the gait state. The network though, seeing the distance increase between the leg centers immediately in the next frame fixes its first guess. A higher accuracy for the gait analysis network may be important for using the gait states and making conclusions about the patient's health, but its behavior with the current insufficient dataset is promising and indicates that this network could be capable of solving the problem of gait analysis efficiently.

In Fig. 5.4 we show the output of the final network, when it receives as input data from a different laser sensor than the one used for training, validating and testing. We test the performance of the network on the i-Walk [10], equipped with a Hokuyo LiDAR UST-10LX, with mean sampling period of about 25msec/scan, scanning range of 0.06 to 10m, angle range -135^o to 135^o and angular resolution 0.25^o. The leg tracking was for once more able to track both leg centers. The gait analysis network presents overall good performance, but seems to predict a premature change on the gait state on frame #127, which though corrects in the next frames.

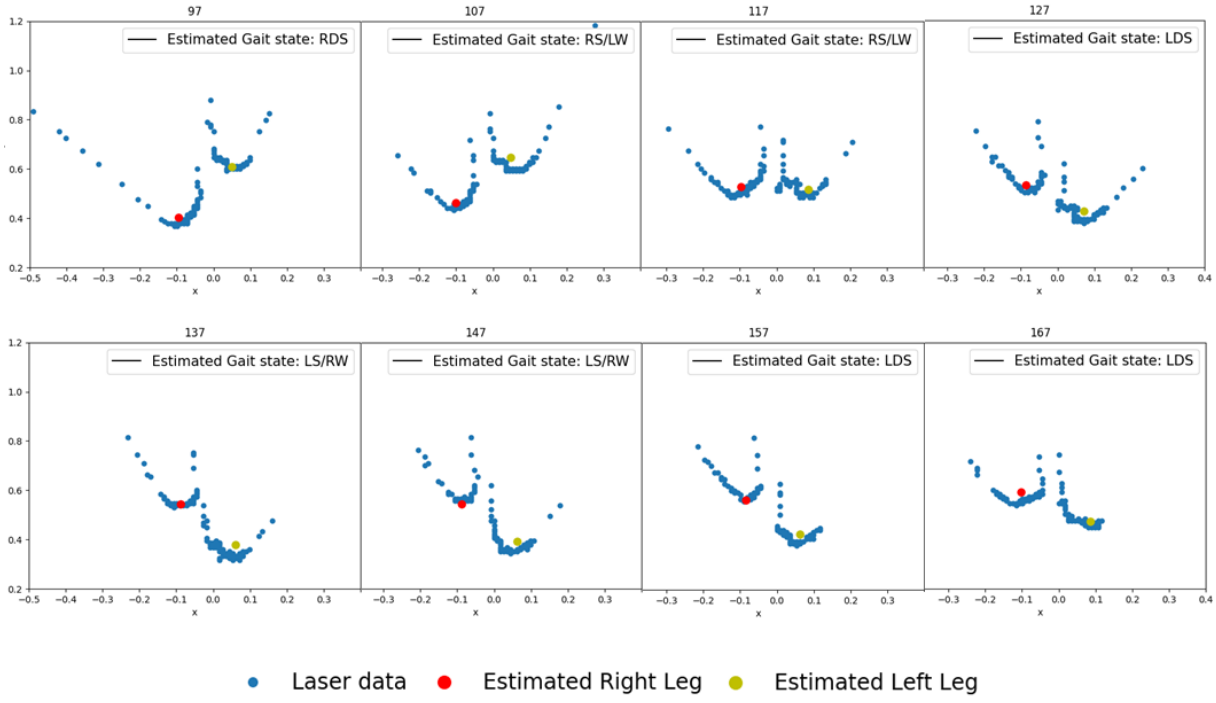


Figure 5.4: An example of testing both networks on data from a different laser sensor with bigger area scanning range and angular resolution.

In this input example the leg tracking network was more than capable to deal with the different input structure, even though it did not conform to the training dataset, whereas the gait analysis network, considering the difficulties in the dataset, managed to produce satisfactory gait state estimates.

