LEARNING HEALTH INFORMATION FROM FLOOR SENSOR DATA WITHIN A PERVASIVE SMART HOME ENVIRONMENT

by

NICHOLAS BRENT BURNS

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

September 2020

Supervising Committee:

Gergely Záruba, Supervising Professor
Manfred Huber, Co-Supervising Professor
Farhad Kamangar
Kathryn Daniel

For Mom.

ACKNOWLEDGEMENTS

I would like to thank my supervising and co-supervising professors Dr. Gergely Záruba and Dr. Manfred Huber. Their welcoming nature made my graduate experience truly wonderful.

Dr. Záruba taught me how to solder properly, gave me a tremendous amount of electronic knowledge, gave me a fun teaching opportunity, took me to faraway conferences, and informed me that you do not bowl on Shabbos. The best mentor one could ask for.

Dr. Huber’s advice and direction was absolutely critical in navigating the more difficult aspects of research. His endless knowledge of seemingly all computer science topics along with his amazing personality might make him the most valuable resource at UTA. I am truly indebted to your help, especially from the last year. Thank you for all you taught me.

The SmartCare project would not have been possible without the work of Dr. Kathryn Daniel. Her behind the scenes work and organization was invaluable for the project. Thank you for being a great and easy person to talk with.

I would like to thank Dr. Farhad Kamangar for offering his time and expertise on my committee and providing extremely useful and interesting classes.

I would like to thank Dr. Darin Brezeale for introducing me to the CSE Department’s undergraduate and graduate programs, teaching some of my favorite courses, and being a great person to talk with.

I would also like to thank Bito Irie for all the great conversations, advice, and random run-ins over the years.

Most importantly, I would like to thank my mom, Pam. Your never-ending support and optimism makes everything worth it. I love you.

August 21, 2020

ABSTRACT

LEARNING HEALTH INFORMATION FROM FLOOR SENSOR DATA

WITHIN A PERVASIVE SMART HOME ENVIRONMENT

Nicholas Brent Burns, Ph.D.

The University of Texas at Arlington, 2020

Supervising Professor: Gergely Záruba

Spatial and temporal gait analysis can provide useful measures for determining a person's state of health while also identifying deviations in day-to-day activity. The SmartCare project is a multi-discipline health technologies project that aims to provide an unobtrusive and pervasive system that provides in-home health monitoring for the elderly. This research work focuses on the pressure-sensitive smart floor of the SmartCare project by using an experimental floor to develop methods for future use on a floor deployed within a home.

This work presents a procedure to automatically calibrate a smart floor's pressure sensors without specialized physical effort. The calibration algorithm automatically filters out non-human static weight and only retains weight generated by human activity. This technique is designed to correctly translate sensor values to kg weight units even when direct independent access to the pressure sensors is prohibited and when a shared tile floor sits above the sensor grid.

Using the calibrated sensor values, machine learning techniques are used to extract individual contact points on a smart floor generated by a person's walking cycle. This work presents a three-step process of building and training neural network models of different architectures (feedforward, convolutional, and autoencoder) to learn the unique non-linear relationship between weight distribution, tile coupling, and physical floor variations.

Finally, this research work presents a recursive Hierarchical Clustering Analysis algorithm that uses the individual contact points generated by the floor model to extract individual footfalls of a person during a walking cycle. The footfall clusters are further grouped and segmented into walking sequences. Spatial gait analysis is performed on the resulting footfall clusters within each walking sequence to measure a variety of gait parameters. The results of the gait analysis are compared to those generated by a high-resolution mat alternative showing comparable results for most of the computed gait metrics.

Table of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

1 Overview

1.1 Introduction

Advances in medical treatment and technology have increased the average life expectancy considerably across the developed world, resulting in a larger elderly population. Falling is a significant health risk for those over the age of 65. How well a person walks, and thus their gait characteristics, can be a good indicator of how likely they might be to experience a tragic fall in the future. Gait analysis also has applications in detecting early signs or diagnosing various diseases and conditions [1]. Traditionally gait analysis was performed by a health professional visually and in-person with a patient. Advances in computing and sensing technology have introduced a more analytical and consistent measurement of how well a person walks. These technologies range from high-resolution floor mats, video/image processing, depth camera analysis, and wearable sensors. However, these are rarely used in the patient’s home but rather in health facilities in a highly controlled environment. Embedded pervasive systems integrated within the home of an older individual could offer better insight into their gait characteristics and overall health. Not only could these in-home systems detect obvious emergencies, they could offer predictive and preventative services to avoid serious health events in the future.

The SmartCare project is a multi-discipline health technologies project between the Nursing and Computer Science and Engineering departments at the University of Texas at Arlington. SmartCare offers an unobtrusive and pervasive system that provides in-home health monitoring for the elderly. Using a range of embedded sensing technologies along with hardware, software, and communication infrastructure within one’s home allows continuous health monitoring and alleviates the burden on older individuals in providing reliable activity and health information to care providers and loved ones. The SmartCare system can provide early diagnosis support for medical professionals, continuous activity monitoring, deviation detection, and self-management assistance in the form of home automation and medication intake reminders.

1.2 Statement of Contributions

The specific research in this dissertation is at a system level and includes hardware, middleware, system integration, and advanced software aspects to address the question of how to measure the quality of one's gait in their own home in a non-obtrusive, passive, and effective manner. The specific contributions detailed in this dissertation regard smart floor data calibration, extraction, and learning.

The first contribution is an automatic calibration technique for grouped sensors of a smart floor that does not require rigid and specific physical effort of an expert technician. Rather, a software approach is introduced here that utilizes normal everyday walking data to calibrate the undefined units of pressure sensors into known kg units to make all sensors across the entire floor uniformly reliable in measuring weight. The calibration technique also automatically detects and accounts for non-human static weight that can be filtered out with only human weight and activity remaining. Due to the calibrated sensor units not being sensitive to everchanging static weight on the floor, this calibration technique is useful for a home fitted with this smart floor where furniture and other objects may be rearranged over time. No special accommodations are needed for the smart floor within a home; it is treated as a normal unobtrusive tile floor.

