



university of
groningen

faculty of science
and engineering

Advanced Gait Analysis: Integrating Data Processing for Superior Clinical Outcomes

Bachelor's Project Computing Science

July 2024

Author: Amr Abdou

Student Number: S4678753

First supervisor: Dimka Karastoyanova

Second supervisor: Michel Medema

External supervisor: C. Greve

Abstract

Although gait is the most essential form of human locomotion, diagnosing abnormalities within the walking cycle is a cumbersome, long, and obscure process. The diagnosis process renders many patients unable to receive the prompt care and attention needed for an improved quality of life. This project seeks to leverage recent developments in data processing techniques to develop a system that diagnoses gait abnormalities using the data provided from the 3 Dimensional Clinical Gait Assessment (3D CGA) workflow at the University Medical Center Groningen (UMCG) in a fast and highly accurate manner. The system will integrate into a data-processing pipeline that will be used for clinical assessment and decision-making, ultimately aiming to enhance patient care and treatment outcomes at the UMCG. The outcome includes increased diagnostic accuracy, reduced time for patient assessments, and an overall improvement in the efficiency of clinical workflows, leading to better patient outcomes and an enhanced quality of care.

Contents

1	Introduction	5
2	Related Work	7
2.1	Background Literature	7
2.1.1	Gait Stances	7
2.2	Gait Diagnosis Systems: Other Implementations	8
2.2.1	In-Shoe Systems	8
2.2.2	Remote Systems	10
2.3	Current State of the Art: 3D Clinical Gait Assessment	10
2.3.1	Current Workflow	10
2.3.2	Gait Analysis Tool	11
3	Methodology	13
3.1	Requirements	13
3.2	Architecture	14
3.2.1	Frontend	15
3.2.2	Backend	16
3.2.3	Data Processing	16
3.3	Data Flow	17
3.3.1	Data Reader	18
3.3.2	Phase Splitter	18
3.3.3	Data Comparer	19
3.3.4	Diagnosis Generator	19
4	Individual Contribution	24
5	Results and Discussion	25
5.1	Evaluation of Requirements	25
5.2	Comparison to previous work	27
5.3	Limitations	28
6	Conclusion	29
6.1	Future work	29
A	Physical Examination Values	32
B	Screenshots of the New Gait Diagnosis Generator	33
C	Conda Packages	36

List of Figures

1	The phases of gait[11]	7
2	The architecture of the in-shoe system.	9
3	The proposed CNN architecture used by visual-based remote motion-capture systems	10
4	The flow diagram of the old gait software system	12
5	The high-level overview of the new Gait Analysis system	15
6	The flow diagram of the new system	17
7	Login Page	33
8	Gait Analysis Dashboard	34
9	Files uploaded successfully popup message	34
10	The diagnosis of a patient displayed	35

List of Tables

1	The parameters found in the physical examination (LO.xlsx) file alongside their English translation. PROM stands for Passive Range Of Motion. TONUS stands for Muscle Tone. AOC stands for Angle Of Contracture. CLONUS is Involuntary Muscle Contractions. MRC is Manual Muscle Testing. The values exist for both the right side ('Rechts' in Dutch) and the left side ('Links' in Dutch)	32
2	Conda packages used for the project's back end and data processing modules in Python, obtained through running the command <code>conda list</code>	48

Acknowledgments

I want to express my sincere gratitude to Dimka Karastoyanova and Michel Medema for their invaluable guidance, expertise, and support throughout this project. I want to thank Christian Greve for his time in helping us with all our questions, providing us with a visit to the gait lab at UMCG, and above all, for collaborating with us and facilitating the complex language of biology. I also would like to thank my teammates, Emmanouil Kalostypis and Nikola Zivkovic, for their great teamwork and commitment in realizing this project.

1 Introduction

Gait, or the cycle of walking, is one of the most fundamental forms of human movement, yet patients may experience an impaired ability to do so. People of all ages and conditions, such as cerebral palsy and stroke patients, are more than likely to develop impairments with their gait[5, 7, 10]. An impaired gait affects everyday mobility, participation in social activities, and quality of life. If left untreated, the gait abnormalities will lead to worse life outcomes for those affected.

Impactful gait rehabilitation programs rely on accurately diagnosing the underlying biomechanical root cause. Research has been developed to identify core sets of features and their potential underlying impairments[13], which contributes towards the development of the diagnosis process that is in use by hospitals such as the University Medical Center Groningen (UMCG). The current gold standard workflow in diagnosis is the 3D Clinical Gait Assessment (3D CGA)[1]. It provides records of external forces, muscle activity, joint mobility, spasticity, and muscle strength. Clinical professionals must inspect these records and explain them in a report to provide access to rehabilitation programs that will improve the quality of life for patients[8]. However, the workflow takes a long time, specifically the examination of data undertaken by clinical professionals, which takes 2 hours and 35 minutes on average. This workflow can delay access to much-needed rehabilitation programs.

To address such bottlenecks described above, we propose to provide a gait diagnosis tool to the UMCG. This tool is designed to rapidly analyze and detect patient gait impairments, significantly reducing the time required for clinical evaluations. Consequently, this would not only enhance the efficiency of clinicians by freeing up valuable time but also expedite the provision of care to patients, ensuring they receive timely and effective treatment.

To develop an efficient gait diagnosis tool, we have reviewed existing state-of-the-art systems, pointed out their flaws and benefits for each, and merged such insights into a view that best fits the specific needs of the UMCG. The clinical utility of gait diagnosis is well-documented for patients with walking impairments, underscoring the importance of the quality of our work. Our research utilized data provided by the UMCG to develop a more effective system. Our research utilized data provided by the UMCG to develop a more effective system. Integrating these insights enabled us to succeed in building a new, efficient tool for gait diagnosis attuned to the specific needs of the UMCG. This system has the potential to be adopted by other medical clinics and hospitals, thereby improving diagnostic accuracy and efficiency on a broader scale. This leads to the following research questions:

1. What are the specific limitations of current gait diagnosis systems, and how can these be addressed in a new system tailored for UMCG?
2. How can the new system be designed to ensure ease of use and integration into existing clinical workflows at UMCG?
3. How can the integration of state-of-the-art technologies improve the accuracy and speed of gait diagnosis in clinical settings?

These questions, when answered, guide us in building a system that is easy to use and seamlessly integrates into the existing workflows of the UMCG. Additionally, they enable the development of a prototype that serves as a model for future systems in the field of gait diagnosis. The new system is designed to be accurate and quick, ensuring that the quality of patient care is maintained and even enhanced.

The motivation for this research proposal is derived from the significant importance of healthcare, specifically gait abnormalities treatment, considering the many risks associated with receiving untimely healthcare and attention.

From the healthcare provider side, optimizing time allocation can help provide more care for patients. Traditional methods of manually analyzing data utilize a lot of effort from the clinical providers and are prone to human errors that can harm the patient's quality of life. Automating the decision-making process is desired, as it will allow healthcare professionals to streamline their workflow, provide better care and transparency, and avoid the risk of errors.

As technology develops and becomes more advanced, the use cases of innovative systems increase and adapt to specific domains for a multitude of reasons and purposes. By exploring different models and techniques of explanation in the gait diagnosis domain, this project has a significant chance of improving the efficiency of the gait diagnosis workflow. The increase in efficiency allows clinical professionals to provide more timely care, improving patients' quality of life with higher accuracy. The potential impact of this project is significant, as it will potentially eliminate many bottlenecks in the current system at the University Medical Center Groningen (UMCG), resulting in an increased capacity to treat patients. The increased capacity allows many patients to receive care in time to mitigate the health consequences related to delayed care. This project also has the potential to reduce barriers to adopting advanced systems in healthcare and relying on such systems for future diagnostic procedures in other fields, with the goal of placing patient care at the center.

We will start by discussing the related work that has been accomplished in the field of gait diagnosis and analysis in Section 2. Then, we will introduce the methodologies and approaches we followed in Section 3, which includes the architecture and requirements. Section 4 discusses the individual contributions to the project. Section 5 discusses the results of our work, including an evaluation of the system and its limitations. We conclude our thesis in Section 6, discussing future work that can be implemented to improve the new system further.

2 Related Work

As the importance of developing an automated diagnosis for gait impairments grows, other implementations have been conceived that are relevant to what is accomplished by the project. In this section, we start with explaining background literature relevant to this topic. Then, we examine the current state-of-the-art across different implementations, as well as their methodologies, technologies, and outcomes. Additionally, we analyze the strengths and limitations of these methods, highlighting how they have contributed to the current state of the art. Then, we will discuss the current state-of-the-art at the UMCG. This review provides a comprehensive overview of the foundational work that has paved the way for further advancements in the field and sets the context for our contributions to improving diagnostic accuracy and efficiency in gait impairment detection.

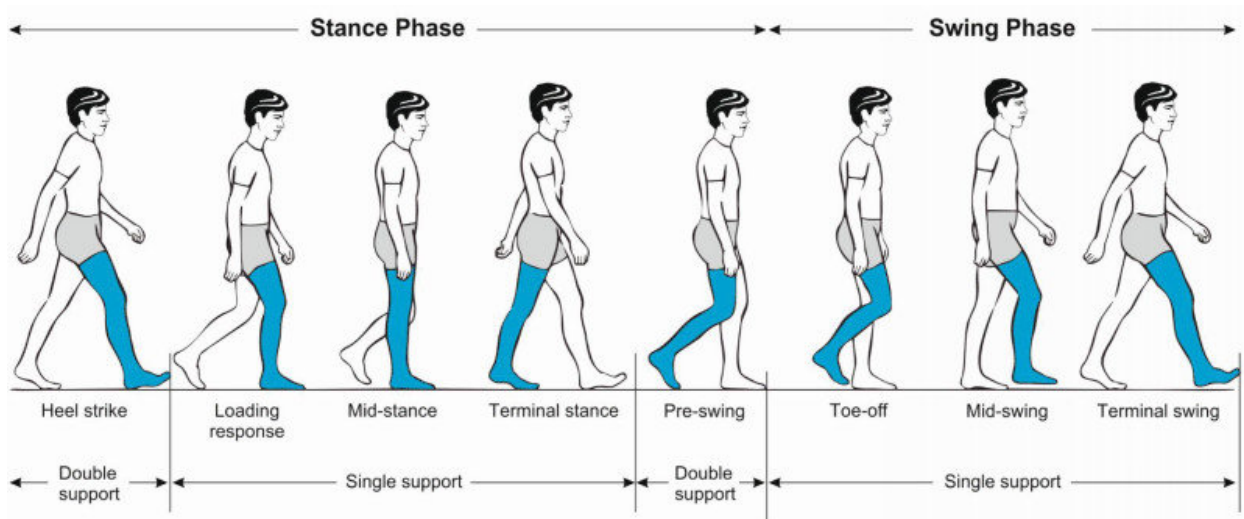


Figure 1: The phases of gait[11]

2.1 BACKGROUND LITERATURE

2.1.1 GAIT STANCES

Gait can be divided into multiple stances or phases, as seen in Figure 1. We will discuss the details of the different gait cycles below [9]:

2.1.1.1 HEEL STRIKE This phase involves the period from the initial contact of the foot with the ground to the immediate response of the body to the initiation of weight transfer. The positions of the joints at this point dictate the pattern of the loading response of the limb. The main objective of this phase is to initiate a stance with a heel rocker and decelerate the impact.

2.1.1.2 LOADING RESPONSE This is the second phase in Figure 1. It follows the initial contact of the foot with the floor and continues until the opposite limb is lifted for the

swing phase. The main goals of this phase are absorbing shock, guaranteeing weight-bearing stability, and maintaining forward progression.

2.1.1.3 MID STANCE This phase starts by lifting the opposite foot off the ground and continues until body weight is aligned over the forefoot. The main objectives include a progression of the swinging foot over the stance foot and ensuring stability of the limb and trunk.

2.1.1.4 TERMINAL STANCE This phase begins with the heel rise and continues until the next contact with the ground by the opposite foot. During the phase, body weight shifts forward over the forefoot. The main objectives are the progression of the body forward over the supporting foot and maintenance of limb and trunk stability.

2.1.1.5 PRE-SWING This final phase of the stance, known as the terminal double stance interval in the gait cycle, starts when the opposite foot touches the ground and ends when the same-side foot lifts off. Sometimes referred to as weight release or weight transfer, this phase is crucial for advancing forward. As body weight shifts quickly, the trailing leg helps propel the body forward and gets ready for the swing phase. The main objectives are to position the limb for the swing phase and to accelerate forward progression. We estimate this phase to be the opposite foot's loading response

2.1.1.6 TOE-OFF The first third of the swing period is the toe-off phase, also known as the initial swing phase, and it runs from the beginning of the toe-off until the swinging foot is opposite the stance foot. The two key objectives are to ensure that the foot is cleared from the ground and to progress the limb from its trailing position.

2.1.1.7 MID-SWING The middle third of the swing period initiation starts when the limb is in line with the stance extremity and continues until the limb advances to the point where the tibia is perpendicular and has hip and knee about the same degree of flexion. The major goals of this movement are to move the limb forward and to ensure the foot will clear the ground. We simplify this phase to be computed as a mid stance in the opposite foot.

2.1.1.8 TERMINAL SWING The final phase of the swing period, from when the tibia is vertical until the foot contacts the ground, the limb advancement is completed as the shank moves ahead of the thigh. The main objectives would be to finalize limb advancement and set the limb for the stance phase. We also estimate this phase as being in the terminal stance for the other foot.

2.2 GAIT DIAGNOSIS SYSTEMS: OTHER IMPLEMENTATIONS

2.2.1 IN-SHOE SYSTEMS

A study published in 2021 made use of 16 dedicated porous graphene-SBR (styrene-butadiene rubber) pressure sensors (PGSPSs) placed inside shoes to allow continuous, day-to-day moni-

toring of gait signals, which include walking, running, and jumping [6]. These sensors proved to be highly accurate and flexible, which enabled comfort for users. The sensors could wirelessly transmit the data to smartphones for real-time gait analysis and monitoring. Figure 2 illustrates that the smart sole system consists of 3 parts: a PGSPS-based sensor system with a circuit board (FPC), a Raspberry PI (RPI) system, a data acquisition (DAQ) circuit, and a printed circuit board (FPC), a Raspberry PI (RPI) system, a data acquisition (DAQ) circuit, data processing, and displaying software.

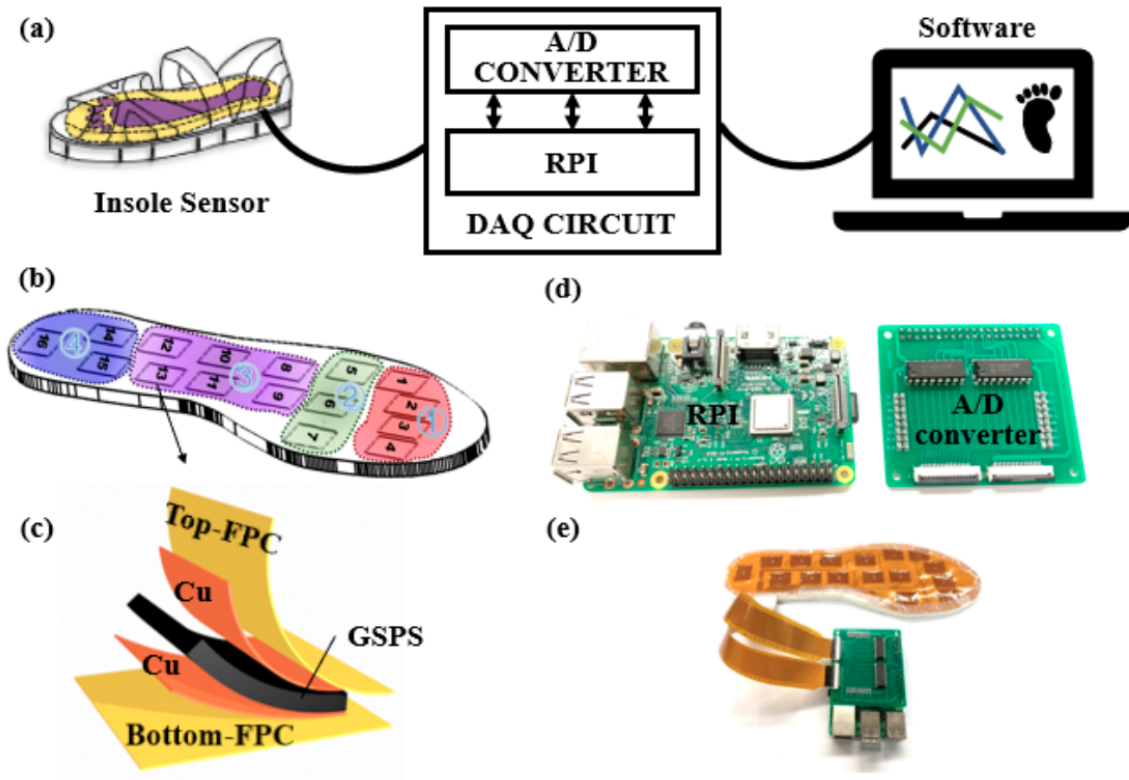


Figure 2: The architecture of the in-shoe system.