5.4 Comparison to state-of-the-art

The most relative work on the literature and the one that this thesis aimed to improve is the one by Chalvatzaki et al. [1]. Their probabilistic approach with Hidden Markov models and particle filters delivered an average mean distance error of 6.69cm while requiring more resources, as the particle filter is computationally more expensive, and therefore challenging to be compatible with the laser scanner's frame rate (40Hz) for online performance. Our lightweight deep method achieves an impressive 52% improvement over the state-of-the-art in tracking accuracy and is able to run real time on any range sensor, as it can predict the leg centers and the gait state on a single frame in approximately 2ms, using a GeForce GTX 1060 6GB GPU. All the above make our network an efficient leg tracking method to be employed in any mobility assistant robot equipped with a 2D range sensor.

5.5 Discussion

It is obvious that the network produces highly accurate leg center detections, even in the difficult occasions of leg occlusions. We have concluded that the real data augmentation combined with the computer generated data have played a significant role in the accuracy of the leg tracking, as the annotated centers were common to diverge from the actual leg centers, due to the difficulties in extracting data from real elderly patients.
Dealing with the input as an image by turning it into an occupancy map enabled us to use for the leg detection and center extraction a CNN, which was once more proven to successfully cope with object detection. The RNN combined with the association loss was able to track both legs, including situations when one leg is occluded by the other. This was achieved in fast real time performance, making the leg tracking network more than suitable for online tracking tasks.
The gait analysis network shows overall good performance and is very lightweight. The LSTM shows promising results in gait analysis, with its memory mechanism being able to learn the succession of the states in a gait cycle and the possible transition points between them. Although, in order for it to be used in real life, for example by doctors or rehabilitation staff for making conclusions about the patient's health, a higher produced accuracy would definitely be required. As we tested many different architectures for the gait analysis network, we concluded that a higher accuracy calls for an extended collection of real gait data. The data augmentation and computer generated data, which on the leg tracking network were immensely helpful, are not that effective on the gait analysis one. Concerning the computer generated data, we created them following the gait analysis

statistics, which show that the percentages of swing state and stance state are around 60% and 40% respectively. In this way, we are lead to the creation of a computer generated dataset lacking in variety, which could not though get much further improved, as the imitation of the real gait is a very difficult task. We believe that capturing more gait analysis data would stabilize the neural network training and enable it to learn a lot more efficiently.

Chapter 6

Conclusion and Future Work

We proposed LTGADnet, a novel, lightweight deep learning architecture for efficient human leg tracking and gait analysis from 2D range data. LTGADnet can be employed as an off-the-shelf method to any mobility assistant robot equipped with a laser sensor that scans the user's walking area. The superior feature extraction power of convolutions combined with the ability of LSTMs in learning temporal dependencies in data offers the possibility to learn the motion dynamics of walking, tracking both legs, and even dealing with challenging cases of leg occlusions, e.g. during turning. We also produce gait state estimates with a simple LSTM that learns a high-level classification of the human gait phases, thanks to a one-hot key encoding of the walking states. Our experimental results demonstrated the improved performance of LTGADnet in the tracking by detection problem with respect to the costly probabilistic state-of-the-art method. While we got sufficient results on the gait analysis problem, we believe that the lack of a bigger dataset hinders our method's performance and that the enhancement of the gait analysis dataset with new data would help achieve an enormous improvement.

Many ideas for improving the work of this thesis have arised, both on the leg tracking and the gait analysis network.

Although the leg tracking presents very high accuracy and excellent occlusion handling, it is unable at the moment to produce the velocities of the leg centers. This could be important information for someone studying and monitoring the patient using the rollator and would be a really useful addition to the leg tracking. A way of achieving this is the addition of a Kalman filter after the leg tracking neural network, which would use the leg centers produced by the previous network to update its state. Also, even though the network already proved to be efficient in cases where a laser sensor with different characteristics was used, it is impossible to be sure of its behavior in any laser sensor. Training on more data deriving from sensors with various area scanning ranges and angular resolutions would secure a predictable behavior of the leg tracking network on any rollator setup, or help achieve an even higher accuracy.

Concerning the gait analysis network, many are the changes that could be implemented towards increasing the network's accuracy and stability. Firstly, capturing more gait analysis data seems inevitable. A variety of patients should be included in the dataset, in order for the network to be able to learn how to distinguish the specific gait patterns for every patient. A different augmentation of the dataset might also help more effectively the training process. As the class imbalance that the data present leads to an imbalanced classification, with the network learning better how to recognize the dominant states LS/RW and RS/LW, it would be useful to break this imbalance by further augmenting only sequences containing the less occurring states LDS and RDS. Moreover, for the training of the gait analysis network the centers produced by the leg tracking network are currently used. This centers are not that smooth as the annotated centers, which though do not exist in the gait analysis dataset, as we do not have a combination of both datasets, but rather separately the annotated centers and gait states. Using as training data annotated centers for the gait analysis network would definitely improve the learning process, so the creation of a unified dataset would be critical. Another interesting idea is the use of a bidirectional LSTM for post-gait analysis, which produces greater accuracy than the simple LSTM, but is not applicable in an online system.