The second contribution is a multi-stage neural network model that extracts single contact points from a person's footfalls using the calibrated sensor units. Even in the low-resolution environment of the smart floor (1-sqft sensor spacing) and in the presence of multiple contact points for that person or the presence of multiple individuals, the model can accurately extract individual moments of foot contact at any location across the tiled smart floor. The model learns and accounts for the unique imperfections associated with shifting and flexing tile movement which rests upon the floor sensors. The model learns the non-linear relationships and how weight spreads (pressure profiles) across all areas of the smart floor.

The final contribution is a method to cluster and segment a person's individual footfalls using the model's single contact point output data. Using a recursive Hierarchical Clustering Analysis (HCA) technique, sequential footfalls are extracted while removing outlier contact points. The resulting footfall clusters are further grouped and sorted into walking segments to allow the calculation and measurement of various gait parameters of a person. The gait and walking features are used to classify and identify different people from one another.

1.3 Outline

Chapter 2 discusses the current state of in-home healthcare methods and the uniqueness of SmartCare's smart floor while also providing background for key methods and algorithms used throughout this dissertation. Chapter 3 provides the details of the SmartCare project's sensor-rich nursing home apartment. Chapter 4 describes the process of floor sensor calibration. Chapter 5 discusses the model architecture and procedure for extracting a person's contact points when walking across the smart floor. Chapter 6 details how individual footfalls are clustered over time and how the extracted footfalls are used in gait analysis. Chapter 7 compares the accuracy of the gait measurements from the low-resolution smart floor against those of a high-resolution walking mat and demonstrates a preliminary effort of person identification (classification) using smart floor gait data. Chapter 8 summarizes the conclusions of this work and describes how these methods could be used in the future for application within the SmartCare apartment.

2 BACKGROUND AND RELATED WORK

2.1 In-Home Healthcare Monitoring and Smart Floors

Two prominent research projects regarding smart home environments for the elderly are ongoing at the University of Missouri's AgingMO Tiger Place [2] and Washington State University's Center for Advanced Studies in Adaptive Systems (CASAS) [3]. These two projects have produced gait and activity studies collecting data via personal scoring, video analysis, image processing, IR sensors, wearable sensors, motion sensors, and pressure floor mats [4] [5].

There are three main tools used to generate data for gait analysis: video recording, wearable sensors, and floor sensors. Past studies have used shallow [6] and deep learning [1] [7] [8] techniques to extract and learn gait parameters from one or a combination of these three data collection methods. In particular, Support Vector Machines, Linear Discriminant Analysis, Random Forest, Convolutional Neural Networks, Recurrent Neural Networks, Long Short-Term Memory Networks, and Deep Belief Networks have been used in these prior projects. These papers offer great insight into clever methods of fusing different data collection mediums to improve the accuracy of their model's output in detecting gait abnormalities and person identification.

There are three main technologies used for capturing walking data from a smart floor environment: 1) optical and visual, 2) capacitive touch, and 3) pressure measurement. An optical method such as GravitySpace extracts detailed location data and movements but does not provide pressure information and thus cannot calculate weight or impact forces of a person's walking pattern [9]. While capacitive methods such as SensFloor provide accurate location data with simple integration into existing flooring, it also suffers from the lack of pressure and weight sensing [10]. Pressure-based smart floors, such as the ASU Pressure Mat Floor, offer high-resolution 1cmx1cm data [11]. While the sensing technology can provide detailed information of individual footfalls and walking patterns, it is prohibitively expensive, non-scalable, susceptible to damage, and not suitable for deployment over an entire home. A low-resolution pressure-sensitive smart floor such as the EMFi floor provides a more reasonable cost by only having a resolution of 30cmx30cm [12] [13]. However, due to the flooring structure above the sensing materials and how sensors interact, the system cannot resolve center of pressure readings and therefore cannot provide reliable measurements of gait and balance analysis. Using a rigid flooring surface can help resolve the need for pressure distribution variations required to calculate center of pressures and walking profiles.

The uniqueness of the SmartCare project, in terms of the floor, is its size, coverage, rigid tile surface, and low-cost. Other floor sensor projects either use off-the-shelf high-resolution mats or small custom-built smart floors that only exist in labs. Deploying enough high-resolution mats to cover the entire area of the SmartCare apartment is cost prohibitive. While the SmartCare floor requires specialized knowledge and labor to create and install, the price tag is magnitudes cheaper than purchasing enough high-resolution mats to cover an equal amount of space. Also, the large sensor-array floor is deployed within an actual living environment, not just a lab solely used for experiments. The smart floor captures real-time data of individuals living their normal lives within a typical apartment. This helps eliminate or reduce the white coat hypertension effect someone may experience if asked to walk on a floor subsection within a lab or health facility while being analyzed.

Initial SmartCare floor research has already began with the works of Suhas Reddy [14] and Oluwatosin Oluwadare [15]. Their work involved extracting gait parameters from the smart floor for general health monitoring, person identification, and anomaly detection.

2.2 Gait Analysis

Human gait analysis is the study of how a person walks. The repetitive walking pattern a person performs for locomotion is referred to as the gait cycle. The gait cycle consists of two separate phases: the stance phase and the swing phase [16]. The stance phase contains five actions: heel-strike, foot-flat, midstance, heel-off, and toe-off. The swing phase contains four actions: pre-swing, initial swing, mid-swing, and terminal swing.