While the pressure sensors allow the system to gather detailed plantar pressure data during various gait activities, it does not integrate data from physical examinations performed by medical specialists. This means the system may not be able to provide a complete picture of the entire gait cycle, which could be important for certain diagnostic or rehabilitation applications. An area for potential improvement would be to explore ways to incorporate data from clinical gait analysis, such as kinematic and kinetic measurements, to complement the pressure sensor data. This could help create a more holistic understanding of the user's gait patterns and potentially enhance the system's capabilities for applications like gait disorder diagnosis and rehabilitation monitoring. Overall, the paper demonstrates an innovative wearable gait analysis system with many promising features. However, further development to capture the full gait cycle, potentially by integrating with clinical assessment methods, could help expand the system's usefulness and reliability for certain medical and rehabilitation use cases.

2.2.2 REMOTE SYSTEMS

Other systems, such as the system proposed in this paper published in 2021, rely on vision-based motion capture techniques that can be used remotely in unconstrained environments such as homes or clinics.[4] The analysis is done through the upload into a server and the execution of a web service that classifies their gait.

Using a shallow Convolutional Neural Network (CNN) architecture, The novel gait-type classification system achieves an accuracy of over 90% on a limited dataset. It demonstrates significant generalization ability, correctly associating available gait types with corresponding pathologies, even when tested on a relatively noisy dataset captured in a home setting. The CNN architecture is highlighted in Figure 3. The limitations, however, involve the limited dataset being used to train the CNN, which may result in overfitting and a lack of robustness when applied to real-world data. Also, it doesn't incorporate muscle activity or physical examinations performed by medical professionals into the data, hence failing to capture fully the nuances of the gait cycle.

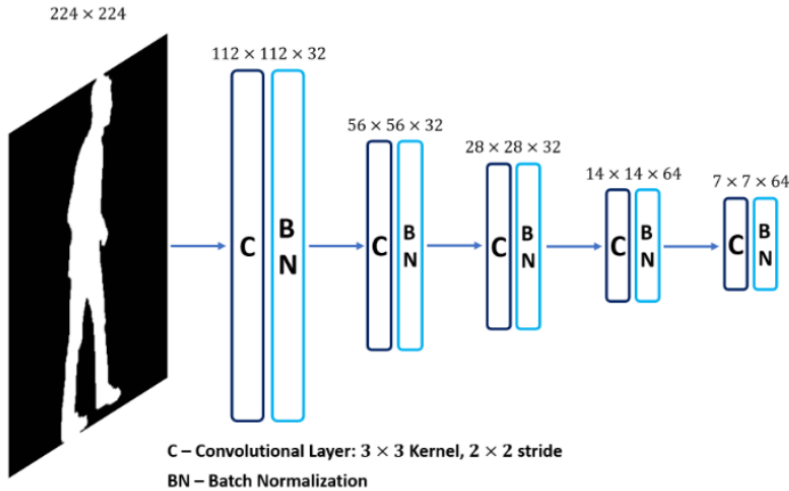


Figure 3: The proposed CNN architecture used by visual-based remote motion-capture systems

2.3 CURRENT STATE OF THE ART: 3D CLINICAL GAIT ASSESSMENT

2.3.1 CURRENT WORKFLOW

The current state of the art for gait diagnosis in the UMCG using 3-Dimensional Clinical Gait Analysis (3D CGA) is defined as the following workflow[1]. The workflow begins with the first step: a three-dimensional motion capture analysis, performed in a laboratory setting to record measurements of external forces and muscle activity (electromyography), a physical assessment, and joint angles and coordinates which takes on average 1 hour and 30 minutes

to complete. The physical examination captures data about joint mobility, spasticity, and muscle strength. The full collection of physical examination parameters, which is written in Dutch originally and translated to English, can be found in the appendix. These joint angles and coordinates are measured in all 3 planes of motion which include the frontal, sagittal or lateral, and transversal planes. Next, we have the second step, which is the expert interpretation of clinicians, where they compare the collected data mentioned above to conventional data from healthy individuals at every specific stage of the gait cycle. A gait deficiency is labeled if any of the thresholds of the healthy gait parameters have been exceeded. Then, we move onto the third step, where the labeled gait deficiency is used, alongside the compiled physical assessments of joint mobility, spasticity, and muscle strength and muscle activation analysis, to diagnose the underlying root cause of the gait deficiency. Steps 2 and 3 combined take an hour and 45 minutes to complete. After the interpretation process, the fourth step involves the creation of a diagnostic report, followed by incorporating the diagnostic report into the electronic patient dossier (EPD), requiring 50 minutes to complete. Finally, the concluding treatment plan is discussed at the Medical Doctor's Office (MDO) during a 20-minute appointment. The workflow defined above is tedious and extensive, taking 4 hours and 25 minutes. Some steps in the workflow above can be accelerated and digitized, which are steps 2, 3, and 4. We will refer to these steps as the variable components. Steps 1 and 5 are the invariable components in the workflow since they require human input and intervention, which cannot be optimized.

2.3.2 GAIT ANALYSIS TOOL

A proposed solution that entails a software system has been conceptualized to enhance the speed of the variable components in the workflow[3]. After multiple iterations, the current software system with its data flow diagram is seen in Figure 4.

The software system receives hard-coded values, representing the threshold values of the various physical assessment parameters discussed above. Alongside the physical assessment values obtained, the video, 3D motion capture of joint motion, joint load, and muscle activation source files (c3d file) of the recorded motion capture done in the first step of the workflow of the 3D CGA, and the physical examination(LO.xlsx) are included as input, where they are then processed appropriately. The software system then computes the mid-stance, which is when the distance between both knees is closest to the distance between both ankles from the processed information. Next, the peaks of the loaded filtered signals are computed and validated by human input that verifies the correctness of the peaks and corrects the chosen peaks if necessary. The verified and corrected peaks are then processed and compared to the threshold values to evaluate whether the limit has exceeded any parameters. Finally, a report with the results of the evaluation is generated and displayed to the user through the graphical user interface (GUI). These results are in the form of images from the input videos of when a certain parameter is abnormal as well as the evaluation results of that parameter. It is important to note that all the analysis is done in the lateral plane of motion.

The system attempts to speed up the variable components of the workflow of the 3D CGA, however, its reliance on user input can introduce unnecessary delays to the workflow of the

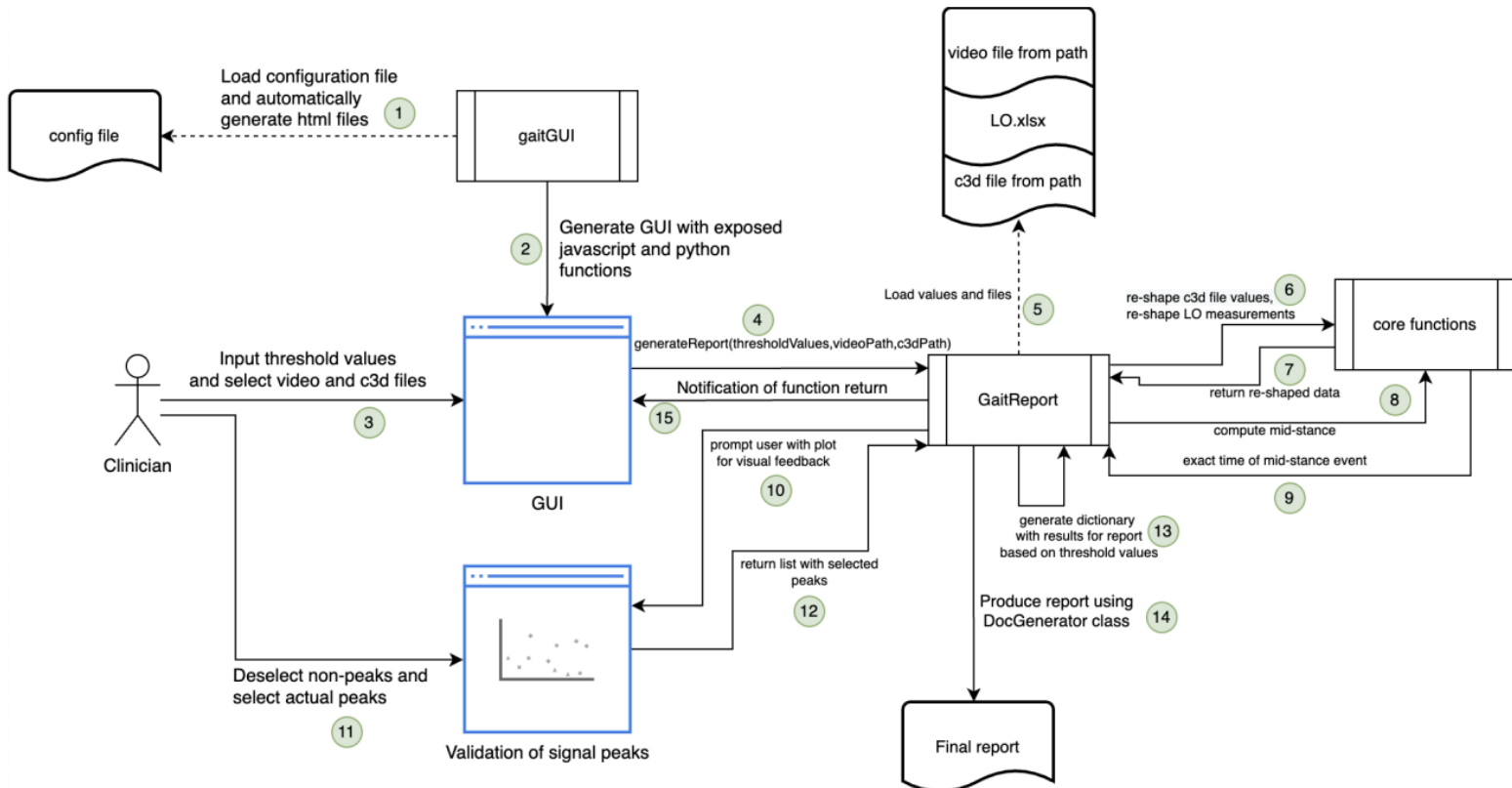


Figure 4: The flow diagram of the old gait software system

3D CGA. Furthermore, the system's rigidity hampers its scalability and extendibility, diminishing care quality due to a lack of versatility. Moreover, there are unresolved functionalities and missing features, impeding the full utilization of the system. On the other hand, the system utilized important decision trees for detecting gait impairments, developed in collaboration with medical professionals at the UMCG, which will be needed to build a new system with similar capabilities.

This project aims to improve and automate the current techniques used in the current gait diagnosis system. As mentioned previously, the current system that is being used lacks in structure, functionality, and ease of use. Therefore, the system can not be used in clinical settings, and it is difficult to restructure the code-base due to the aforementioned lack of structure in both architecture and code-base. In the upcoming section, we will discuss the methodology that enabled us to produce our new system.

3 Methodology

For this project, we have been provided with a dataset of .c3d files containing 3D motion capture data of patients and .xlsx files containing the corresponding physical examination data from the UMCG. Additionally, we were given the previous version of the Gait Analysis tool, discussed in Section 2.3.2, to investigate the reusable components for the implementation and architecture of the new system. Following this investigation, we identified key reusable components and set out to develop a new system composed of multiple components to enhance modularity and scalability [2]. This approach ensures that each component can be developed, tested, and maintained independently, simplifying the overall development process and facilitating easier updates and improvements in the future. The decision to adopt a more modular and scalable design was driven by the need for easier maintenance, enhanced performance, and the flexible integration of new features and technologies. Once the requisite frameworks were thoroughly researched and validated for the construction of our system, we uploaded the .c3d and .xlsx files from the UMCG dataset to validate the system's results and effectiveness. The results were validated against the expert interpretation of our external supervisor, C. Greve, due to his medical expertise. We begin by laying out the requirements of the new system, followed by a discussion of the system's architecture. Subsequently, we provide an in-depth examination of the data processing module, including its functionality and the specific algorithms employed to ensure efficient and accurate data handling.

3.1 REQUIREMENTS

The development of a gait diagnosis system aims to enhance the evaluation and treatment of patients with gait abnormalities by providing a comprehensive and automated diagnostic tool. This system is designed to integrate various components, including a user-friendly front-end interface, robust back-end services, and advanced data processing models. The following requirements outline the essential features and functionalities necessary to achieve a reliable, efficient, and scalable gait diagnosis system.

Each requirement has a unique ID, denoted as $[R][\#]-[F/NF]$ where:

- **R**: Requirement
- **#**: the number of the requirement
- **F/NF**: meaning functional/non-functional

The requirements are:

- **[R-1-F]**: Secure login portal where clinicians can enter their credentials to access the system.
- **[R-2-F]**: Upload file of a 3D motion capture of patient walk for diagnosis.
- **[R-1-NF]**: The file of a 3D motion capture of patient walk has to be in .c3d format.

- **[R-3-F]**: Upload file of a patient’s physical examination for diagnosis.
- **[R-2-NF]**: The file of a patient’s physical examination has to be in .xlsx format.
- **[R-4-F]**: Produce a diagnosis of the patient
- **[R-3-NF]**: The diagnosis should include gait impairments based on joint kinematics.
- **[R-4-NF]**: The diagnosis should include relevant values from the physical examination.
- **[R-5-NF]**: The diagnosis should include gait impairments based on the muscle activity.
- **[R-6-NF]**: The diagnosis should include gait impairments based on the ground reaction forces.
- **[R-7-NF]**: The diagnosis should delivers rapid and accurate diagnoses within 5 seconds.
- **[R-5-F]**: Provide an executable to initialize and configure the entire system, ensuring that all components and services are properly set up and ready for operation.
- **[R-8-NF]**: The executable should run on the Windows operating system.
- **[R-6-F]**: Display the diagnosis to the clinicians.
- **[R-9-NF]**: The diagnosis displayed should show what are the gait impairments detected and when have they been detected.
- **[R-10-NF]**: The diagnosis is displayed in tables.
- **[R-7-F]**: Export the diagnosis.
- **[R-11-NF]**: The diagnosis should be exportable to PDF format.

Based on these requirements, we have developed an architecture that will accommodate for all the requirements, which we will detail in the following section.

3.2 ARCHITECTURE

The architecture of the new system can be seen in Figure 5. We have 3 main components: the Front end, the Data Processing, and the Back end. We will discuss each component, with their subcomponents in this section.