Finally, self-supervised learning methods for both leg tracking and gait analysis would be an excellent solution to alleviate the need for annotations that are admittedly difficult to produce for such a dataset.

Bibliography

[1] Georgia Chalvatzaki, Xanthi S. Papageorgiou, Costas Tzafestas, and Petros Maragos. "Augmented Human State Estimation using Interacting Multiple Model Particle Filters with Probabilistic Data Association". In: *IEEE Robotics and Automation Letters* PP (Jan. 2018), pp. 1–1. DOI: 10.1109/LRA.2018.2800084.

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL, 2014, pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[4] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. "Temporal convolutional networks for action segmentation and detection". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Nov. 2017, pp. 1003–1012. DOI: 10.1109/CVPR.2017.113.

[5] Shaojie Bai, J. Z. Kolter, and V. Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *ArXiv* abs/1803.01271 (2018).

[6] Georgia Chalvatzaki, Xanthi S. Papageorgiou, Costas S. Tzafestas, and Petros Maragos. "Estimating double support in pathological gaits using an HMM-based analyzer for an intelligent robotic walker". In: *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2017, pp. 101–106. DOI: 10.1109/ROMAN.2017.8172287.

[7] Jacquelin Perry and Judith Burnfield. "Gait Analysis: Normal and Pathological Function". In: *Slack Incorporated* (Feb. 2010).

[8] David Levine, Jim Richards, and Michael Whittle. *Whittle's gait analysis*. July 2012. ISBN: 9780702042652.

[9] Hannah Tough, Johannes Siegrist, and Christine Fekete. "Social relationships, mental health and wellbeing in physical disability: A systematic review". In: *BMC Public Health* 17 (May 2017). DOI: 10.1186/s12889-017-4308-6.

[10] Georgia Chalvatzaki, Petros Koutras, Antigoni Tsiami, Costas S. Tzafestas, and Petros Maragos. "i-Walk Intelligent Assessment System: Activity, Mobility, Intention, Communication". In: *Computer Vision - ECCV 2020 Workshops, Proceedings, Part IV*. Vol. 12538. Lecture Notes in Computer Science. 2020, pp. 500–517. DOI: 10.1007/978-3-030-66823-5_30.

[11] K Riel, K Hartholt, Martien Panneman, P Patka, Ed Beeck, and Tischa van der Cammen. "Four-wheeled walker related injuries in older adults in the Netherlands". In: *Injury prevention : journal of the International Society for Child and Adolescent Injury Prevention* 20 (Apr. 2013). DOI: 10.1136/injuryprev-2012-040593.

[12] Veronica Schiariti and Gustavo Pelligra. "171: Developmental Milestones of Assistive Technology: From Wood Walking Sticks to Virtual Reality". In: *Paediatrics Child Health* 20 (June 2015), e95–e96. DOI: 10.1093/pch/20.5.e95b.

[13] Maria Martins, Cristina Santos, Anselmo Frizera, and R. Ceres. "Assistive Mobility Devices Focusing on Smart Walkers: Classification and Review". In: *Robotics and Autonomous Systems* 60 (Apr. 2012), pp. 548–562. DOI: 10.1016/j.robot.2011.11.015.

[14] Silvia Chaparro Cárdenas, Alejandro Lozano-Guzmán, Julian Ramirez, and Antonio Zavala. "A review in gait rehabilitation devices and applied control techniques". In: *Disability and Rehabilitation: Assistive Technology* 13 (Mar. 2018), pp. 1–15. DOI: 10.1080/17483107.2018.1447611.

[15] Sara Bradley and Cameron Hernandez. "Geriatric Assistive Devices". In: *American family physician* 84 (Aug. 2011), pp. 405–11.

[16] Hamid Bateni and Brian Maki. "Assistive devices for balance and mobility: Benefits, demands, and adverse consequences". In: *Archives of physical medicine and rehabilitation* 86 (Feb. 2005), pp. 134–45. DOI: 10.1016/j.apmr.2004.04.023.