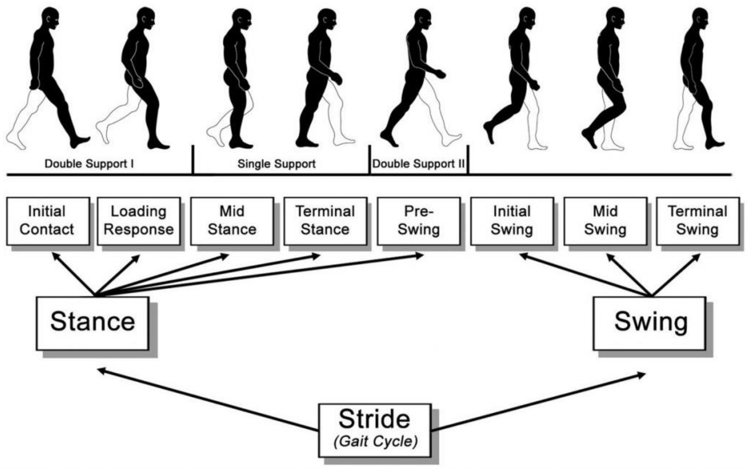


Figure 2.1: Gait Cycle Phases [17]

From a spatial point of view, gait parameters can be measured that relate to footfall locations. A person’s left and right step length, step width, stride length, and foot/step angles can be calculated. Also, parameters involving time can be learned such as step time, stride time, step speed, and stride speed. These spatial parameters are not easily obtainable when observing a person’s gait from the perspective shown in Figure 2.1. Therefore, some type of sensing mat or floor is required to accurately capture these gait features.

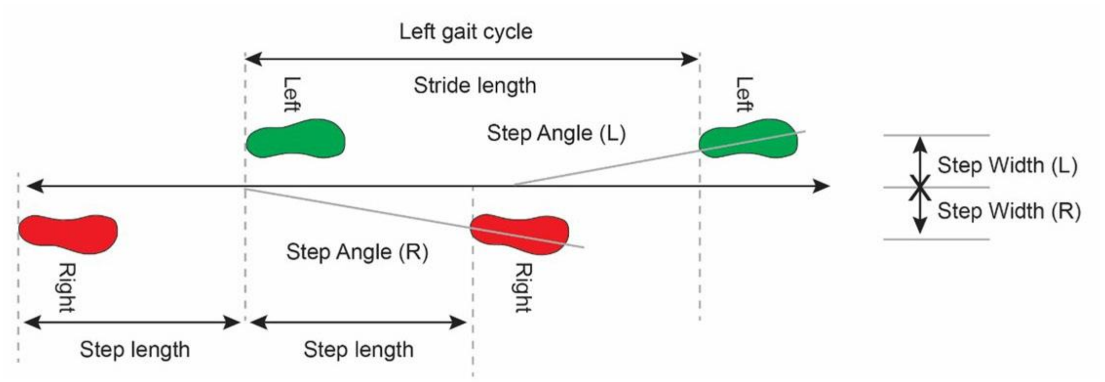


Figure 2.2: Bird's Eye View of Footfalls for Gait Analysis [18]

2.3 Hierarchical Clustering Analysis

Hierarchical Clustering Analysis (HCA) is an unsupervised method of grouping data points into a hierarchy of clusters [19]. HCA investigates the similarity of training samples and their features. There are no labels associated with the sample data points and they can be of any dimension. HCA is a great tool for inspecting unlabeled data and discovering similarity relationships amongst data. An advantage of HCA over another popular clustering method, *k*-means, is that it is not required to specify the number of clusters to be found.

There are two main strategies for HCA: agglomerative and divisive. Agglomerative is the “bottom-up” approach where each data sample begins in its own cluster and throughout the algorithm clusters are merged repeatedly until only a single large cluster remains. Divisive is the “top-down” approach where all samples are grouped within a single cluster at the start and throughout the process are split continually until every sample point belongs within its own unique cluster. The results of HCA can be viewed in the form of a dendrogram to visually and analytically evaluate how training sample points clustered together throughout the procedure.

To determine the similarity between sample points or clusters, a distance metric must be specified. Commonly used measures include Euclidean (l_2 norm), Manhattan (l_1 norm), and Cosine distances [20].

Name	**Formula**
Euclidean Distance	$D(A,B) = \sqrt{\sum_i (a_i - b_i)^2}$
Manhattan Distance	$D(A,B) = \sum_i \lvert a_i - b_i \rvert$
Cosine Similarity	$D(A,B) = \frac{A \cdot B}{\lvert\lvert A \rvert\rvert \, \lvert\lvert B \rvert\rvert}$

Table 2-1: Clustering Distance Metrics

Regardless of which distance metric is used to compare data points, a specific method must be chosen of how to compare a set of points (cluster) to another, generally referred to as a linkage type. Some of the most common methods are single linkage, average linkage, complete linkage, and ward linkage [21]. When using single linkage, also called minimum linkage, the distance between two clusters is the shortest distance between two data points from the clusters, one from each. Using single linkage can result in the chaining of clustered data points. Depending on the application, this chaining effect can be beneficial when dealing with sequential temporal data (i.e. footstep data). With average linkage, the distance between two clusters is the unweighted mean distance of every data point from one cluster to every data point of another. Complete linkage, also called maximum linkage, defines the distance between two clusters as the longest distance between any two data points in the clusters. Ward linkage aims to minimize the variance within each cluster's set of points; how much the set of points vary from the mean of the cluster.