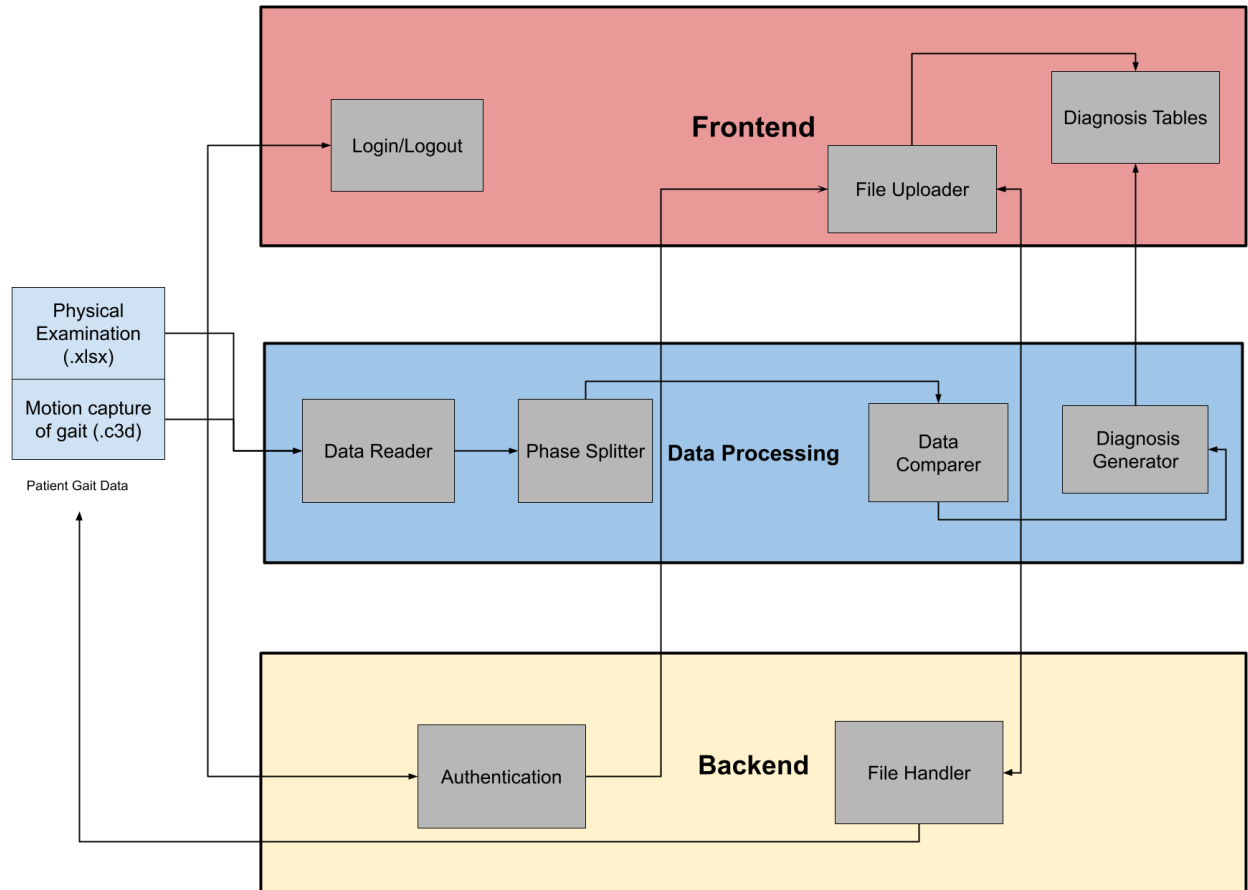


Figure 5: The high-level overview of the new Gait Analysis system

3.2.1 FRONTEND

The front-end is the main user interface (UI) that the medical professionals interact with to obtain results. It consists of a Login component, where medical professionals can securely add their login credentials to access the system. Then, they can upload both .c3d files and LO.xlsx corresponding to the current patient being diagnosed. Then, these files are sent to the Backend component which will continue the data processing procedure. Finally, after the diagnosis is complete, the Diagnosis Tables, consisting of the relevant gait impairments and their relevant LO.xlsx data, are rendered and displayed to the user for further analysis. The front end of the application is developed using the React framework, which is based on JavaScript. React's component-based architecture facilitates the creation of reusable UI components, promoting consistency and maintainability throughout the application. Its efficient updating and rendering of components in response to data changes are vital for developing a responsive and interactive user interface for the gait diagnosis system. React also integrates effectively with Flask, enabling seamless communication between the front-end and back-end through RESTful APIs. This compatibility ensures a smooth data flow and interaction between client-side and server-side components. Additionally, React's robust

developer tools and active ecosystem offer a wide array of libraries, extensions, and plugins, empowering us to build a highly functional and user-friendly interface.

3.2.2 BACKEND

The back-end handles all user requests from the front-end as well as houses the Data Processing module which can detect gait impairments by extracting and processing data from the .c3d and .xlsx files uploaded. It can authenticate users who are willing to use the application and give the users access to upload the aforementioned files above for the generation of the diagnosis. The back-end then stores the files in a directory for the diagnosis as part of the handling process. It then sends the relevant gait impairments and their associated LO.xlsx data to the front-end to be displayed for the user. For the back-end, we chose Flask, a Python-based web framework. The rationale for this decision is multifaceted: since our data processing is managed in Python, using Flask ensures seamless integration and minimizes the overhead associated with context switching between different programming languages. Flask is designed to be simple and flexible, providing the essentials without enforcing a specific project structure, making it an ideal solution for our custom gait diagnosis system. Flask also supports various extensions that can add necessary functionality, such as database integration, form validation, and authentication, facilitating the development of a scalable and maintainable application. Its lightweight nature allows for fine-tuned control over the application, which is crucial for ensuring that the back-end can efficiently handle real-time data processing demands.

3.2.3 DATA PROCESSING

Finally, the data processing component is integrated into the back-end, but its complexity and scope warrant it being considered a distinct module. It reads the .c3d and .xlsx files uploaded through the front-end and stored by the back-end. The data is then processed and split into the different gait phases, and then gait phases are established. Additionally, the system processes a pre-stored .c3d file, designated as the "ideal" data collection, which serves as a benchmark for comparison against other .c3d data. This comparative analysis is integral to identifying deviations and detecting gait impairments accurately. From there, the identified deviations and gait impairments are collected and sent, alongside their relevant LO.xlsx data which a medical professional would look at to further develop an understanding of the patient's case at hand into 2 separate tables. These 2 tables are then sent back to the back-end, which sends it back to the front-end for display. The data processing module was developed exclusively in Python, leveraging the extensive range of libraries that facilitate efficient data processing. Additionally, Python was chosen to ensure seamless integration with the back-end of the application, which is implemented using the Flask framework.

It is important to know that the package manager used for both modules written in Python is Conda. Conda is an open-source package management and environment management system that runs on Windows, macOS, and Linux[12]. It allows users to quickly install, run, and update packages and their dependencies, as well as manage multiple environments with different versions of Python and other libraries. This is particularly useful for ensuring

compatibility and reproducibility in scientific computing and data science projects. One of the important libraries available through Conda is `ezc3d`, which is essential for working with C3D files in biomechanics and gait analysis. The full list of libraries used can be found in the appendix.

3.3 DATA FLOW

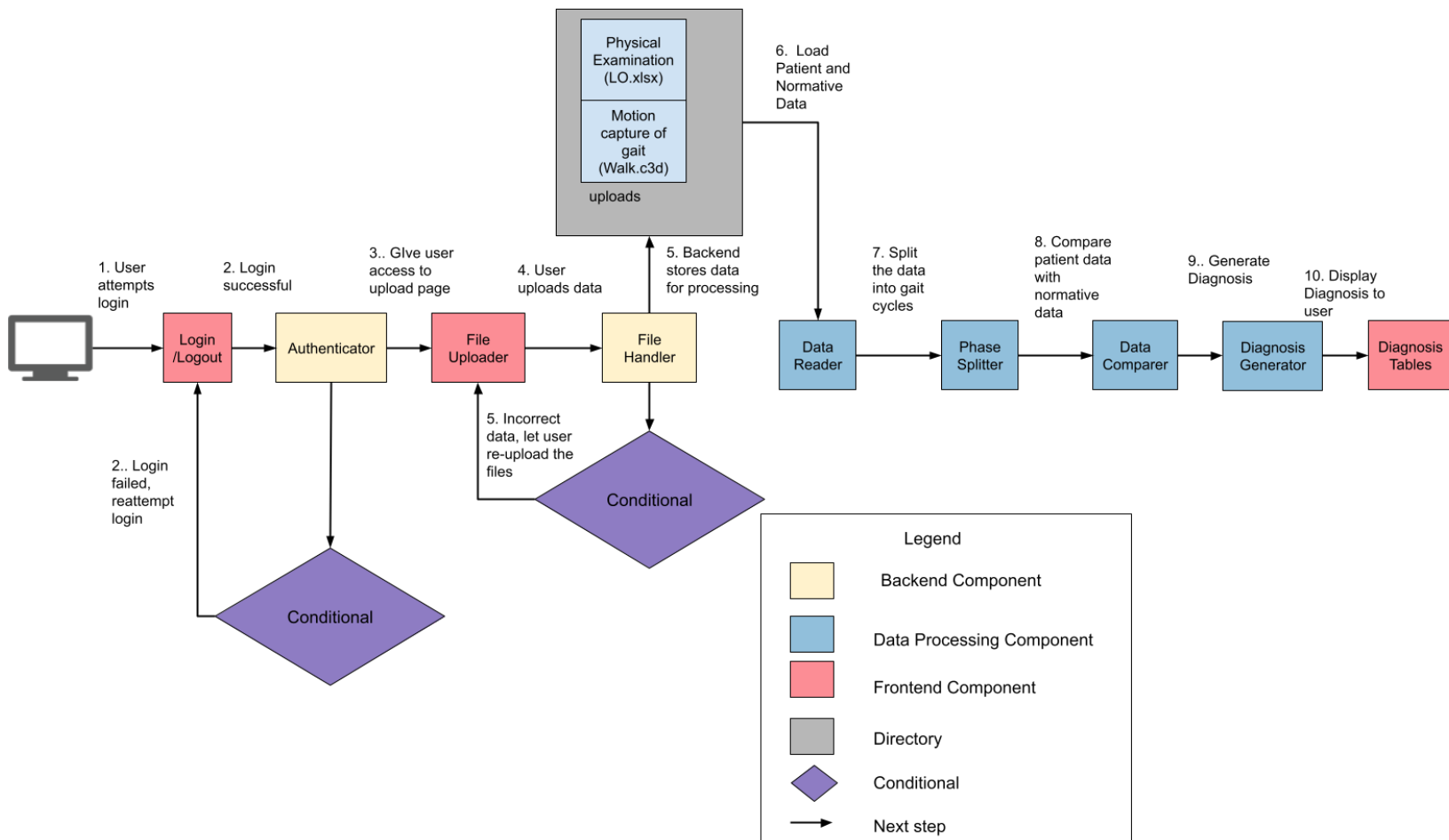


Figure 6: The flow diagram of the new system

In this section, we will conduct an in-depth examination of the data flow within our system, with a particular emphasis on the data processing components. An illustration can be seen in Figure 6. Given that the primary focus of this thesis is on these components, we will provide a comprehensive analysis of their functionality and significance. The discussion will encompass the mechanisms of data ingestion, transformation, storage, and retrieval, highlighting the intricate processes that ensure data integrity and efficiency. This detailed exploration will not only elucidate the technical intricacies of our system but also demonstrate the critical role of data processing in achieving our research objectives.

The user first attempts to log into our system through the Login subcomponent. These credentials will be passed on to the Authenticator subcomponent for validation, and if successful, will allow access to the next part where the user can upload both the LO.xlsx and Walk.c3d files for analysis. Otherwise, the login attempt has failed and they will have to login again and use the correct credentials. Then, the user will upload the necessary files mentioned above, and if successful, will be passed on to the File Handler subcomponent for storing in the `uploads` directory. If the incorrect file types have been uploaded, an error will appear and the user will have to resubmit the correct file(s). Then, the system will be able to start the analysis process, which we will detail deeply in the following overview.

3.3.1 DATA READER

The data reader component reads the Walk.c3d files of both the patient as well as the normative gait stored within the system, using the `ezc3d` library. Then, it processes the the Walk.c3d files by extracting the labels and their associated values, the number of frames and the number of labels, the frame rate, the first frame when the gait has commenced, the joint angles of both sides in the sagittal plane, the coordinates of every indicator at every frame, and the frames where a Foot Off or a Foot Strike has occurred. It then combines the coordinates of every indicator and their associated indicator under one `pandas` data frame. From there it moves onto the Phase Splitter subcomponent, which can extract the rest of the phases.

3.3.2 PHASE SPLITTER

The phase splitter takes the data frame of coordinates and indicators and the frame rate and computes the missing stances that were not given through the Walk.c3d files of both the patient and the norm. Those stances are Loading Response, Mid Stance, and Terminal Stance. We compute the above stance using the following heuristics, based on the observations of the external supervisor C. Greve. For the midstance, we start by finding out the possible ranges of when a midstance may occur, for both sides. For the left midstance, we would be looking between the initial contact of the left foot with the ground and the left foot off the ground events, which we extract from the list of frames where a Foot Off or a Foot Strike occurs. We also compute the range for the right midstance, which is between the right foot's initial contact and the right foot being off the ground. After we calculate those ranges, we then compute the minimum difference $d1-d2$ where $d1$ is the distance between the left knee and the left ankle for the right foot and between the right knee and the right ankle for the second foot. $D2$ is the distance between the knees. Once we establish the minimum distance, we can extract the frame where it happens, which is what is returned by this function. The pseudo-code for the midstance calculation can be found in Algorithm 1. As for the loading response, we estimate it to occur 10 frames before the opposite foot leaves the ground, which would mean that a right loading response happens 10 frames before the left foot leaves the ground, and a left loading response will be 10 frames before the right foot leaves the ground. Finally, we compute the terminal stance similarly to the loading response, where we also estimate it. However, we estimate it as 10 frames before the opposite foot strikes the ground. A right terminal stance would happen 10 frames before the left foot strikes the ground, and

a left terminal stance would happen 10 frames before the right foot strikes the ground. After calculating all the stances, we add them to a list of global events, which contains the event, the context, and the frame where it happens. Then, we move on to the data comparer.

3.3.3 DATA COMPARER

This subcomponent focuses on comparing the joint angles of the patient and norm, which are the hip, knee, and ankle joints, for both sides, at the frames of when the gait cycles defined above occur. The joint angles of the patient walk are compared to the norm walk, and if it exceeds a specified threshold value, whether being above the threshold or more, or being below the threshold or less, then we mark a gait deficiency as either positive if it is above, or negative if it is below. Otherwise, we mark there to be no specific gait deficiencies detected, and we move on to the final part, which is generating the diagnosis. We also append the joint angle value, as it will be needed in the diagnosis.

3.3.4 DIAGNOSIS GENERATOR

This subcomponent focuses on identifying the gait impairments, based on the joint angle and the gait phase it occurs in. Based on specific deficiencies that have been detected, this subcomponent would also append to another table specific values from the LO.xlsx file, which is the physical examination, that medical professionals would look at to get a full picture of the gait deficiency. It is important to note that we extract the LO values of the examined side since the parameters exist for both the left and right sides. The decisions that the algorithm makes to come up with these diagnoses have been developed in consultation with the external supervisor C. Greve. We will detail exactly what we look for in each stance and joint angle difference combinations.

3.3.4.1 FOOT STRIKE In a foot strike, we lookout for the following gait deficiencies: Plantar flexion, Increased Knee Flexion, Decreased Knee flexion (other foot), Increased Hip Flexion, and Decreased Hip Flexion. Plantar flexion is identified when the patient fails to meet the designated threshold in the ankle joint of the examined side. Specifically, plantar flexion of the right foot is only detected during the occurrence of a right foot strike, and similarly, plantar flexion of the left foot is only identified during a left foot strike. We extract the following LO values based on the above finding for the specific foot: DorsiflexieMRC, DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, DorsiflexiegestrektAOC. We also append the ankle joint range of motion in degrees.