[17] Georgia Chalvatzaki, Xanthi S. Papageorgiou, Costas S. Tzafestas, and Petros Maragos. "Comparative experimental validation of human gait tracking algorithms for an intelligent robotic rollator". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6026–6031. DOI: 10.1109/ICRA.2017.7989713.

[18] Marianna Amboni, Paolo Barone, and Jeffrey Hausdorff. "Cognitive Contributions to Gait and Falls: Evidence and Implications". In: *Movement disorders : official journal of the Movement Disorder Society* 28 (Sept. 2013), pp. 1520–33. DOI: 10.1002/mds.25674.

[19] Olivier Beauchet, Gilles Allali, Gilles Berrut, Caroline Hommet, Veronique Dubost, and Fred Assal. "Gait analysis in demented subjects: Interests and perspectives". In: *Neuropsychiatric disease and treatment* 4 (Mar. 2008), pp. 155–60. DOI: 10.2147/NDT.S2070.

[20] Georgia Chalvatzaki, Petros Koutras, Jack Hadfield, Xanthi S. Papageorgiou, Costas Tzafestas, and Petros Maragos. "LSTM-based Network for Human Gait Stability Prediction in an Intelligent Robotic Rollator". In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 4225–4232. DOI: 10.1109/ICRA.2019.8793899.

[21] Christian Werner, Georgia Chalvatzaki, Xanthi S. Papageorgiou, Costas Tzafestas, Juergen Bauer, and Klaus Hauer. "Assessing the concurrent validity of a gait analysis system integrated into a smart walker in older adults with gait impairments". In: *Clinical Rehabilitation* 33 (May 2019), pp. 1682–1687. DOI: 10.1177/0269215519852143.

[22] Yiannis Koumpouros, Thomas Toulias, Costas Tzafestas, and George Moustris. "Assessment of an intelligent robotic rollator implementing navigation assistance in frail seniors". In: *Technology and Disability* 32 (June 2020), pp. 1–19. DOI: 10.3233/TAD-200271.

[23] Frederick Hook, Dale Demonbreun, and Barry Weiss. "Ambulatory Devices for Chronic Gait Disorders in the Elderly". In: *American family physician* 67 (May 2003), pp. 1717–24.

[24] Gerard Lacey, Shane Mac Namara, and Kenneth M. Dawson-Howe. "Personal adaptive mobility aid for the infirm and elderly blind". In: *Assistive Technology and Artificial Intelligence: Applications in Robotics, User Interfaces and Natural Language Processing*. 1998, pp. 211–220. ISBN: 978-3-540-68678-1. DOI: 10.1007/BFb0055980.

[25] Cristina P. Santos. "ASBGo – Multimodal Smart Walker for rehabilitation assistance and clinical evaluation". In: *Journal of Bioengineering & Biomedical Science*. 2016.

[26] Geunho Lee, Eui-Jung Jung, Takanori Ohnuma, Nak Young Chong, and Byung-Ju Yi. "JAIST Robotic Walker control based on a two-layered Kalman filter". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3682–3687. DOI: 10.1109/ICRA.2011.5979784.

[27] Anselmo Frizera, R. Ceres, Eduardo Rocon, and Jose Pons. "Empowering and Assisting Natural Empowering and Assisting Natural Human Mobility: The Simbiosis Walker". In: *International Journal of Advanced Robotic Systems* 8 (Aug. 2011). DOI: 10.5772/10666.

[28] Xanthi S. Papageorgiou, Georgia Chalvatzaki, Athanasios C. Dometios, Costas S. Tzafestas, and Petros Maragos. "Intelligent Assistive Robotic Systems for the Elderly: Two Real-Life Use Cases". In: *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*. 2017, pp. 360–365. DOI: 10.1145/3056540.3076184.

[29] O. Taheri, H. Salarieh, and A. Alasti. "Human Leg Motion Tracking by Fusing IMUs and RGB Camera Data Using Extended Kalman Filter". In: *ArXiv* abs/2011.00574 (2020).

[30] Marina Gavrilova, Faisal Ahmed, Samiul Azam, Padma Polash Paul, Md Wasiur Rahman, Madeena Sultana, and Fatema Tuz Zohra. "Emerging Trends in Security System Design Using the Concept of Social Behavioural Biometrics". In: *Information Fusion for Cyber-Security Analytics*. Vol. 691. Oct. 2017, pp. 229–251. DOI: 10.1007/978-3-319-44257-0_10.