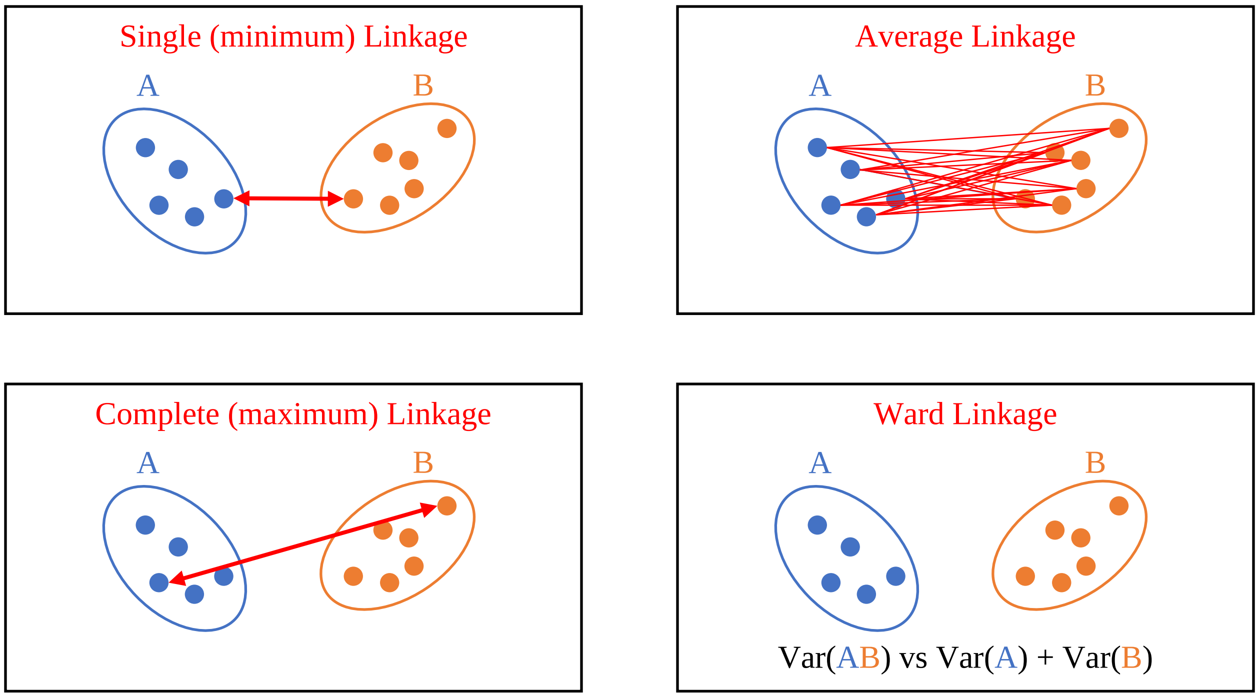


Figure 2.3: Different HCA Linkage Criteria

When HCA is performed, the entire linkage or distance matrix is computed. When viewing the resulting hierarchy, it is up to the individual to decide how many clusters best represent the desired result. When viewing the results through a dendrogram plot, a horizontal "cut" along the tree can be made at any height giving a potential cluster range of *n* to 2. With agglomerative, bottom-up clustering, any cluster merges above the cut line are ignored while only retaining the clusters formed below. The decision of where to place this cut line can be handcrafted or based off some strict automatic method. Since HCA is generally used for data exploration, there is not a definitive method for deciding where the "best cut" should be placed. A common method, when visualizing the linkage matrix as a dendrogram, is to find the max vertical distance between any cluster merger throughout the entire hierarchy. The middle *y*-axis value is found within the vertical distance span and becomes the final "best cut" line. Figure 2.4 contains a visual example.

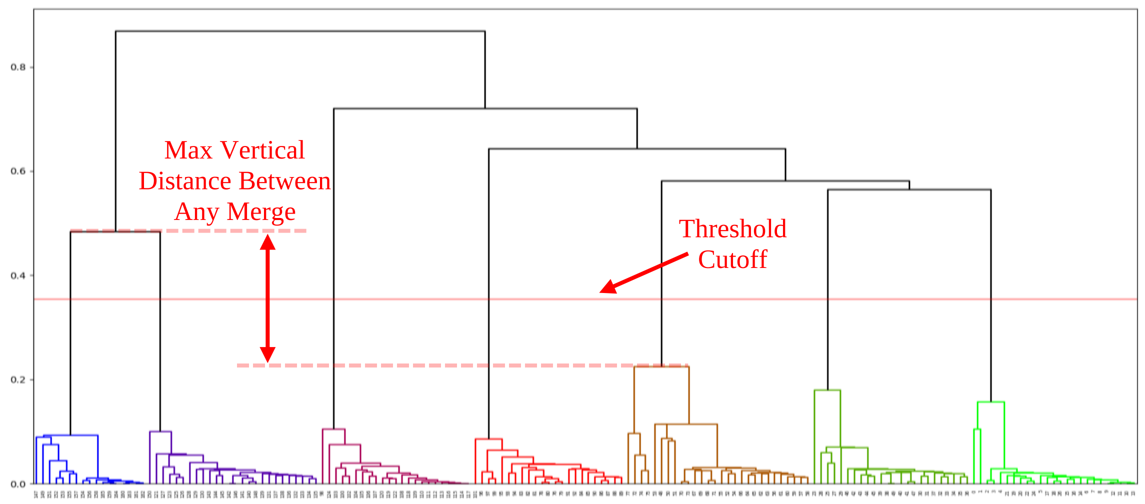


Figure 2.4: Dendrogram Best Cut Line Resulting in 7 Clusters

Figure 2.5 shows a scatter plot of contact point walking data. The data is not clustered and only contains the spatial features of *x* and *y* clearly showing six unique footsteps. Using the “best cut” method described previously, Figure 2.6 andFigure 2.7 show the results of agglomerative hierarchical clustering using the Euclidean distance metric on the *x* and *y* features with four different linkage types: single, average, complete, and Ward. Average, complete, and Ward all resulted in two final clusters. Single linkage, in this instance, resulted in the desired six clusters due to the chaining effect of joining points and clusters of the shortest distance. If a time feature was added for each contact point the single linkage method might perform even more reliably in the case of a person walking back to the same location over time.