Increased knee flexion is detected when the patient has exceeded the norm by the threshold or more in the knee joint, and we extract the following LO values for the specific foot: Pop-hoekPROM, Pop-hoekAOC, Knie-extensiePROM. We add the knee joint range of motion in degrees here.

We also detect decreased knee flexion in the other side not being currently examined, which happens when the patient did not meet the norm by more than the threshold in the other knee joint. We extract the following LO values: Duncan-ElyPROM, Duncan-ElyAOC. We

Algorithm 1 Find MidStance

```

Require: markersWithLabels:           ▷ DataFrame of coordinates and labels
Require: Fs_MarkerPositions:         ▷ Frame rate
Require: times:                       ▷ Array of event times
Require: contexts:                   ▷ Array of event contexts
Require: events:                     ▷ Array of event types (FootOff or FootStrike)
1: evnt_ic_right, evnt_ic_left = ∞;
2: evnt_footoff_right, evnt_footoff_left = -1;
3: for each event in events do
4:   if (evnt_ic_right is ∞) and (event is 'Foot Strike') and (context is 'Right') then
5:     evnt_ic_right = corresponding time           ▷ Set initial contact for right foot
6:   else if (evnt_footoff_right is -1) and (event is 'Foot Off') and (context is 'Right')
   and (time > evnt_ic_right) then
7:     evnt_footoff_right = corresponding time     ▷ Set foot off for right foot after
   initial contact
8:   else if (evnt_ic_left is ∞) and (event is 'Foot Strike') and (context is 'Left') then
9:     evnt_ic_left = corresponding time           ▷ Set initial contact for left foot
10:  else if (evnt_footoff_left is -1) and (event is 'Foot Off') and (context is 'Left')
   and (time > evnt_ic_left) then
11:    evnt_footoff_left = corresponding time ▷ Set foot off for left foot after initial
   contact
12:  end if
13: end for
14: evnt_ic_right, evnt_footoff_right *= Fs_MarkerPositions
15: evnt_ic_left, evnt_footoff_left *= Fs_MarkerPositions
16: min_diff_right, min_diff_left = ∞
17: for frame from event_ic_right to event_footoff_right do
18:   Calculate  $d1\_right = \text{abs}(LKNE[frame, 2] - LANK[frame, 2])$            ▷ Distance left
   knee to left ankle
19:   Calculate  $d2\_right = \text{abs}(RKNE[frame, 2] - LKNE[frame, 2])$            ▷ Horizontal
   distance between knees
20:   Calculate  $\text{diff\_right} = |d1\_right - d2\_right|$ 
21:   if  $\text{diff\_right} < \text{min\_diff\_right}$  then
22:      $\text{min\_diff\_right} = \text{diff\_right}$ ;  $\text{midStance\_right} = \text{frame}$ 
23:   end if
24: end for
25: for frame from event_ic_left to event_footoff_left do
26:   Calculate  $d1\_left = \text{abs}(RKNE[frame, 2] - RANK[frame, 2])$            ▷ Distance right
   knee to right ankle
27:   Calculate  $d2\_left = \text{abs}(RKNE[frame, 2] - LKNE[frame, 2])$            ▷ Horizontal
   distance between knees
28:   Calculate  $\text{diff\_left} = |d1\_left - d2\_left|$ 
29:   if  $\text{diff\_left} < \text{min\_diff\_left}$  then
30:      $\text{min\_diff\_left} = \text{diff\_left}$ ;  $\text{midStance\_left} = \text{frame}$ 
31:   end if
32: end for
   return  $\text{midStance\_right}, \text{midStance\_left}$ 

```

also add the knee joint range of motion in degrees here.

Increased and Decreased hip flexion are detected when the patient has exceeded or has not met the threshold respectively. There are no associated LO values that are looked at when this occurs, and this is only detected for the examined side.

3.3.4.2 LOADING RESPONSE In a loading response, we look out for the following in the examined side: No/decreased plantar flexion, increased knee flexion, decreased knee flexion, knee hyperextension, and increased/decreased hip flexion. There are no LO values that are inspected when there are gait deficiencies labeled during this phase.

We detect no/decreased plantar flexion if the following conditions are met: there is no plantar flexion detected in the foot strike, and the patient's walk has not met the norm by a margin of the threshold in the ankle joint.

Increased knee flexion is labeled when the patient exceeds the norm by the threshold in the knee joint. The knee joint range of motion in degrees is also added. Decreased knee flexion and knee hyperextension are both detected when the patient's walk lags behind the norm by the threshold or more. Here, we rely on the actual degrees of the knee joint that we appended in the Data Comparer section. If the knee joint angle degrees are positive or zero, then we label the gait deficiency that is labeled is decreased knee flexion, otherwise we detect knee hyperextension. We also add the knee joint range of motion in degrees.

As for the decreased and increased hip flexion, we follow the same detection done in the foot strike.

3.3.4.3 MID STANCE In the mid-stance, we look out for the following gait deficiencies in the examined side: plantar flexion, increased dorsiflexion, increased knee flexion, decreased knee flexion, knee hyperextension, increased hip flexion, and decreased hip flexion. As for the other foot, we also detect plantar flexion.

For plantar flexion in the ankle joint of both feet, we check if the patient's walk has not met the norm by the threshold. Only when the non-examined foot has a label of plantar flexion do we add the following LO values: DorsiflexieMRC, DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, and DorsiflexiegestrektAOC. We also add the ankle range of motion in degrees.

If the examined foot exceeds the given threshold, then we add the increased dorsiflexion label. Otherwise, if the examined foot has no gait deficiencies, then we display the range of motion in degrees of the ankle joint. As for the knee joint on the examined side, we follow a similar detection procedure as the one displayed in the loading response.

We detect increased knee flexion if we exceed the norm by the threshold, and we display the following LO values: Knie-extensiePROM and HeupextensiePROM. As for both increased

knee flexion and knee hyper extension, we detect either of them if the patient has not met the norm. In contrast to the loading response detection, we use a different threshold, named **knee_midstance_threshold** in the program, which we use specifically for this portion of the detection. If the patient's walk is higher than the **knee_midstance_threshold**, we detect decreased knee flexion, otherwise, we detect knee hyper extension. For the decreased knee flexion, we add the following LO values: DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, and DorsiflexiegestrektAOC. For all the knee deficiencies, we add the range of motion of the knee joint to the diagnosis.

Increased and decreased hip flexion in the examined foot are detected in the same manner as the previous two phases, but in this phase, we add the following LO values for increased hip flexion: Knie-extensiePROM and HeupextensiePROM.

3.3.4.4 TERMINAL STANCE Finally, in the terminal stance, we detect the following gait impairments for the examined foot: decreased dorsiflexion, increased dorsiflexion, increased knee flexion, knee hyperextension, and no hip extension. As for the other foot, we can detect plantar flexion only.

Decreased dorsiflexion is detected when the patient's walk falls short of the norm in the ankle joint, and the following LO values are relevant to the diagnosis: DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, and DorsiflexiegestrektAOC. We add the ankle range of motion of the examined foot in degrees to the diagnosis.

As for increased dorsiflexion, the patient's walk goes beyond the threshold. There are no associated LO values that are captured here. Similarly to decreased dorsiflexion, we also add the ankle range of motion in degrees to the diagnosis. Otherwise, the patient's walk is within the norm, and we add the ankle joint range of motion in degrees of the examined foot.

As for the other foot, we detect plantar flexion if the patient's walk falls behind the norm. We also add the actual ankle joint range of motion in degrees here, and we add the following LO values: DorsiflexieMRC, DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, and DorsiflexiegestrektAOC.

Increased knee flexion and knee hyperextension occur when the patient's walk has exceeded or fell short of the norm respectively. The associated LO values for increased knee flexion are Knie-extensiePROM and HeupextensiePROM. We also add the knee joint range of motion in degrees. The associated LO values for knee hyperextension are DorsiflexiegebogenPROM, DorsiflexiegebogenAOC, DorsiflexiegestrektPROM, and DorsiflexiegestrektAOC. We also include the knee joint range of motion in degrees.

Finally, we have no hip extension, which we detect if we have exceeded the threshold. We add the following LO values to the diagnosis: Knie-extensiePROM, HeupextensiePROM, and HeupextensieMRC.

The code that includes the decision trees can be found in the appendix. After the diagnosis process is completed, the gait deficiencies have been labeled, and the LO values have been extracted, we collect the results of the diagnosis and the LO values in 2 separate tables, and the LO table is then formatted to remove duplicate entries. The program then returns those two tables to the back-end, which is sent to the front end for display.

4 Individual Contribution

For clarity, the new gait system was undertaken by 2 other RUG students, Emmanouil Kalostypis and Nikola Zivkovic, alongside myself. We decided to go for an even split, where Nikola was responsible for the front-end completely. Nikola also contributed to some endpoints in the back-end. Emmanouil completed the rest of the back end endpoints, as well as working alongside myself to complete the Data Reader, Phase Splitter, and Data Comparer subcomponents. The Diagnosis Generator was done completely by myself. We made an effort to find an even split of the work as much as possible, avoiding much disparity in the workload.

5 Results and Discussion

This section outlines the requirements or objectives that were successfully met during the project. Each requirement should include a brief description and how it was achieved. We will also discuss some limitations of the new system. We will rewrite the requirements and detail them one by one.

5.1 EVALUATION OF REQUIREMENTS

[R-1-F]: Secure login portal where clinicians can enter their credentials to access the system. We were able to achieve this requirement through the Login/Logout and the Authenticator subcomponents, which allowed the clinicians to input their credentials into the system, and allow access only if the credentials were correct.

[R-2-F]: Upload file of a 3D motion capture of patient walk for diagnosis. We were able to achieve this requirement through the File Uploader and File Handler subcomponents, which handle the files being uploaded and store them for further processing.

[R-1-NF]: The file of a 3D motion capture of patient walk has to be in .c3d format. Through the File Handler subcomponent, we can check whether the file of the 3D motion capture of the patient's walk was in the intended format, which is .c3d. We are then able to store the file if it is. Otherwise, an error is thrown and the user is prompted to upload another file for further analysis.

[R-3-F]: Upload file of a patient's physical examination for diagnosis. Similarly to requirement **R-2-F**, we were able to rely on the same subcomponents to receive the uploaded files and store them in the backend.

[R-2-NF]: The file of a patient's physical examination has to be in .xlsx format. In a similar manner to **R-1-NF**, we are also able to check for the intended file format and store it if the file is in the correct file format. Otherwise, an error is thrown similarly to prompt the upload of a file with the correct form.

[R-4-F]: Produce a diagnosis of the patient: The data processing module has successfully produced a proof-of-concept demonstrating its capabilities and advocating for further advancements. This module captures critical data points, specifically joint kinematics, and shows potential for enhancement through the incorporation of higher-quality data and the possible integration of artificial intelligence (AI). The module's comparison to normative data ensures a high standard of accuracy and precision in diagnosis, as each phase of the gait cycle is meticulously analyzed. This meticulous approach facilitates rapid and precise diagnosis. The system's capability to specify multiple diagnoses for each leg within a single gait cycle offers a nuanced and comprehensive understanding of the patient's gait impairments. This feature is particularly beneficial in clinical settings where patients may exhibit impairments in only one leg or specific regions of that leg. Consequently, the software precisely identifies

gait impairments, focusing on the specific body parts affected.

[R-3-NF]: The diagnosis should include gait impairments based on joint kinematics. The diagnosis results were validated through several test cases conducted in collaboration with external supervisor C. Greve. These tests confirmed the module's accuracy in identifying all gait impairments based on joint kinematics. The successful inclusion of all relevant impairments underscores the system's efficacy and reliability in clinical practice. However, further testing against a larger dataset is necessary to fully ensure the quality and robustness of the diagnosis.

[R-4-NF]: The diagnosis should include relevant values from the physical examination. We have managed to successfully extract the necessary and relevant values from the physical examination through the Diagnosis Generator subcomponent.

[R-5-NF]: The diagnosis should include gait impairments based on the muscle activity. Due to time constraints and the complexity of the task, there remains a need for an implementation that successfully integrates muscle activity into the diagnostic process.

[R-6-NF]: The diagnosis should include gait impairments based on the ground reaction forces. We were unable to fulfill this requirement due to a combination of time constraints and the inherent complexity of accurately measuring ground reaction forces. Our research and consultations with external supervisor C. Greve highlighted potential inaccuracies in the readings, which could compromise the reliability of the diagnosis. Additionally, the task is complicated by the occurrence of multiple steps on the same plates, which generate the ground reaction forces, making it challenging to isolate and analyze individual gait cycles accurately. Consequently, further development and testing are necessary to address these issues and ensure the robustness of the diagnosis based on ground reaction forces.

[R-7-NF]: The diagnosis should deliver rapid and accurate diagnoses within 5 seconds. We have successfully managed to produce a diagnosis within 1 second, which has exceeded the requirement.

[R-5-F]: Provide an executable to initialize and configure the entire system, ensuring that all components and services are properly set up and ready for operation. Due to time constraints, we were unable to fulfill this requirement. We explored the possibility of hosting the system on an online server; however, maintaining such a server would incur additional costs and necessitate ongoing measures for upkeep. Our goal is to reduce the barriers to clinical adoption, making a self-hosted solution less feasible. Subsequently, we considered packaging the entire system into a single executable. However, the use of multiple technologies and programming languages complicates this approach, making it challenging to create a unified executable. The importance of this requirement cannot be overstated, as the current setup process for the system is complex and may hinder its usability in clinical settings. Simplifying the initialization and configuration through an executable would significantly enhance the ease of adoption and operational efficiency for end-users.

[R-8-NF]: The executable should run on the Windows operating system. An executable has not been produced as mentioned above, hence this requirement has not been fulfilled yet.

[R-6-F]: Display the diagnosis to the clinicians. The data processing module has been able to produce a diagnosis, that was displayed to the clinicians through the Diagnosis Tables subcomponent.

[R-9-NF]: The diagnosis displayed should show what the gait impairments detected and when have they been detected. The diagnosis displays the detected gait impairments, indicating the specific part of the gait cycle, the affected joint, and the side (left or right) on which the impairment has been identified.

[R-10-NF]: The diagnosis is displayed in tables. The diagnosis is structured in 2 separate tables, the first one is of the actual diagnosis itself. This table contains the gait impairment, the phase of gait where it occurs, the affected joint, and the side where the impairment was detected. The second table displays the relevant physical examination parameters and their associated values.

[R-7-F]: Export the diagnosis. Due to the complexity and prioritization of certain tasks, we have decided to address this at a later stage, when we can incorporate all relevant gait parameters comprehensively.

[R-11-NF]: The diagnosis should be exportable to PDF format. Export functionality has not been Incorporated and as a result, this is also left uncompleted.

5.2 COMPARISON TO PREVIOUS WORK

The old system attempted to accelerate the variable components of the 3D CGA workflow but was hampered by its reliance on user input, which introduced unnecessary delays. Additionally, its rigidity limited scalability and extendibility, thereby diminishing care quality due to a lack of versatility. Unresolved functionalities and missing features further impeded the full utilization of the system.

In contrast, the new system has addressed these flaws by implementing a secure login portal, enabling clinicians to efficiently access the system. It supports the upload and handling of essential files, verifying formats to ensure accurate data processing. The new system produces a high-precision diagnosis by meticulously analyzing joint kinematics and displaying the results comprehensively to clinicians.

The transition from the old to the new system represents a substantial advancement in architectural design. The previous system was characterized by a rigid and less intuitive structure, which impeded both its maintainability and extendability. In contrast, the new system adopts a more modular and cohesive architecture, reducing decoupling and facilitat-

ing easier maintenance and integration of new features. While some functionalities remain unresolved in both systems, the new system is operational and offers tangible improvements, thereby enhancing the overall effectiveness compared to its predecessor. This architectural refinement not only addresses the limitations of the old system but also provides a more adaptable and user-friendly solution for medical professionals.