[31] Nicola Bellotto and Huosheng Hu. "Multisensor-Based Human Detection and Tracking for Mobile Service Robots". In: *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* 39 (Feb. 2009), pp. 167–81. DOI: 10.1109/TSMCB.2008.2004050.

[32] M. Martins, A. Frizera, R. Ceres, and C. Santos. "Legs tracking for walker-rehabilitation purposes". In: *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*. 2014, pp. 387–392. DOI: 10.1109/BIOROB.2014.6913807.

[33] K. Arras, Boris Lau, S. Grzonka, M. Luber, Óscar Martínez Mozos, Daniel Meyer-Delius, and W. Burgard. "Range-Based People Detection and Tracking for Socially Enabled Service Robots". In: *Towards Service Robots for Everyday Environments*. 2012.

[34] Angus Leigh, Joelle Pineau, Nicolas Olmedo, and Hong Zhang. "Person tracking and following with 2D laser scanners". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2015 (June 2015), pp. 726–733. DOI: 10.1109/ICRA.2015.7139259.

[35] Xiaoyang Zhao, Zhi Zhu, Mingshan Liu, Chongyu Zhao, Yafei Zhao, Jia Pan, Zheng Wang, and Chuan Wu. "A Smart Robotic Walker With Intelligent Close-Proximity Interaction Capabilities for Elderly Mobility Safety". In: *Frontiers in Neurorobotics* 14 (2020), p. 75. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.575889. URL: https://www.frontiersin.org/article/10.3389/fnbot.2020.575889.

[36] Lucas Beyer, Alexander Hermans, Timm Linder, Kai Arras, and Bastian Leibe. "Deep Person Detection in 2D Range Data". In: *IEEE Robotics and Automation Letters* PP (Apr. 2018). DOI: 10.1109/LRA.2018.2835510.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, pp. 234–241.

[38] Ángel Guerrero-Higueras, Claudia Álvarez-Aparicio, Maria Carmen Olivera, Francisco Rodríguez Lera, Camino Fernández, Francisco Martín, and Vicente Matellán. "Tracking People in a Mobile Robot From 2D LIDAR Scans Using Full Convolutional Neural Networks for Security in Cluttered Environments". In: *Frontiers in Neurorobotics* 12 (Jan. 2019), p. 85. DOI: 10.3389/fnbot.2018.00085.

[39] Walter Pirker and Regina Katzenschlager. "Gait disorders in adults and the elderly: A clinical guide". In: *Wiener klinische Wochenschrift* 129 (Oct. 2016). DOI: 10.1007/s00508-016-1096-4.

[40] Alvaro Muro, Begoña Zapirain, and Amaia Mendez-Zorrilla. "Gait Analysis Methods: An Overview of Wearable and Non-Wearable Systems, Highlighting Clinical Applications". In: *Sensors (Basel, Switzerland)* 14 (Feb. 2014), pp. 3362–94. DOI: 10.3390/s140203362.

[41] Joseph Redmon, S. Divvala, Ross B. Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 779–788.

[42] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. "Online Video Object Detection Using Association LSTM". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2363–2371. DOI: 10.1109/ICCV.2017.257.

[43] Guanghan Ning, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050867.

[44] Cheema Noshaba, Somayeh Hosseini, Janis Sprenger, Erik Herrmann, Han Du, Klaus Fischer, and Philipp Slusallek. "Dilated Temporal Fully-Convolutional Network for Semantic Segmentation of Motion Capture Data". In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*. The Eurographics Association, 2018. ISBN: 978-3-03868-070-3. DOI: 10.2312/sca.20181185.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV 2015)* 1502 (Feb. 2015). DOI: 10.1109/ICCV.2015.123.

[46] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. 2015, pp. 448–456.

[47] I.P.I. Pappas, M.R. Popovic, T. Keller, V. Dietz, and M. Morari. "A reliable gait phase detection system". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 9.2 (2001), pp. 113–125. DOI: 10.1109/7333.928571.

[48] Ciara O'Connor, Susannah Thorpe, Mark O'Malley, and Christopher Vaughan. "Automatic detection of gait events using kinematic data". In: *Gait & posture* 25 (Apr. 2007), pp. 469–74. DOI: 10.1016/j.gaitpost.2006.05.016.