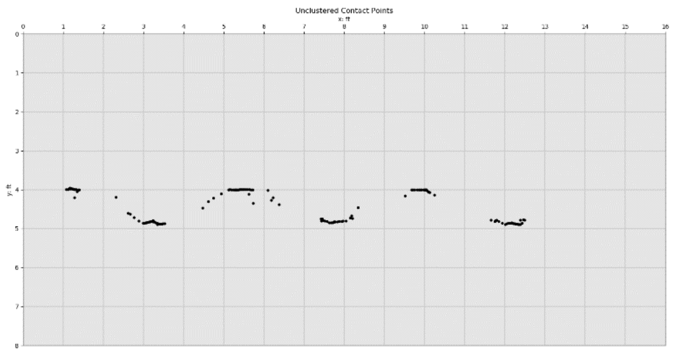


Figure 2.5: Scatter Plot of Raw Unclustered Contact Points

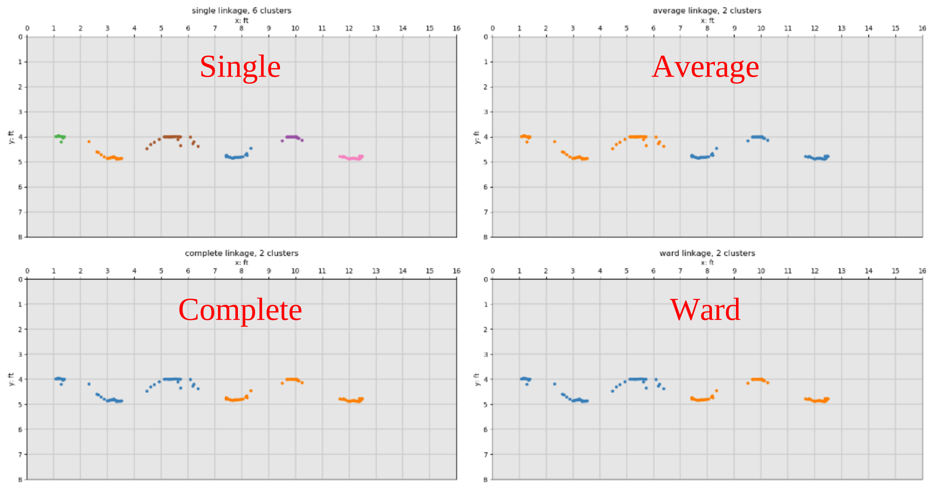


Figure 2.6: Scatter Plots of HCA Cluster Results of Different Linkage Types (color = label)

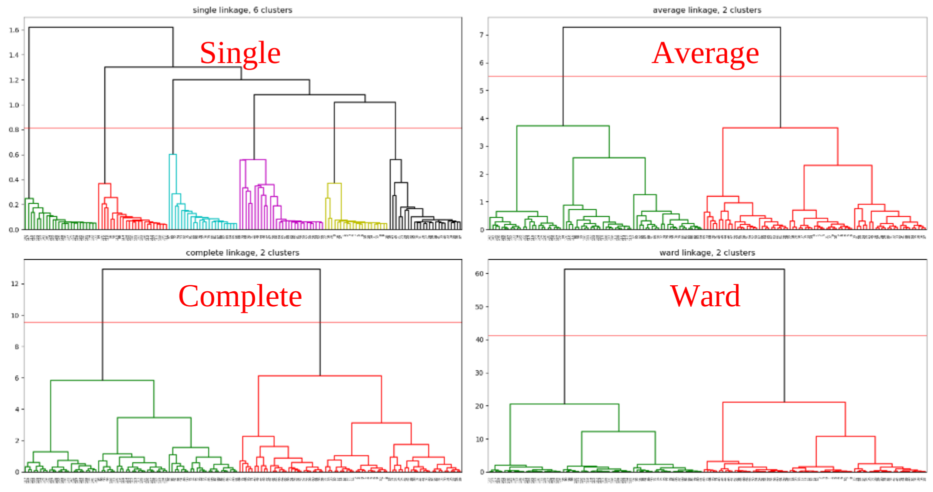


Figure 2.7: Dendrograms of HCA Cluster Results of Different Linkage Types (color = label)

2.4 Moving Average

Calculating the moving average of temporal data can help smooth out short-term fluctuations, spikes, and valleys [22]. By removing this noise, the data's general trend can be more easily understood. Techniques differ in terms of weighting schemes, window size, and cumulative contributions.

2.4.1 Simple Moving Average

A simple moving average (SMA) or sliding window average computes the unweighted mean of *n* sequential samples, with *n* being the size of the window. Since all sample values are unweighted, they each have an equal contribution to the SMA value. The window can be before, after, or equally split about the central value in question.

$$\frac{1}{n}\sum_{i=0}^{n-1} x_i$$

2.4.2 Exponential Moving Average

An exponential moving average (EMA) applies a weighting factor and incorporates the previously calculated EMA. The value of the weighting or smoothing factor α determines the smoothness of the average trend and how much influence past averages have on the new calculation. A higher α decreases past observation influence faster, while a lower α allows their influence to linger longer and results in a smoother result. Also, a higher α can allow signal noise to have a greater influence. The first EMA calculated is simply the first data sample's value.

$$EMA_t \leftarrow \begin{cases} x_0 & if\ t = 0 \\ \alpha x_t + (1-\alpha)EMA_{t-1} & if\ t > 0 \end{cases}$$