5.3 LIMITATIONS

Despite the advancements achieved with the new system, several limitations persist that warrant attention. Firstly, certain functionalities remain unresolved, which may impact the system's full potential. Notably, the integration of muscle activity into the diagnostic process has yet to be implemented due to time constraints and the inherent complexity of the task. This omission limits the comprehensiveness of the gait analysis, as muscle activity is a crucial component in understanding gait impairments.

Additionally, the system has not yet addressed the requirement for including gait impairments based on ground reaction forces. The complexity of accurately measuring and analyzing ground reaction forces, coupled with potential inaccuracies in readings as identified during consultations, has prevented the fulfillment of this requirement. This limitation reduces the system's ability to provide a complete assessment of gait impairments. One more thing to note is that currently, all the analysis takes place in the sagittal plane, which excludes the frontal and transversal planes of the diagnosis.

Another significant limitation is the lack of an executable for initializing and configuring the entire system. Efforts to host the system on an online server or package it into a unified executable were hindered by time constraints and technological challenges. Consequently, the current setup process remains complex and may impede ease of adoption in clinical settings.

Finally, the system's export functionality for generating PDF reports is still under development. The absence of this feature restricts the ability to produce easily shareable and portable diagnostic reports, which could be beneficial for clinical documentation and communication. In summary, while the new system offers substantial improvements over its predecessor, addressing these limitations will be crucial for enhancing its functionality and usability in clinical practice.

6 Conclusion

This thesis discussed the importance, conception, and implementation of a new gait diagnosis system for use by medical professionals at the UMCG. By focusing mainly on modularity, cohesion, maintainability, extendability, and utility, we were able to showcase significant improvement over the old gait diagnosis system which lacked in many aspects. With modern technologies being used, namely the React.js framework for the frontend and the Flask framework for the backend, we were able to ensure seamless interaction between modules and subcomponents, as well as efficient data processing that is vital to ensure a smooth process for clinicians and patients alike.

The key contributions made in this project include the design of the architecture of this project, with cohesion, modularity, maintainability, and extendability in mind such that any new features can be seamlessly integrated. Additionally, the system's ability to accurately analyze joint kinematics and compare patient data to normative benchmarks has demonstrated a high standard of diagnostic precision. Establishing secure user authentication and user-friendly interactions through the intuitive front-end are significant advancements from the previous system. Together, these contributions enhance the system's utility, reliability, and potential for future development, thus providing a solid foundation for ongoing advancements in gait analysis technology.

6.1 FUTURE WORK

Although this system represents a huge step forward in the diagnosis of gait impairment, some areas of enhancement for further research have been recognized.

The most critical task is the development of an executable that is capable of initializing and configuring the entire system. This will make the setup easier to use, increasing accessibility and therefore reducing the barriers to clinical adoption. Having an executable will ease deployment and maintenance since the users of the system can quickly get it up and running without detailed technical knowledge.

Future work should also involve the integration of diagnostics from different planes of motion. The system mainly operates by analyzing joint kinematics in the sagittal plane. If the inclusion of the frontal and transversal planes were to be done, that would give a more holistic view of gait impairments. This shall involve the enhancement of the data processing module to handle and analyze multi-planar motion data for improved accuracy and depth of diagnoses.

Another area of future development is more extensive testing with a large dataset. Initial test cases showed the accuracy of the system, but more extensive testing would be required to obtain robustness and reliability across different patient populations and under varying conditions. This would weed out edge cases and improve generalizability.

Finally, integrating muscle activity and ground reaction forces into the diagnostic process remains an important goal. Both of these measurements are integral parts of gait analysis, and such incorporation will further strengthen the abilities of the system in diagnosis concerning more types of impairments. This involves developing methods for the accurate capturing and analysis of muscle activity and ground reaction forces data alongside joint kinematics.

In summary, this thesis thus lays the proper foundations for advanced diagnosis of gait impairment and makes tremendously improved strides compared with any predecessors. Further development and fine-tuning will ensure this system's lead in clinical gait analysis technology, improving patient outcomes and advancing medical diagnostics. The milestones completed in this thesis contribute to the current body of knowledge and create a robust foundation for future research and development in the field of gait analysis and diagnosis.

REFERENCES

- [1] Gaids (gait diagnostic system): Ai-enabled, automated diagnostics in clinical gait rehabilitation. Unpublished Proposal. Health Technology Research and Innovation Cluster (HTRIC).
- [2] Amr Abdou, Emanoulli Kalostypis, and Nikola Zivkovic. Rug diagnosis system. <https://github.com/akr115/RUG-Gait-Diagnosis-System>, 2024.
- [3] Adrian Segura Lorente. Gait analysis tool: Improvement and optimization. Internal Document, 2022. Supervisors: Dr. Dimka Karastoyanova and Dr. Christian Greve.
- [4] Pedro Albuquerque, João Pedro Machado, Tanmay Tulsidas Verlekar, Paulo Lobato Correia, and Luís Ducla Soares. Remote gait type classification system using markerless 2d video. *Diagnostics*, 11(10):1824, October 2021.
- [5] Birol Balaban and Fatih Tok. Gait disturbances in patients with stroke. *PM&R*, 6(7):635–642, 2014.
- [6] Tianrui Cui, Le Yang, Xiaolin Han, Jiandong Xu, Yi Yang, and Tianling Ren. A low-cost, portable, and wireless in-shoe system based on a flexible porous graphene pressure sensor. *Materials*, 14(21):6475, October 2021.
- [7] Merck Manual. Gait disorders in older adults, 2023. Accessed 2/22/2023.
- [8] T F Novacheck. *Improving quality of life for individuals with cerebral palsy through treatment of gait impairment*. Mac Keith Press, Cambridge, England, December 2020.
- [9] Jacquelin Perry and Judith Burnfield. *Gait analysis*. SLACK, Thorofare, 2 edition, March 2010.
- [10] S. Pesenti, L. Garcia, V. Pomeroy, G. Authier, and C. Boulay. Gait abnormalities in adolescent idiopathic scoliosis patients: Do curvature amplitude matter? *Gait & Posture*, 97:399, 2022.
- [11] Walter Pirker and Regina Katzenschlager. Gait disorders in adults and the elderly: A clinical guide. *Wiener klinische Wochenschrift*, 129(3–4):81–95, October 2016.
- [12] Conda Development Team. *Conda Documentation*, 2024. Accessed: 2024-07-24.
- [13] Marjolein M. van der Krogt, Han Houdijk, Koen Wishaupt, Kim van Hutten, Sarah Dekker, and Annemieke I. Buizer. Development of a core set of gait features and their potential underlying impairments to assist gait data interpretation in children with cerebral palsy. *Frontiers in Human Neuroscience*, 16, October 2022.

A Physical Examination Values

Name (Dutch)	Name (English)
Dorsiflexie gebogen PROM	Dorsiflexion bent PROM
Dorsiflexie gebogen TONUS	Dorsiflexion bent TONUS
Dorsiflexie gebogen AOC	Dorsiflexion bent AOC
Dorsiflexie gebogen CLONUS	Dorsiflexion bent CLONUS
Dorsiflexie gestrekt PROM	Dorsiflexion extended PROM
Dorsiflexie gestrekt TONUS	Dorsiflexion extended TONUS
Dorsiflexie gestrekt AOC	Dorsiflexion extended AOC
Dorsiflexie gestrekt CLONUS	Dorsiflexion extended CLONUS
Plantairflexie PROM	Plantar flexion PROM
Pop-hoek PROM	Popliteal angle PROM
Pop-hoek TONUS	Popliteal angle TONUS
Pop-hoek AOC	Popliteal angle AOC
Knieflexie PROM	Knee flexion PROM
Knie-extensie PROM	Knee extension PROM
Heupflexie PROM	Hip flexion PROM
Heup-endorotatie PROM	Hip internal rotation PROM
Duncan-Ely PROM	Duncan-Ely Test PROM
Duncan-Ely TONUS	Duncan-Ely Test TONUS
Duncan-Ely AOC	Duncan-Ely Test AOC
Femorale-anteversiehoek (statiek)	Femoral anteversion angle (static)
Dijbeen-voethoek (statiek)	Thigh-Foot Angle (static)
Calcaneus varus PROM	Calcaneus varus PROM
Calcaneus Valgus PROM	Calcaneus Valgus PROM
Voorvoet pronatie PROM	Forefoot Pronation PROM
Voorvoet supinatie PROM	Forefoot Supination PROM
Knieflexie MRC	Knee flexion MRC
Heupextensie MRC	Hip extension MRC
Heupextensie PROM	Hip extension PROM
Knie-extensie MRC	Knee extension MRC
Heupflexie MRC	Hip flexion MRC
Dorsiflexie MRC	Dorsiflexion MRC
Inversie MRC	Inversion MRC
Eversie MRC	Eversion MRC

Table 1: The parameters found in the physical examination (LO.xlsx) file alongside their English translation. PROM stands for Passive Range Of Motion. TONUS stands for Muscle Tone. AOC stands for Angle Of Contracture. CLONUS is Involuntary Muscle Contractions. MRC is Manual Muscle Testing. The values exist for both the right side ('Rechts' in Dutch) and the left side ('Links' in Dutch)

B Screenshots of the New Gait Diagnosis Generator



The screenshot displays the login interface for the Gait Diagnosis System UMCG. The page features a light blue header with the UMCG logo on the left, the system name "Gait Diagnosis System UMCG" in the center, and the University of Groningen logo on the right. The main content area is a light blue gradient. In the center, there is a dark blue login box containing two white input fields labeled "Username" and "Password", and a blue "Login" button below them.

Figure 7: Login Page

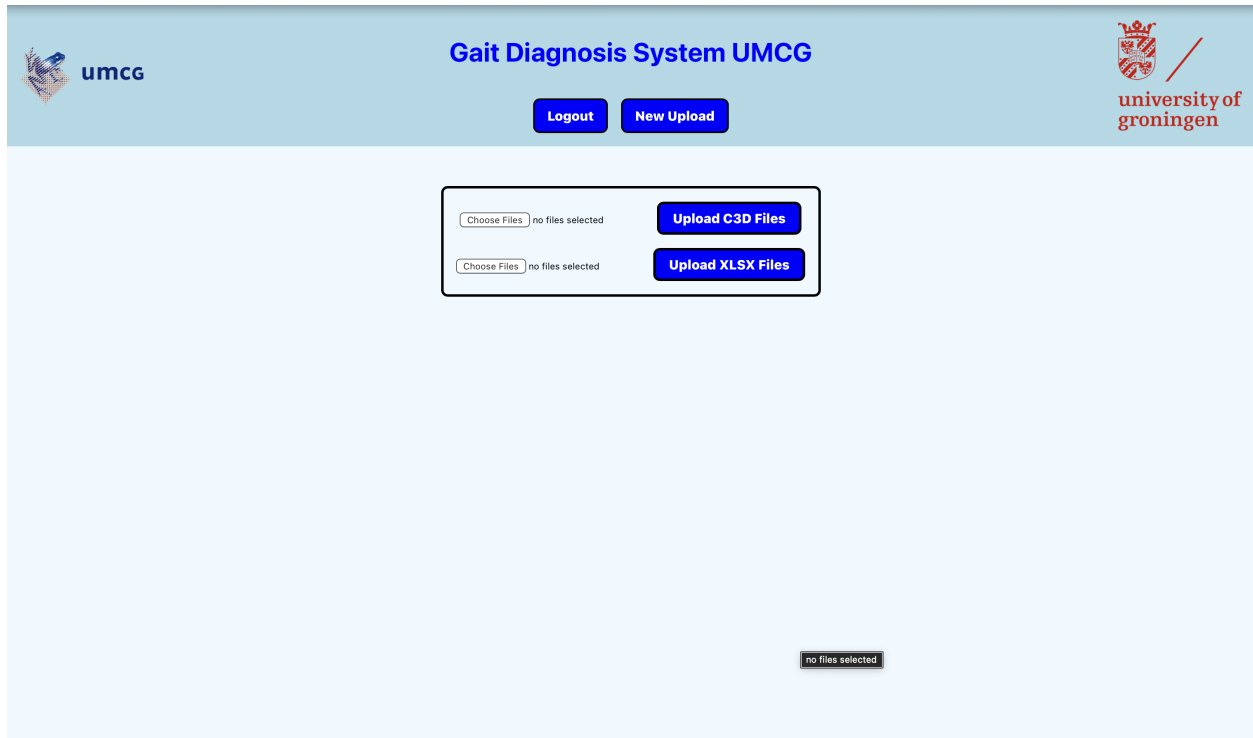


Figure 8: Gait Analysis Dashboard

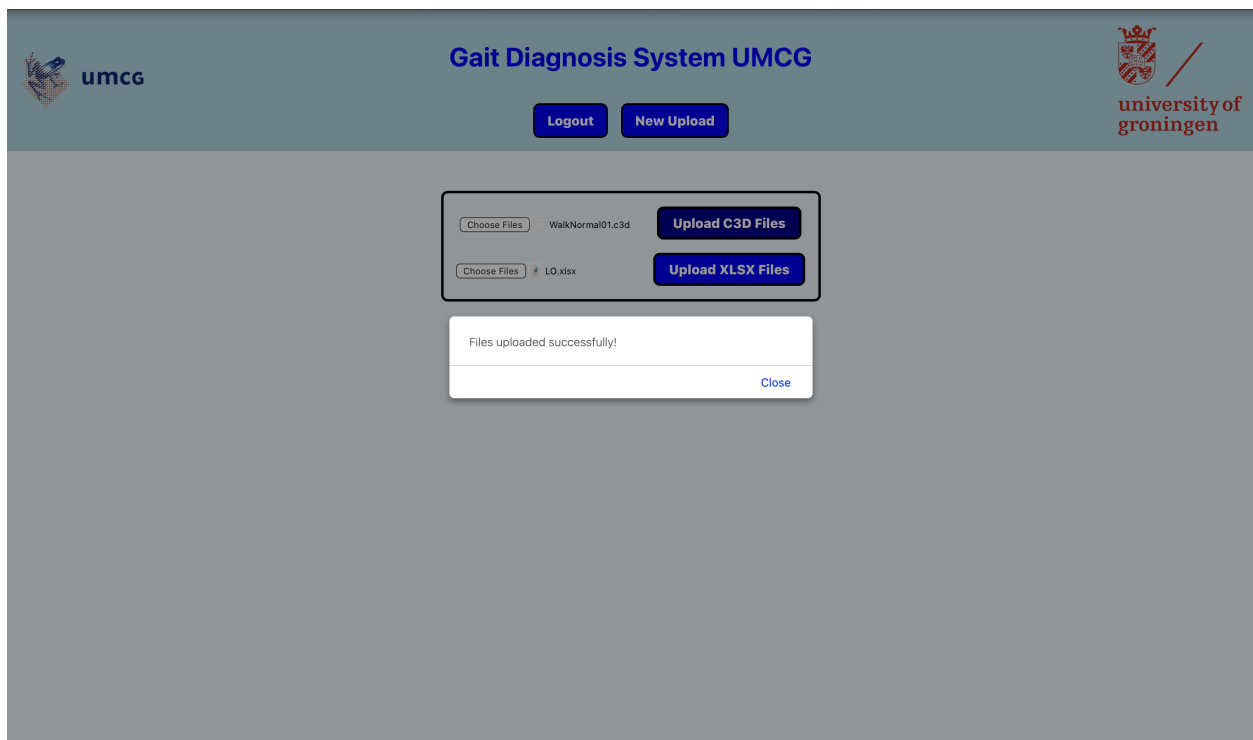




Figure 9: Files uploaded successfully popup message


umcg

Gait Diagnosis System UMCG



university of
 groningen

WalkNormal01.c3d

LO.xlsx

LO Table	
Name	Value
DorsiflexiegestrektPROMLinks	5
DorsiflexiegestrekTAOLinks	0
Knie-extensiePROMLinks	-5
HeupextensiePROMRechts	-5
Pop-hoekPROMRechts	80
DorsiflexieMRCLinks	4
Pop-hoekAOCRechts	(geen)
HeupextensiePROMLinks	0