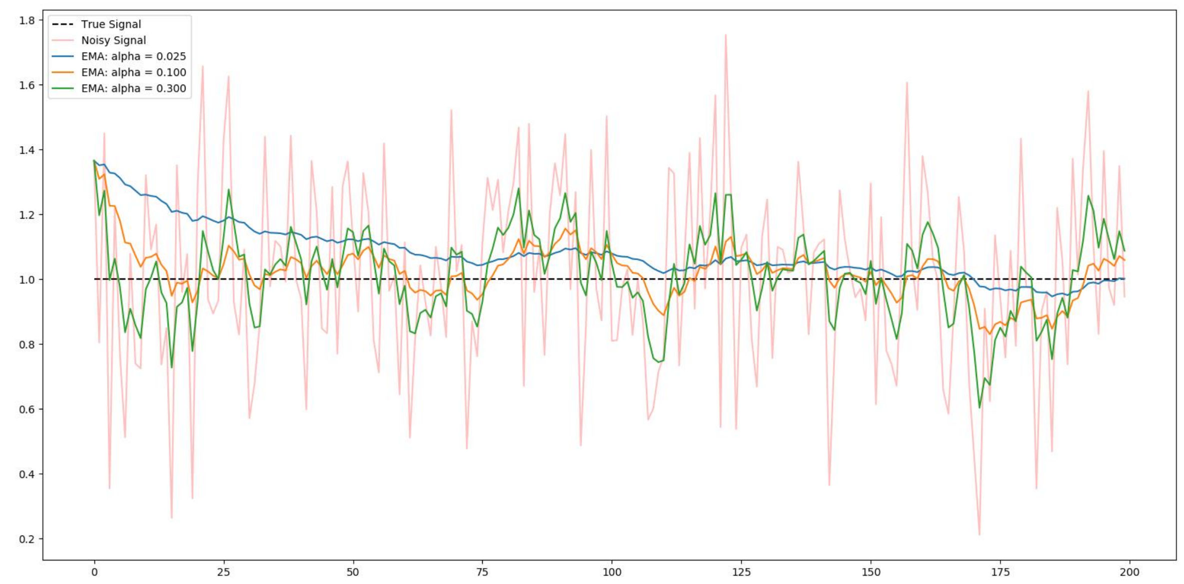


Figure 2.8: Different EMA Alpha Smoothing Factors Applied to Noisy Data

3 SMARTCARE APARTMENT INFRASTRUCTURE

3.1 Overview

The SmartCare apartment is located at Lakewood Village Retirement Community in Fort Worth, Texas. The interior has the appearance of a normal apartment with the latest high-end appliances and fixtures, but embedded under the floor, in the ceiling, and in the walls are various sensing and automation technologies to provide 24/7 health monitoring for the elderly. It is a live-in laboratory to run various experiments and gather short-term and long-term data. The apartment's unveiling was in May 2015.

The apartment infrastructure was designed and constructed under the supervision of Dr. Záruba, Dr. Huber, and Dr. Daniel. Throughout this work, Nicholas Burns was instrumentally involved in all aspects, including cleaning, construction, product research, hardware and sensor installation, software design, and networking which made the smart apartment a reality.

From the recruitment efforts of Dr. Daniel, residents of the Lakewood Village Retirement Community have volunteered to live in our SmartCare apartment. From May 2017 to May 2019 various single and coupled residents have stayed in our apartment for various lengths of time; usually about one month. During these stays we have recorded 24/7 data from the smart floor, IR sensors, door sensors, water usage, electricity activity, etc. This semi long-form data will hopefully provide insight into building a model of a resident's activities and learning health information. We also had a resident stay twice at our apartment with a significant time gap in between, hopefully this data can identify any differences this resident underwent over that period.

3.2 Embedded Technologies

The apartment’s technologies include a smart pressure-sensitive floor, Z-Wave sensors and actuators, home automation devices, high-resolution bed mats, water/electricity monitoring and control all fed to a computer system that runs custom-built software to manage the hundreds of sensors and data collection. The hardware, software, visualization, and infrastructure details are explained in our previous papers [23] [24]. The visualization in [24] was through the hard work of Peter Sassaman.

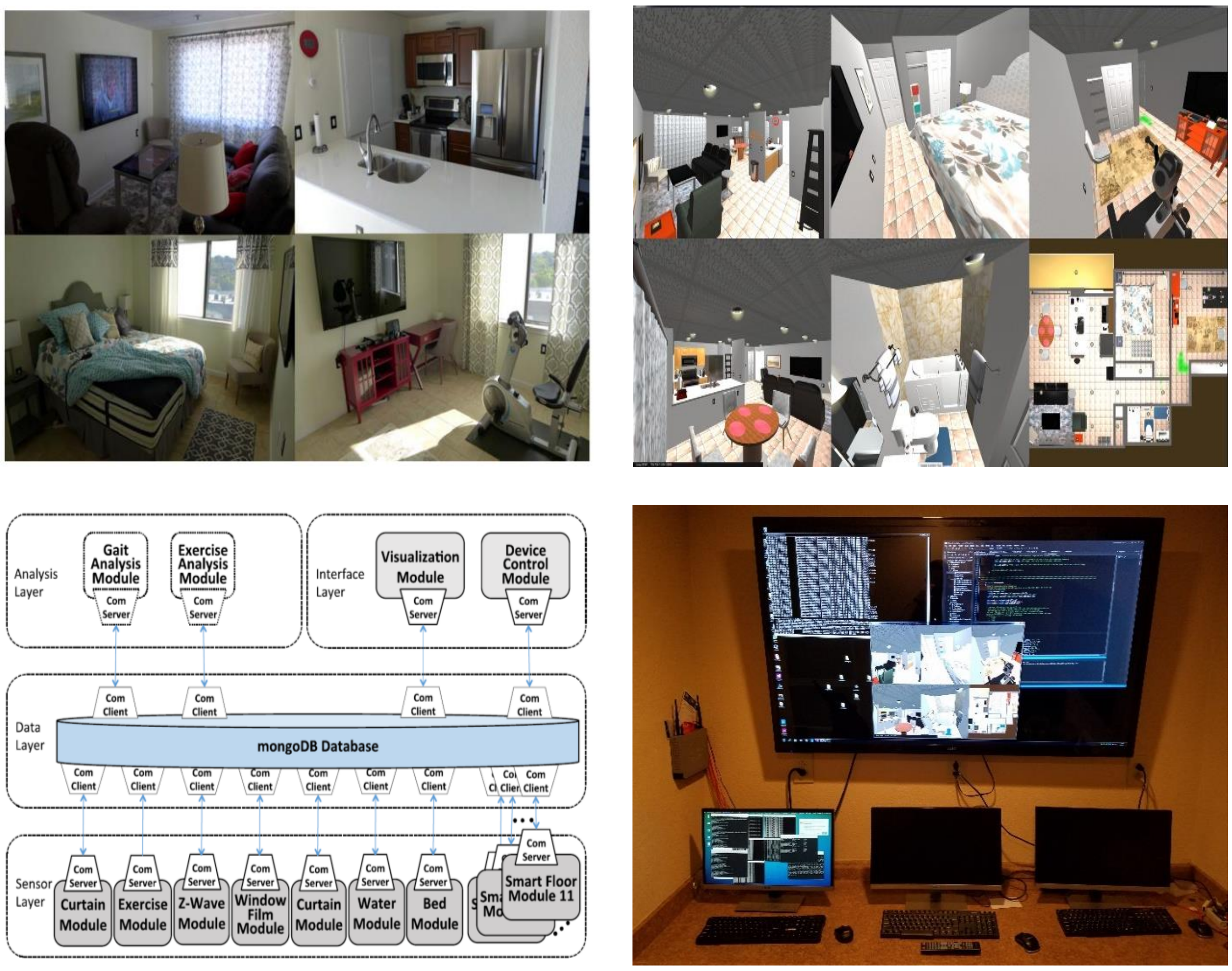


Figure 3.1: SmartCare Apartment Interior, Visualization, System Architecture Overview, and Computer Room

3.3 Apartment Smart Floor

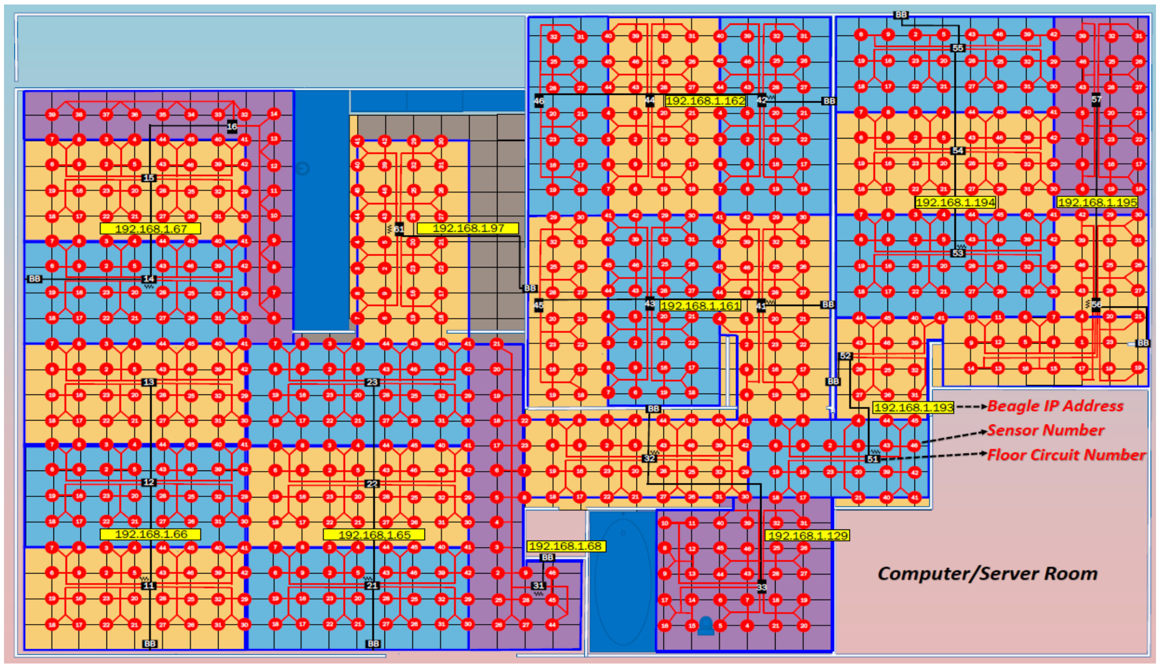


Figure 3.2: SmartCare Apartment Tile and Floor Sensor Layout

The physical construction of the apartment floor and the lab floor (described in Section 4) was completed by Dr. Gergely Záruba, Dr. Manfred Huber, and Nicholas Burns. Embedded circuitry, embedded software, middleware software, high-level software, and networking connectivity were designed and built by Drs. Záruba and Huber.

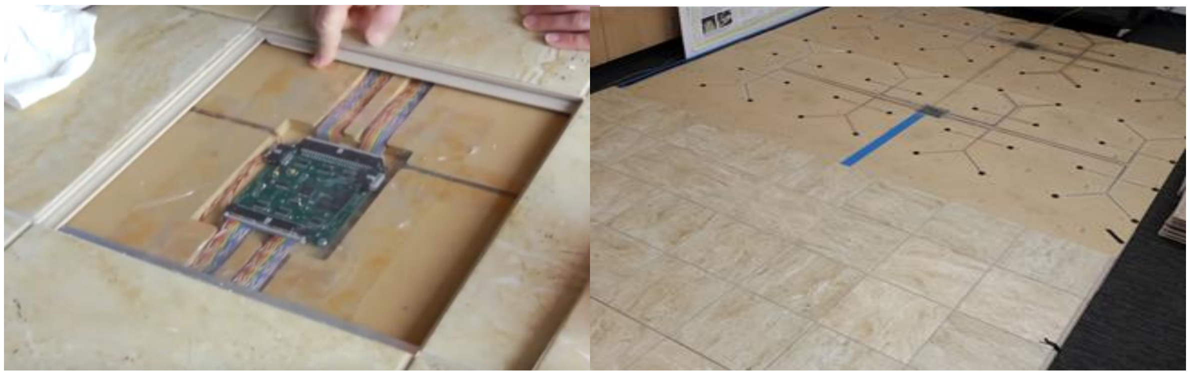


Figure 3.3: Smart Floor Construction, Hardware, and Wiring

The entire SmartCare apartment is equipped with a pressure sensitive smart floor with a resolution of about one pressure sensor per square foot. The choice for pressure sensors was the *Tekscan FlexiForce* pressure sensor [25]. The floor is built on disc-like sensors with rubber padding deployed in a 1-sqft square matrix configuration. On top of these sensors we have a click-together ceramic tile rigid flooring structure that can bend along the tile lines. Thus, each single tile is floating over four *Tekscan* sensors in the sensor-matrix. These tiles are rigid enough to handle people and object weights usually encountered in a home by being supported only by their corners. The floor is built in 4ft by 8ft sections with each having a custom designed acquisition board using a PIC24FJ64GA004 microcontroller recording the readings of 32 sensors at 25Hz. Data from the acquisition boards is relayed via *BeagleBone Black* computers (BBB) [26] deployed in the walls of the apartment, which in turn preprocess and relay the data to the central server using Ethernet. The central SmartCare server is responsible for ensuring that all smart floor acquisition boards and their controlling BBBs are running correctly with the appropriate servers.