Diagnosis Table				
Diagnosis	Joint Foot	Joint	Event Foot	Event
Plantairflexie (-9.0 graden)	Left	Ankle	Left	Foot Strike
Toegenomen knieflexie (17.0 graden)	Left	Knee	Left	Foot Strike
Geen relevante bevindingen tijdens	Left	Hip	Left	Foot Strike
Geen relevante bevindingen	Right	Knee	Left	Foot Strike
Geen relevante bevindingen	Left	Knee	Right	Foot Strike

Figure 10: The diagnosis of a patient displayed

C Conda Packages

Name	Version	Build	Channel
_anaconda_depends	2024.02	py311_openblas_1	conda-forge
abseil-cpp	20230802.0	h313beb8_2	
aiobotocore	2.7.0	py311hca03da5_0	
aiohttp	3.9.3	py311h80987f9_0	
aioitertools	0.7.1	pyhd3eb1b0_0	
aiosignal	1.2.0	pyhd3eb1b0_0	
alabaster	0.7.12	pyhd3eb1b0_0	
altair	5.0.1	py311hca03da5_0	
altgraph	0.17.4	pyhd8ed1ab_0	
anaconda-anon-usage	0.4.3	py311hd6b623d_100	
anaconda-catalogs	0.2.0	py311hca03da5_0	
anaconda-client	1.12.3	py311hca03da5_0	
anaconda-cloud-auth	0.1.4	py311hca03da5_0	
anaconda-navigator	2.5.2	py311hca03da5_0	
anaconda-project	0.11.1	py311hca03da5_0	
anyio	4.2.0	py311hca03da5_0	
aom	3.6.0	h313beb8_0	
appdirs	1.4.4	pyhd3eb1b0_0	
applaunchservices	0.3.0	py311hca03da5_0	
appnope	0.1.2	py311hca03da5_1001	
appscript	1.1.2	py311h80987f9_0	
archspec	0.2.3	pyhd3eb1b0_0	
argon2-ffi	21.3.0	pyhd3eb1b0_0	
argon2-ffi-bindings	21.2.0	py311h80987f9_0	
arrow	1.2.3	py311hca03da5_1	
arrow-cpp	14.0.2	hc7aafb3_1	
astroid	2.14.2	py311hca03da5_0	
astropy	5.3.4	py311hb9f6ed7_0	
asttokens	2.0.5	pyhd3eb1b0_0	
async-lru	2.0.4	py311hca03da5_0	
atomicwrites	1.4.0	py_0	
attrs	23.1.0	py311hca03da5_0	
automat	20.2.0	py_0	
autopep8	1.6.0	pyhd3eb1b0_1	
aws-c-auth	0.6.19	h80987f9_0	
aws-c-cal	0.5.20	h80987f9_0	
aws-c-common	0.8.5	h80987f9_0	
aws-c-compression	0.2.16	h80987f9_0	

aws-c-event-stream	0.2.15	h313beb8_0	
aws-c-http	0.6.25	h80987f9_0	
aws-c-io	0.13.10	h80987f9_0	
aws-c-mqtt	0.7.13	h80987f9_0	
aws-c-s3	0.1.51	h80987f9_0	
aws-c-sdkutils	0.1.6	h80987f9_0	
aws-checksums	0.1.13	h80987f9_0	
aws-crt-cpp	0.18.16	h313beb8_0	
aws-sdk-cpp	1.10.55	h313beb8_0	
babel	2.11.0	py311hca03da5_0	
backports	1.1	pyhd3eb1b0_0	
backports.functools_lru_cache 1.6.4	pyhd3eb1b0_0		
backports.tempfile	1.0	pyhd3eb1b0_1	
backports.weakref	1.0.post1	py_1	
bcrypt	3.2.0	py311h80987f9_1	
beautifulsoup4	4.12.2	py311hca03da5_0	
binaryornot	0.4.4	pyhd3eb1b0_1	
black	23.11.0	py311hca03da5_0	
blas	1.0	openblas	
bleach	4.1.0	pyhd3eb1b0_0	
blinker	1.6.2	py311hca03da5_0	
blosc	1.21.3	h313beb8_0	
bokeh	3.3.4	py311hb6e6a13_0	
boltons	23.0.0	py311hca03da5_0	
boost-cpp	1.82.0	h48ca7d4_2	
botocore	1.31.64	py311hca03da5_0	
bottleneck	1.3.7	py311hb9f6ed7_0	
brotli	1.0.9	h1a28f6b_7	
brotli-bin	1.0.9	h1a28f6b_7	
brotli-python	1.0.9	py311h313beb8_7	
brunli	0.1	hc377ac9_1	
bzip2	1.0.8	h620ffc9_4	
c-ares	1.19.1	h80987f9_0	
c-blosc2	2.12.0	h7df6c2f_0	
ca-certificates	2024.7.4	hf0a4a13_0	conda-forge
cachelib	0.13.0	pypi_0	pypi
cachetools	4.2.2	pyhd3eb1b0_0	
cctools	949.0.1	hc179dcd_25	
cctools_osx-arm64	949.0.1	h332cad3_25	
certifi	2024.7.4	pyhd8ed1ab_0	conda-forge
cff	1.16.0	py311h80987f9_0	
cfitsio	3.470	h7f6438f_7	
chardet	4.0.0	py311hca03da5_1003	
charls	2.2.0	hc377ac9_0	
charset-normalizer	2.0.4	pyhd3eb1b0_0	

click	8.1.7	py311hca03da5_0	
cloudpickle	2.2.1	py311hca03da5_0	
clyent	1.2.2	py311hca03da5_1	
colorama	0.4.6	py311hca03da5_0	
colorcet	3.0.1	py311hca03da5_0	
comm	0.1.2	py311hca03da5_0	
conda	24.5.0	py311hca03da5_0	
conda-build	24.1.2	py311hca03da5_0	
conda-content-trust	0.2.0	py311hca03da5_0	
conda-index	0.4.0	pyhd3eb1b0_0	
conda-libmamba-solver	24.1.0	pyhd3eb1b0_0	
conda-pack	0.6.0	pyhd3eb1b0_0	
conda-package-handling	2.2.0	py311hca03da5_0	
conda-package-streaming	0.9.0	py311hca03da5_0	
conda-repo-cli	1.0.88	py311hca03da5_0	
conda-token	0.4.0	pyhd3eb1b0_0	
conda-verify	3.4.2	py_1	
constantly	23.10.4	py311hca03da5_0	
contourpy	1.2.0	py311h48ca7d4_0	
cookiecutter	2.5.0	py311hca03da5_0	
cryptography	42.0.2	py311hd4332d6_0	
cssselect	1.2.0	py311hca03da5_0	
curl	8.5.0	h02f6b3c_0	
cycler	0.11.0	pyhd3eb1b0_0	
cyrus-sasl	2.1.28	h9131b1a_1	
cytoolz	0.12.2	py311h80987f9_0	
dash	2.7.0	pypi_0	pypi
dash-bootstrap-components 1.2.1	pypi_0	pypi	
dash-core-components	2.0.0	pypi_0	pypi
dash-daq	0.5.0	pypi_0	pypi
dash-extensions	0.1.8	pypi_0	pypi
dash-html-components	2.0.0	pypi_0	pypi
dash-renderer	0.9.0	pypi_0	pypi
dash-table	5.0.0	pypi_0	pypi
dask	2023.11.0	py311hca03da5_0	
dask-core	2023.11.0	py311hca03da5_0	
datashader	0.16.0	py311hca03da5_0	
dateparser	1.1.1	pypi_0	pypi
dav1d	1.2.1	h80987f9_0	
debugpy	1.6.7	py311h313beb8_0	
decorator	5.1.1	pyhd3eb1b0_0	
defusedxml	0.7.1	pyhd3eb1b0_0	
diff-match-patch	20200713	pyhd3eb1b0_0	
dill	0.3.7	py311hca03da5_0	
distributed	2023.11.0	py311hca03da5_0	

distro	1.8.0	py311hca03da5_0	
docstring-to-markdown	0.11	py311hca03da5_0	
docutils	0.18.1	py311hca03da5_3	
editorconfig	0.12.4	pypi_0	pypi
empath	0.89	pypi_0	pypi
entrypoints	0.4	py311hca03da5_0	
et_xmlfile	1.1.0	py311hca03da5_0	
executing	0.8.3	pyhd3eb1b0_0	
ezc3d	1.5.10	py311_python3_h19d7a53_0	conda-forge
filelock	3.13.1	py311hca03da5_0	
flake8	6.0.0	py311hca03da5_0	
flask	2.2.2	pypi_0	pypi
flask-caching	2.0.1	pypi_0	pypi
flask-cors	4.0.0	pyhd8ed1ab_0	conda-forge
flask-wtf	1.2.1	pyhd8ed1ab_0	conda-forge
fnt	9.1.0	h48ca7d4_0	
fonttools	4.25.0	pyhd3eb1b0_0	
fpdf	1.7.2	pypi_0	pypi
freetype	2.12.1	h1192e45_0	
frozendict	2.4.2	py311hca03da5_0	
frozenlist	1.4.0	py311h80987f9_0	
fsspec	2023.10.0	py311hca03da5_0	
future	0.18.3	py311hca03da5_0	
gensim	4.3.0	py311h6956b77_0	
gettext	0.21.0	h13f89a0_1	
gflags	2.2.2	h313beb8_1	
giflib	5.2.1	h80987f9_3	
gitdb	4.0.7	pyhd3eb1b0_0	
gitpython	3.1.37	py311hca03da5_0	
glib	2.78.4	h313beb8_0	
glib-tools	2.78.4	h313beb8_0	
glog	0.5.0	h313beb8_1	
gmp	6.2.1	hc377ac9_3	
gmpy2	2.1.2	py311h40f64dc_0	
greenlet	3.0.1	py311h313beb8_0	
grpc-cpp	1.48.2	hc60591f_4	
gst-plugins-base	1.14.1	h313beb8_1	
gstreamer	1.14.1	h80987f9_1	
gtest	1.14.0	h48ca7d4_0	
h11	0.9.0	pypi_0	pypi
h2	3.2.0	pypi_0	pypi
h5py	3.8.0	pypi_0	pypi
hdf5	1.12.1	h05c076b_3	
heapdict	1.0.1	pyhd3eb1b0_0	
holoviews	1.18.3	py311hca03da5_0	

hpack	3.0.0	pypi_0	pypi
hvplot	0.9.2	py311hca03da5_0	
hyperframe	5.2.0	pypi_0	pypi
hyperlink	21.0.0	pyhd3eb1b0_0	
icu	73.1	h313beb8_0	
idna	3.4	py311hca03da5_0	
imagecodecs	2023.1.23	py311h5e7c512_0	
imageio	2.33.1	py311hca03da5_0	
imagesize	1.4.1	py311hca03da5_0	
imbalanced-learn	0.11.0	py311hca03da5_1	
importlib-metadata	7.0.1	py311hca03da5_0	
importlib_metadata	7.0.1	hd3eb1b0_0	
incremental	22.10.0	pyhd3eb1b0_0	
inflection	0.5.1	py311hca03da5_0	
iniconfig	1.1.1	pyhd3eb1b0_0	
intake	0.6.8	py311hca03da5_0	
intervaltree	3.1.0	pyhd3eb1b0_0	
ipykernel	6.28.0	py311hca03da5_0	
ipython	8.20.0	py311hca03da5_0	
ipython_genutils	0.2.0	pyhd3eb1b0_1	
ipywidgets	7.6.5	pyhd3eb1b0_2	
isort	5.9.3	pyhd3eb1b0_0	
itemadapter	0.3.0	pyhd3eb1b0_0	
itemloaders	1.1.0	py311hca03da5_0	
itsdangerous	2.2.0	pyhd8ed1ab_0	conda-forge
jaraco.classes	3.2.1	pyhd3eb1b0_0	
jedi	0.18.1	py311hca03da5_1	
jellyfish	1.0.1	py311h15d1925_0	
jinja2	3.1.2	pypi_0	pypi
jmespath	1.0.1	py311hca03da5_0	
joblib	1.2.0	py311hca03da5_0	
jpeg	9e	h80987f9_1	
jq	1.6	h1a28f6b_1	
jsbeautifier	1.15.1	pypi_0	pypi
json5	0.9.6	pyhd3eb1b0_0	
jsonpatch	1.32	pyhd3eb1b0_0	
jsonpointer	2.1	pyhd3eb1b0_0	
jsonschemas	4.19.2	py311hca03da5_0	
jsonschemas-specifications	2023.7.1	py311hca03da5_0	
jupyter	1.0.0	py311hca03da5_9	
jupyter-lsp	2.2.0	py311hca03da5_0	
jupyter_client	8.6.0	py311hca03da5_0	
jupyter_console	6.6.3	py311hca03da5_0	
jupyter_core	5.5.0	py311hca03da5_0	
jupyter_events	0.8.0	py311hca03da5_0	

jupyter_server	2.10.0	py311hca03da5_0	
jupyter_server_terminals	0.4.4	py311hca03da5_1	
jupyterlab	4.0.11	py311hca03da5_0	
jupyterlab-variableinspector 3.1.0	py311hca03da5_0		
jupyterlab_pygments	0.1.2	py_0	
jupyterlab_server	2.25.1	py311hca03da5_0	
jupyterlab_widgets	3.0.9	py311hca03da5_0	
jxrlib	1.1	h1a28f6b_2	
keyring	23.13.1	py311hca03da5_0	
kiwisolver	1.4.4	py311h313beb8_0	
krb5	1.20.1	hf3e1bf2_1	
langdetect	1.0.9	pypi_0	pypi
lazy-object-proxy	1.6.0	py311h80987f9_0	
lazy_loader	0.3	py311hca03da5_0	
lcms2	2.12	hba8e193_0	
ld64	530	hb29bf3f_25	
ld64_osx-arm64	530	h001ce53_25	
ldid	2.1.5	h20b2a84_3	
lerc	3.0	hc377ac9_0	
libaec	1.0.4	hc377ac9_1	
libarchive	3.6.2	h62fee54_2	
libavif	0.11.1	h80987f9_0	
libboost	1.82.0	h0bc93f9_2	
libbrotlicommon	1.0.9	h1a28f6b_7	
libbrotlidec	1.0.9	h1a28f6b_7	
libbrotlienc	1.0.9	h1a28f6b_7	
libclang	14.0.6	default_h1b80db6_1	
libclang13	14.0.6	default_h24352ff_1	
libcurl	8.5.0	h3e2b118_0	
libcxx	17.0.6	h5f092b4_0	conda-forge
libdeflate	1.17	h80987f9_1	
libedit	3.1.20230828	h80987f9_0	
libev	4.33	h1a28f6b_1	
libevent	2.1.12	h02f6b3c_1	
libexpat	2.6.2	hebf3989_0	conda-forge
libffi	3.4.4	hca03da5_0	
libgfortran	5.0.0	11_3_0_hca03da5_28	
libgfortran5	11.3.0	h009349e_28	
libglib	2.78.4	h0a96307_0	
libiconv	1.16	h1a28f6b_2	
liblief	0.12.3	h313beb8_0	
libllvm14	14.0.6	h7ec7a93_3	
libmamba	1.5.6	h15e39b3_0	
libmambapy	1.5.6	py311h1c5506f_0	
libnghttp2	1.57.0	h62f6fdd_0	

libopenblas	0.3.21	h269037a_0	
libpng	1.6.39	h80987f9_0	
libpq	12.17	h02f6b3c_0	
libprotobuf	3.20.3	h514c7bf_0	
libsodium	1.0.18	h1a28f6b_0	
libsolv	0.7.24	h514c7bf_0	
libspatialindex	1.9.3	hc377ac9_0	
libsqlite	3.45.3	h091b4b1_0	conda-forge
libssh2	1.10.0	h02f6b3c_2	
libthrift	0.15.0	h73c2103_2	
libtiff	4.5.1	h313beb8_0	
libwebp-base	1.3.2	h80987f9_0	
libxml2	2.10.4	h0dcf63f_1	
libxslt	1.1.37	h80987f9_1	
libzlib	1.2.13	hfb2fe0b_6	conda-forge
libzopfli	1.0.3	hc377ac9_0	
linkify-it-py	2.0.0	py311hca03da5_0	
llvm-openmp	14.0.6	hc6e5704_0	
llvmlite	0.42.0	py311h313beb8_0	
locket	1.0.0	py311hca03da5_0	
lxml	4.9.3	py311h50ffb84_0	
lz4	4.3.2	py311h80987f9_0	
lz4-c	1.9.4	h313beb8_0	
lzo	2.10	h1a28f6b_2	
macholib	1.16.3	pyhd8ed1ab_0	conda-forge
markdown	3.4.1	py311hca03da5_0	
markdown-it-py	2.2.0	py311hca03da5_1	
markupsafe	2.1.3	py311h80987f9_0	
matplotlib	3.6.2	pypi_0	pypi
matplotlib-inline	0.1.6	py311hca03da5_0	
mccabe	0.7.0	pyhd3eb1b0_0	
mdit-py-plugins	0.3.0	py311hca03da5_0	
mdurl	0.1.0	py311hca03da5_0	
menuinst	2.0.2	py311hca03da5_0	
mistune	2.0.4	py311hca03da5_0	
more-itertools	8.14.0	pypi_0	pypi
mpc	1.1.0	h8c48613_1	
mpfr	4.0.2	h695f6f0_1	
mpmath	1.3.0	py311hca03da5_0	
msgpack-python	1.0.3	py311h48ca7d4_0	
multidict	6.0.4	py311h80987f9_0	
multipledispatch	0.6.0	py311hca03da5_0	
munkres	1.1.4	py_0	
mypy	1.8.0	py311h80987f9_0	
mypy_extensions	1.0.0	py311hca03da5_0	

mysql	5.7.24	ha71a6ea_2	
navigator-updater	0.4.0	py311hca03da5_1	
nbclient	0.8.0	py311hca03da5_0	
nbconvert	7.10.0	py311hca03da5_0	
nbformat	5.9.2	py311hca03da5_0	
ncurses	6.4	h313beb8_0	
nest-asyncio	1.6.0	py311hca03da5_0	
networkx	3.1	py311hca03da5_0	
nltk	3.7	pypi_0	pypi
notebook	7.0.8	py311hca03da5_0	
notebook-shim	0.2.3	py311hca03da5_0	
ntlm-auth	1.5.0	pypi_0	pypi
numba	0.59.0	py311h7aadaa7_0	
numexpr	2.8.7	py311h6dc990b_0	
numpy	1.24.3	pypi_0	pypi
numpydoc	1.5.0	py311hca03da5_0	
oauthlib	3.2.2	pypi_0	pypi
oniguruma	6.9.7.1	h1a28f6b_0	
openjpeg	2.3.0	h7a6adac_2	
openpyxl	3.1.2	py311h80987f9_0	
openssl	3.3.1	hfb2fe0b_1	conda-forge
orc	1.7.4	hdca1487_1	
overrides	7.4.0	py311hca03da5_0	
packaging	23.1	py311hca03da5_0	
pandas	1.4.2	pypi_0	pypi
pandocfilters	1.5.0	pyhd3eb1b0_0	
panel	1.3.8	py311hca03da5_0	
param	2.0.2	py311hca03da5_0	
parsel	1.8.1	py311hca03da5_0	
parso	0.8.3	pyhd3eb1b0_0	
partd	1.4.1	py311hca03da5_0	
patch	2.7.6	h1a28f6b_1001	
pathspecc	0.10.3	py311hca03da5_0	
patsy	0.5.3	py311hca03da5_0	
pcre2	10.42	hb066dcc_0	
pexpect	4.8.0	pyhd3eb1b0_3	
pickleshare	0.7.5	pyhd3eb1b0_1003	
pillow	10.2.0	py311h80987f9_0	
pip	23.3.1	py311hca03da5_0	
pkce	1.0.3	py311hca03da5_0	
pkginfo	1.9.6	py311hca03da5_0	
platformdirs	3.10.0	py311hca03da5_0	
plotly	5.11.0	pypi_0	pypi
pluggy	0.12.0	pypi_0	pypi
ply	3.11	py311hca03da5_0	

prometheus_client	0.14.1	py311hca03da5_0	
prompt-toolkit	3.0.43	py311hca03da5_0	
prompt_toolkit	3.0.43	hd3eb1b0_0	
protego	0.1.16	py_0	
protobuf	3.20.3	py311h313beb8_0	
psutil	5.9.0	py311h80987f9_0	
ptyprocess	0.7.0	pyhd3eb1b0_2	
pure_eval	0.2.2	pyhd3eb1b0_0	
py-cpuinfo	9.0.0	py311hca03da5_0	
py-lief	0.12.3	py311h313beb8_0	
pyarrow	14.0.2	py311ha07b5f9_0	
pyasn1	0.4.8	pyhd3eb1b0_0	
pyasn1-modules	0.2.8	py_0	
pybind11-abi	4	hd3eb1b0_1	
pycodestyle	2.10.0	py311hca03da5_0	
pycosat	0.6.6	py311h80987f9_0	
pyparser	2.21	pyhd3eb1b0_0	
pyct	0.5.0	py311hca03da5_0	
pycurl	7.45.2	py311h02f6b3c_1	
pydantic	1.10.12	py311h80987f9_1	
pydeck	0.8.0	py311hca03da5_2	
pydispatcher	2.0.5	py311hca03da5_2	
pydocstyle	6.3.0	py311hca03da5_0	
pyerfa	2.0.0	py311h80987f9_0	
pyflakes	3.0.1	py311hca03da5_0	
pygments	2.15.1	py311hca03da5_1	
pyinstaller	6.7.0	py311ha566751_0	conda-forge
pyinstaller-hooks-contrib 2024.7	pyhd8ed1ab_0	conda-forge	
pyjwt	2.4.0	py311hca03da5_0	
pylint	2.16.2	py311hca03da5_0	
pylint-venv	2.3.0	py311hca03da5_0	
pyls-spyder	0.4.0	pyhd3eb1b0_0	
pyobjc-core	9.0	py311h3eb5a62_1	
pyobjc-framework-cocoa	9.0	py311hb094c41_0	
pyobjc-framework-coreservices 9.0	py311hdd8dd1f_0		
pyobjc-framework-fsevents 9.0	py311hca03da5_0		
pyodbc	5.0.1	py311h313beb8_0	
pyopenssl	24.0.0	py311hca03da5_0	
pyoxidizer	0.24.0	pypi_0	pypi
pyparsing	3.0.9	py311hca03da5_0	
pyqt	5.15.10	py311h313beb8_0	
pyqt5-sip	12.13.0	py311h80987f9_0	
pyqtwebengine	5.15.10	py311h313beb8_0	
pysocks	1.7.1	py311hca03da5_0	
pytables	3.9.2	py311h0326f10_0	

pytest	7.4.0	py311hca03da5_0	conda-forge
python	3.11.8	hdf0ec26_0_cpython	
python-dateutil	2.8.2	pyhd3eb1b0_0	conda-forge
python-dotenv	0.21.0	py311hca03da5_0	
python-fastjsonschema	2.16.2	py311hca03da5_0	conda-forge
python-json-logger	2.0.7	py311hca03da5_0	
python-libarchive-c	2.9	pyhd3eb1b0_1	conda-forge
python-lmdb	1.4.1	py311h313beb8_0	
python-lsp-black	1.2.1	py311hca03da5_0	conda-forge
python-lsp-jsonrpc	1.0.0	pyhd3eb1b0_0	
python-lsp-server	1.7.2	py311hca03da5_0	conda-forge
python-slugify	5.0.2	pyhd3eb1b0_0	
python-snappy	0.6.1	py311h313beb8_0	conda-forge
python-tzdata	2023.3	pyhd3eb1b0_0	
python.app	3	py311h80987f9_0	conda-forge
python_abi	3.11	4_cp311	
pytoolconfig	1.2.6	py311hca03da5_0	conda-forge
pytz	2023.3.post1	py311hca03da5_0	
pyviz_comms	3.0.0	py311hca03da5_0	conda-forge
pywavelets	1.5.0	py311hb9f6ed7_0	
pyyaml	6.0.1	py311h80987f9_0	conda-forge
pyzmq	25.1.2	py311h313beb8_0	
qdarkstyle	3.0.2	pyhd3eb1b0_0	conda-forge
qstylizer	0.2.2	py311hca03da5_0	
qt-main	5.15.2	h0917680_10	conda-forge
qt-webengine	5.15.9	h2903aaf_7	
qtawesome	1.2.2	py311hca03da5_0	conda-forge
qtconsole	5.4.2	py311hca03da5_0	
qtpy	2.4.1	py311hca03da5_0	conda-forge
queuelib	1.6.2	py311hca03da5_0	
re2	2022.04.01	hc377ac9_0	conda-forge
readline	8.2	h1a28f6b_0	
referencing	0.30.2	py311hca03da5_0	conda-forge
regex	2022.3.2	pypi_0	
reproc	14.2.4	hc377ac9_1	conda-forge
reproc-cpp	14.2.4	hc377ac9_1	
requests	2.31.0	py311hca03da5_1	conda-forge
requests-file	1.5.1	pyhd3eb1b0_0	
requests-toolbelt	1.0.0	py311hca03da5_0	conda-forge
retrying	1.3.4	pypi_0	
rfc3339-validator	0.1.4	py311hca03da5_0	conda-forge
rfc3986-validator	0.1.1	py311hca03da5_0	
rich	13.3.5	py311hca03da5_0	conda-forge
rope	1.7.0	py311hca03da5_0	
rpds-py	0.10.6	py311hf0e4da2_0	conda-forge

rtree	1.0.1	py311hca03da5_0	
ruamel.yaml	0.17.21	py311h80987f9_0	
ruamel_yaml	0.17.21	py311h80987f9_0	
s3fs	2023.10.0	py311hca03da5_0	
scikit-image	0.22.0	py311h7aadaa7_0	
scikit-learn	1.4.2	pypi_0	pypi
scipy	1.12.0	pypi_0	pypi
scrapy	2.8.0	py311hca03da5_0	
seaborn	0.12.2	py311hca03da5_0	
semver	2.13.0	pyhd3eb1b0_0	
send2trash	1.8.2	py311hca03da5_0	
service_identity	18.1.0	pyhd3eb1b0_1	
setuptools	68.2.2	py311hca03da5_0	
sip	6.7.12	py311h313beb8_0	
six	1.16.0	pyhd3eb1b0_1	
smart_open	5.2.1	py311hca03da5_0	
smmap	4.0.0	pyhd3eb1b0_0	
snappy	1.1.10	h313beb8_1	
sniffio	1.3.0	py311hca03da5_0	
snowballstemmer	2.2.0	pyhd3eb1b0_0	
sortedcontainers	2.4.0	pyhd3eb1b0_0	
soupsieve	2.5	py311hca03da5_0	
sphinx	5.0.2	py311hca03da5_0	
sphinxcontrib-applehelp	1.0.2	pyhd3eb1b0_0	
sphinxcontrib-devhelp	1.0.2	pyhd3eb1b0_0	
sphinxcontrib-htmlhelp	2.0.0	pyhd3eb1b0_0	
sphinxcontrib-jsmath	1.0.1	pyhd3eb1b0_0	
sphinxcontrib-qthelp	1.0.3	pyhd3eb1b0_0	
sphinxcontrib-serializinghtml	1.1.5	pyhd3eb1b0_0	
spyder	5.4.3	py311hca03da5_1	
spyder-kernels	2.4.4	py311hca03da5_0	
sqlalchemy	2.0.25	py311h80987f9_0	
sqlite	3.41.2	h80987f9_0	
stack_data	0.2.0	pyhd3eb1b0_0	
statsmodels	0.14.0	py311hb9f6ed7_0	
streamlit	1.30.0	py311hca03da5_0	
sympy	1.12	py311hca03da5_0	
tabulate	0.9.0	py311hca03da5_0	
tapi	1100.0.11	h8754e6a_1	
tbb	2021.8.0	h48ca7d4_0	
tblib	1.7.0	pyhd3eb1b0_0	
tenacity	8.2.2	py311hca03da5_0	
terminado	0.17.1	py311hca03da5_0	
text-unidecode	1.3	pyhd3eb1b0_0	
textdistance	4.2.1	pyhd3eb1b0_0	

threadpoolctl	2.2.0	pyh0d69192_0	
three-merge	0.1.1	pyhd3eb1b0_0	
tiffle	2023.4.12	py311hca03da5_0	
tinycss2	1.2.1	py311hca03da5_0	
tk	8.6.13	h5083fa2_1	conda-forge
tldextract	3.2.0	pyhd3eb1b0_0	
toml	0.10.2	pyhd3eb1b0_0	
tomlkit	0.11.1	py311hca03da5_0	
toolz	0.12.0	py311hca03da5_0	
tornado	6.3.3	py311h80987f9_0	
tqdm	4.65.0	py311hb6e6a13_0	
traitlets	5.7.1	py311hca03da5_0	
truststore	0.8.0	py311hca03da5_0	
twisted	23.10.0	py311hca03da5_0	
typing-extensions	4.9.0	py311hca03da5_1	
typing_extensions	4.9.0	py311hca03da5_1	
tzdata	2023d	h04d1e81_0	
tzlocal	2.1	py311hca03da5_1	
uc-micro-py	1.0.1	py311hca03da5_0	
ujson	5.4.0	py311h313beb8_0	
unidecode	1.2.0	pyhd3eb1b0_0	
unixodbc	2.3.11	h1a28f6b_0	
urllib3	2.0.7	py311hca03da5_0	
utf8proc	2.6.1	h80987f9_1	
validators	0.18.2	pyhd3eb1b0_0	
w3lib	2.1.2	py311hca03da5_0	
watchdog	4.0.1	py311hd3f4193_0	conda-forge
wcwidth	0.2.5	pyhd3eb1b0_0	
webencodings	0.5.1	py311hca03da5_1	
websocket-client	0.58.0	py311hca03da5_4	
werkzeug	2.2.2	pypi_0	pypi
whatthepatch	1.0.2	py311hca03da5_0	
wheel	0.41.2	py311hca03da5_0	
widgetsnbextension	3.5.2	py311hca03da5_1	
wrapt	1.14.1	py311h80987f9_0	
wtforms	3.1.2	pyhd8ed1ab_0	conda-forge
wurlitzer	3.0.2	py311hca03da5_0	
xarray	2023.6.0	py311hca03da5_0	
xlwings	0.29.1	py311hca03da5_0	
xyzservices	2022.9.0	py311hca03da5_1	
xz	5.4.5	h80987f9_0	
yaml	0.2.5	h1a28f6b_0	
yaml-cpp	0.8.0	h313beb8_0	
yapf	0.31.0	pyhd3eb1b0_0	
yaml	1.9.3	py311h80987f9_0	

zeromq	4.3.5	h313beb8_0	conda-forge
zfp	1.0.0	h313beb8_0	
zict	3.0.0	py311hca03da5_0	
zipp	3.17.0	py311hca03da5_0	
zlib	1.2.13	hfb2fe0b_6	
zlib-ng	2.0.7	h80987f9_0	
zope	1.0	py311hca03da5_1	
zope.interface	5.4.0	py311h80987f9_0	
zstandard	0.19.0	py311h80987f9_0	
zstd	1.5.5	hd90d995_0	

Table 2: Conda packages used for the project’s back end and data processing modules in Python, obtained through running the command `conda list`

D Data Processing Decision Trees

Ankle Joint Diagnosis Decision Trees

```

1 def diagnose_ankle(data, side):
2     df_stance = data[data['Foot'] == side]
3     df_swing = data[data['Foot'] != side]
4     plantar_flag = 0
5     lo_variables = []
6     joint = "Ankle"
7     results = []
8
9     # Diagnosis for the stance phase
10    for index, row in df_stance.iterrows():
11        # Access the associated joint angle with the side
12        ankle = row['LAnkle'] if side == 'Left' else row['RAnkle']
13        ankle_degrees = row['LAnkle Degrees'] if side == 'Left' else row['
14            RAnkle Degrees']
15        if row.Event == 'Foot Strike':
16            if ankle == -1:
17                # Plantarflexion
18                if side == 'Left':
19                    # Extract relevant variables for the LO
20                    lo_variables.extend(["DorsiflexieMRCLinks", "
21                        DorsiflexiegebogenPROMLinks", "
22                        DorsiflexiegebogenAOCLinks",
23                        "DorsiflexiegestrektPROMLinks", "
24                        DorsiflexiegestrektAOCLinks"])
25                else:
26                    # Extract relevant variables for the LO
27                    lo_variables.extend(["DorsiflexieMRCRechts", "
28                        DorsiflexiegebogenPROMRechts",
29                        "DorsiflexiegebogenAOCRechts", "
30                        DorsiflexiegestrektPROMRechts",
31                        "DorsiflexiegestrektAOCRechts"])
32                results.append([f"Plantairflexie ({str(np.round(
33                    ankle_degrees))} graden)", side,
34                    joint, row.Foot, row.Event])
35            else:
36                # No relevant finding in this case
37                results.append(["Geen relevante bevindingen", side, joint,
38                    row.Foot, row.Event])
39                plantar_flag = 1
40        elif row.Event == 'Loading Response':
41            if plantar_flag == 1 and ankle == -1:
42                # No/decreased plantarflexion
43                results.append([f"Geen/afgenomen plantairflexiebeweging",
44                    side, joint, row.Foot, row.Event])
45            else:
46                # No relevant finding in this case
47                results.append(["Geen relevante bevindingen", side,
48                    joint, row.Foot, row.Event])
49        elif row.Event == 'Mid Stance':

```

```

41     if ankle == -1:
42         # Plantarflexion
43         results.append([f"Plantarflexie ({str(np.round(
44             ankle_degrees))} graden)",
45             side, joint, row.Foot, row.Event])
46     elif ankle == 1:
47         # Increased dorsiflexion
48         results.append([f"Toegenomen dorsaalflexie", side,
49             joint, row.Foot, row.Event])
50     else:
51         # Ankle range of motion within normal range
52         results.append([f"Enkel range of motion ({str(np.round(
53             ankle_degrees))} graden) binnen "
54             f"range van normaal",
55             side, joint, row.Foot, row.Event])
56 elif row.Event == 'Terminal Stance':
57     if ankle == -1:
58         # Decreased dorsiflexion
59         if side == 'Left':
60             # Extract relevant variables for the LO
61             lo_variables.extend(["DorsiflexiegebogenPROMLinks", "
62                 DorsiflexiegebogenAOCLinks",
63                 "DorsiflexiegestrektPROMLinks", "
64                 DorsiflexiegestrektAOCLinks"])
65         else:
66             # Extract relevant variables for the LO
67             lo_variables.extend(["DorsiflexiegebogenPROMRechts", "
68                 DorsiflexiegebogenAOCRechts",
69                 "DorsiflexiegestrektPROMRechts", "
70                 DorsiflexiegestrektAOCRechts"])
71         results.append([f"Afgenomen dorsaalflexie ({str(np.round(
72             ankle_degrees))} graden)",
73             side, joint, row.Foot, row.Event])
74     elif ankle == 1:
75         # Increased dorsiflexion
76         results.append([f"Toegenomen dorsaalflexie ({str(np.round(
77             ankle_degrees))} graden)",
78             side, joint, row.Foot, row.Event])
79     else:
80         # Ankle range of motion within normal range
81         results.append([f"Enkel range of motion ({str(np.round(
82             ankle_degrees))} graden) binnen "
83             f"range van normaal",
84             side, joint, row.Foot, row.Event])
85 # Diagnosis for the swing phase
86 for index, row in df_swing.iterrows():
87     # Access the associated joint angle with the side
88     ankle = row['LAnkle'] if side != 'Left' else row['RAnkle']
89     ankle_degrees = row['LAnkle Degrees'] if side != 'Left' else row['
90         RAnkle Degrees']
91     if row.Event == 'Mid Stance':
92         if ankle == -1:
93             # Plantarflexion other foot
94             if side == 'Left':

```

```

85         # Extract relevant variables for the LO
86         lo_variables.extend(["DorsiflexieMRCLinks", "
87                             DorsiflexiegebogenPROMLinks",
88                             "DorsiflexiegebogenAOCLinks", "
89                             DorsiflexiegestrektPROMLinks",
90                             "DorsiflexiegestrektAOCLinks"])
91     else:
92         # Extract relevant variables for the LO
93         lo_variables.extend(["DorsiflexieMRCRechts", "
94                             DorsiflexiegebogenPROMRechts",
95                             "DorsiflexiegebogenAOCRechts", "
96                             DorsiflexiegestrektPROMRechts"
97                             ,
98                             "DorsiflexiegestrektAOCRechts"])
99     results.append([f"Plantarflexie andere voet ({str(np.round
100                    (ankle_degrees))} graden)",
101                    side, joint, row.Foot, row.Event])
102 else:
103     # No relevant finding in this case
104     results.append(["Geen relevante bevindingen", side,
105                    joint, row.Foot, row.Event])
106 elif row.Event == 'Terminal Stance':
107     if ankle == -1:
108         # Plantarflexion other foot
109         if side == 'Left':
110             lo_variables.extend(["DorsiflexieMRCLinks", "
111                                 DorsiflexiegebogenPROMLinks", "
112                                 DorsiflexiegebogenAOCLinks",
113                                 "DorsiflexiegestrektPROMLinks", "
114                                 DorsiflexiegestrektAOCLinks"])
115         else:
116             lo_variables.extend( ["DorsiflexieMRCRechts", "
117                                 DorsiflexiegebogenPROMRechts", "
118                                 DorsiflexiegebogenAOCRechts",
119                                 "DorsiflexiegestrektPROMRechts", "
120                                 DorsiflexiegestrektAOCRechts"])
121         results.append([f"Plantarflexie andere voet ({str(np.round
122                    (ankle_degrees))} graden)",
123                    side, joint, row.Foot, row.Event])
124     else:
125         # No relevant finding in this case
126         results.append(["Geen relevante bevindingen", side,
127                    joint, row.Foot, row.Event])
128 # Return the results and the list of relevant variables
129 return pd.DataFrame(results, columns=result_labels), lo_variables

```

Listing 1: Python function to diagnose the ankle joint based on gait data

Knee Joint Diagnosis Decision Trees

```

1 def diagnose_knee(data, side):
2     df_stance = data[data['Foot'] == side]
3     df_swing = data[data['Foot'] != side]
4     joint = "Knee"
5     results = []
6     lo_variables = []
7     knee_midstance_threshold = -4
8
9     # Diagnosis for the stance phase
10    for index, row in df_stance.iterrows():
11        knee = row['LKnee'] if side == 'Left' else row['RKnee']
12        knee_degrees = row['LKnee Degrees'] if side == 'Left' else row['
13            RKnee Degrees']
14        if row.Event == 'Foot Strike':
15            if knee == 1:
16                if side == 'Left':
17                    # Extract relevant variables for the LO
18                    lo_variables.extend(["Pop-hoekPROMLinks", "Pop-
19                        hoekAOCLinks", "Knie-extensiePROMLinks"])
20                else:
21                    # Extract relevant variables for the LO
22                    lo_variables.extend(["Pop-hoekPROMRechts", "Pop-
23                        hoekAOCRechts", "Knie-extensiePROMRechts"])
24                # Increased knee flexion
25                results.append([f"Toegenomen knieflexie ({str(np.round(
26                    knee_degrees))} graden)", side,
27                    joint, row.Foot, row.Event])
28            else:
29                # No relevant finding in this case
30                results.append(["Geen relevante bevindingen", side,
31                    joint, row.Foot, row.Event])
32        elif row.Event == 'Loading Response':
33            if knee == 1:
34                # Increased knee flexion
35                results.append([f"Toegenomen knieflexie ({str(np.round(
36                    knee_degrees))} graden) tijdens", side,
37                    joint, row.Foot, row.Event])
38            elif knee == -1:
39                if knee_degrees >= 0:
40                    # TODO: Knee Moment Saggital
41                    # Decreased knee flexion
42                    results.append([f"Afgenomen knieflexie ({str(np.round(
43                    knee_degrees))} graden) tijdens", side,
44                    joint, row.Foot, row.Event])
45                else:
46                    # Knee Hyperextension
47                    results.append([f"Knie-extensiemomenten tijdens",
48                    side,
49                    joint, row.Foot, row.Event])
46            else:
47                results.append(["Geen relevante bevindingen tijdens",
48                    side,

```

```

45         joint, row.Foot, row.Event])
46     elif row.Event == 'Mid Stance':
47         if knee == 1:
48             # Increased knee flexion
49             if side == 'Left':
50                 # Extract relevant variables for the LO
51                 lo_variables.extend(["Knie-extensiePROMLinks", "
                    HeupextensiePROMLinks"])
52             else:
53                 # Extract relevant variables for the LO
54                 lo_variables.extend(["Knie-extensiePROMRechts", "
                    HeupextensiePROMRechts"])
55             results.append([f"Toegenomen knieflexie ({str(np.round(
                    knee_degrees))} graden)", side,
56                             joint, row.Foot, row.Event])
57         elif knee == -1:
58             if side == 'Left':
59                 # Extract relevant variables for the LO
60                 lo_variables.extend(["DorsiflexiegebogenPROMLinks", "
                    DorsiflexiegebogenAOCLinks",
61                                     "DorsiflexiegestrektPROMLinks", "
                    DorsiflexiegestrektAOCLinks"])
62             else:
63                 # Extract relevant variables for the LO
64                 lo_variables.extend(["DorsiflexiegebogenPROMRechts", "
                    DorsiflexiegebogenAOCRechts",
65                                     "DorsiflexiegestrektPROMRechts", "
                    DorsiflexiegestrektAOCRechts"])
66             if knee_degrees > knee_midstance_threshold:
67                 # Decreased knee flexion
68                 #TODO: Knee Saggital Moment
69                 results.append([f"Afgenomen knieflexie ({str(np.
                    round(knee_degrees))} graden)", side,
70                                 joint, row.Foot, row.Event])
71             else:
72                 # Knee Hyperextension
73                 # TODO: Knee moment Saggital
74                 results.append([f"Kniehyperextensie ({str(np.round(
                    knee_degrees))} graden)", side,
75                                 joint, row.Foot, row.Event])
76         else:
77             # No relevant finding in this case
78             results.append(["Geen relevante bevindingen", side,
79                             joint, row.Foot, row.Event])
80     elif row.Event == 'Terminal Stance':
81         if knee == 1:
82             # Increased knee flexion
83             #TODO: Knee Saggital Moment
84             if side == 'Left':
85                 # Extract relevant variables for the LO
86                 lo_variables.extend(["Knie-extensiePROMLinks", "
                    HeupextensiePROMLinks"])
87             else:
88                 # Extract relevant variables for the LO

```

```

89         lo_variables.extend(["Knie-extensiePROMRechts", "
90             HeupextensiePROMRechts"])
91     results.append([f"Toegenomen knieflexie ({str(np.round(
92         knee_degrees))} graden)", side,
93         joint, row.Foot, row.Event])
94 elif knee == -1:
95     # Knee hyperextension
96     if side == 'Left':
97         # Extract relevant variables for the LO
98         lo_variables.extend(["DorsiflexiegebogenPROMLinks", "
99             DorsiflexiegebogenAOCLinks",
100             "DorsiflexiegestrektPROMLinks", "
101             DorsiflexiegestrektAOCLinks"])
102     else:
103         # Extract relevant variables for the LO
104         lo_variables.extend(["DorsiflexiegebogenPROMRechts", "
105             DorsiflexiegebogenAOCRechts",
106             "DorsiflexiegestrektPROMRechts", "
107             DorsiflexiegestrektAOCRechts"])
108     results.append([f"Kniehyperextensie ({str(np.round(
109         knee_degrees))} graden)", side,
110         joint, row.Foot, row.Event])
111 else:
112     # No relevant finding in this case
113     results.append(["Geen relevante bevindingen", side,
114         joint, row.Foot, row.Event])
115 # Diagnosis for the swing phase
116 for index, row in df_swing.iterrows():
117     knee = row['LKnee'] if side == 'Left' else row['RKnee']
118     knee_degrees = row['LKnee Degrees'] if side == 'Left' else row['
119         RKnee Degrees']
120     if row.Event == 'Foot Strike':
121         if knee == -1:
122             # Decreased knee flexion other foot
123             if side == "Left":
124                 # Extract relevant variables for the LO
125                 lo_variables.extend(["Duncan-ElyPROMLinks", "Duncan-
126                 ElyAOCLinks"])
127             else:
128                 # Extract relevant variables for the LO
129                 lo_variables.extend(["Duncan-ElyPROMRechts", "Duncan-
130                 ElyAOCRechts"])
131             results.append([f"Afgenomen knieflexie ({str(np.round(
132                 knee_degrees))} graden) bij "
133                 f"andere voet", side, joint, row.Foot,
134                 row.Event])
135         else:
136             # No relevant finding in this case
137             results.append(["Geen relevante bevindingen", side, joint
138                 , row.Foot, row.Event])
139 # Return the results and the list of relevant variables
140 return pd.DataFrame(results, columns=result_labels), lo_variables

```

Listing 2: Python function to diagnose the knee joint based on gait data

Hip Joint Diagnosis Decision Trees

```

1 def diagnose_hip(data, side):
2     df_stance = data[data['Foot'] == side]
3     joint = "Hip"
4     results = []
5     lo_variables = []
6
7     # Diagnosis for the stance phase
8     for index, row in df_stance.iterrows():
9         if row.Event in ['Foot Strike', 'Loading Response', 'Mid Stance']:
10            # Access the associated joint angle with the side
11            hip = row['LHip'] if side == 'Left' else row['RHip']
12            if hip == 1:
13                if row.Event == 'Mid Stance':
14                    # Increased hip flexion
15                    if side == 'Left':
16                        # Extract relevant variables for the LO
17                        lo_variables.extend(["Knie-extensiePROMLinks", "
18                            HeupextensiePROMLinks"])
19                    else:
20                        # Extract relevant variables for the LO
21                        lo_variables.extend(["Knie-extensiePROMRechts", "
22                            HeupextensiePROMRechts"])
23                results.append([f"Toegenomen heupflexie", side, joint, row
24                    .Foot, row.Event])
25            elif hip == -1:
26                # Decreased hip flexion
27                results.append([f"Afgenomen heupflexie", side, joint, row.
28                    Foot, row.Event])
29            else:
30                # No relevant finding in this case
31                results.append([f"Geen relevante bevindingen tijdens",
32                    side, joint, row.Foot, row.Event])
33            if row.Event == 'Terminal Stance':
34                hip = row['LHip'] if side == 'Left' else row['RHip']
35                if hip == 1:
36                    # No hip extension
37                    if side == 'Left':
38                        # Extract relevant variables for the LO
39                        lo_variables.extend(["Knie-extensiePROMLinks", "
40                            HeupextensiePROMLinks", "HeupextensieMRCLinks"])
41                    else:
42                        # Extract relevant variables for the LO
43                        lo_variables.extend(["Knie-extensiePROMRechts", "
44                            HeupextensiePROMRechts", "HeupextensieMRCRechts"])
45                results.append([f"Geen Heupextensie", side, joint, row.
46                    Foot, row.Event])
47            else:
48                # No relevant finding in this case
49                results.append([f"Geen relevante bevindingen tijdens",
50                    side, joint, row.Foot, row.Event])
51            # Return the results and the list of relevant variables
52            return pd.DataFrame(results, columns=result_labels), lo_variables

```

Listing 3: Python function to diagnose the hip joint based on gait data