4 Floor Sensor Calibration

4.1 Introduction

The goal of sensor calibration is to have all sensors generate a consistent output regardless of sensor differences, imperfections, tile coupling, and location. The calibration process is the same for all sensors, but each sensor will have unique parameters learned from the process. Calibration takes each sensor's raw value and translates it to a known unit of weight measurement (lb or kg). Having each sensor properly calibrated is crucial when a Convolutional Neural Network (CNN) is applied across the entire floor for contact point separation since the CNN expects to treat every subsection of the floor equally. A person's weight and weight distribution should be interpreted the same way no matter which part of the floor they are standing or walking on.

Traditional sensor calibration consists of taking direct measurements of the lone sensor under known load conditions (i.e. known kg weights) and using this data to establish the load curves for the specific sensor. However, this is difficult for this smart floor since the flooring material produces and distributes loads among multiple sensors and makes it thus impossible to produce specific known loads on individual sensors. Moreover, movement and shift in flooring materials as well as sensor characteristics can lead to changes in the base values of unloaded sensors and yield hysteresis effects where loading (or unloading) a sensor temporarily changes the baseline for that sensor. A second problem that makes calibration in a live-in environment difficult is shifting furniture and objects that can change the baseline when the floor is to be used for gait parameter estimation.

To address these difficulties, an automatic calibration approach is needed that can estimate sensor calibration values for the entire floor without the need for targeted single sensor calibration actions but rather from common use data. Additionally, this approach should continuously readjust zero load estimates to address sensor drift, hysteresis, and changes in passive (object) loads in the environment. For this, an approach is proposed here that uses a person's (or multiple persons') arbitrary walking data to estimate load and no-load conditions for sensors and subsequently continuously re-estimates zero-load conditions and sensor sensitivity parameters.

4.2 Lab Smart Floor

A smaller version of the apartment's smart floor was built in our ASSIST Lab (UTA – ERB 102) to locally run experiments and develop software. The lab floor was constructed using the same hardware and methodology consisting of 128 *Tekscan FlexiForce* pressure sensors covering a 16ft x 8ft area covered by 128 snap-in ceramic tiles (same as apartment). Each pressure sensor has adhesive rubber padding on both sides placed under each intersection of four tile corners (unless it is the bottom or right floor edge). The upper and left edges of the floor have sensorless rubber strips for tile stability that is of the same thickness as the rubber applied to each sensor.

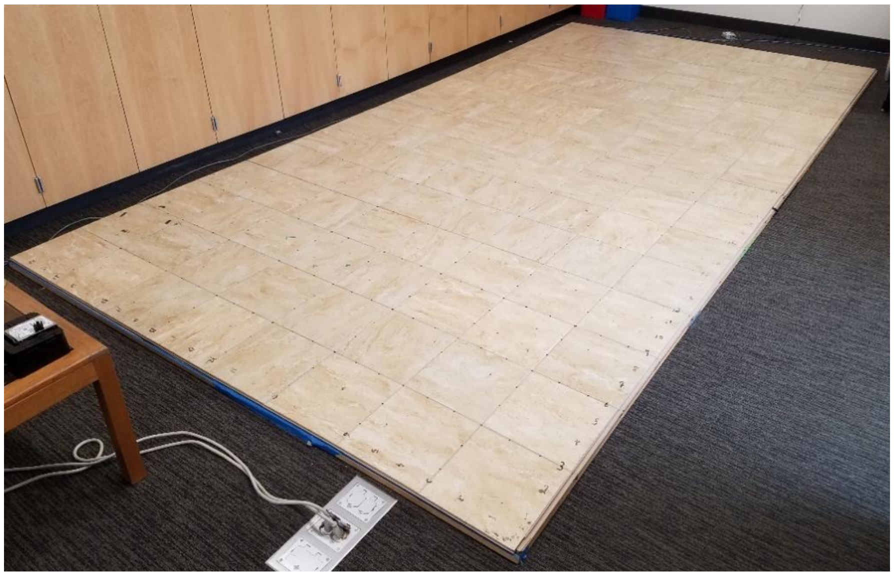


Figure 4.1: Lab Smart Floor for Experimentation

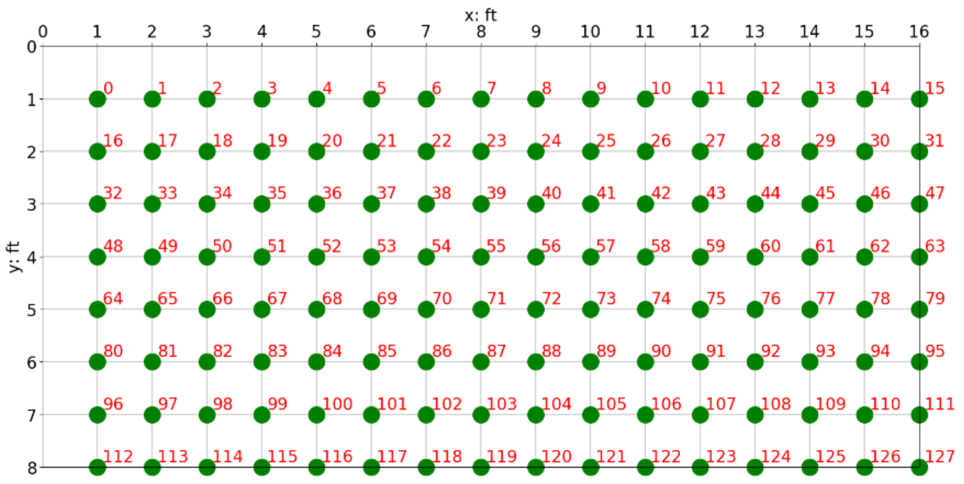


Figure 4.2: Lab Smart Floor Sensor Layout and Numbering Scheme

4.3 Calibration Notation, Model, and Sensor States

The following notation for key terms is used for the calibration process regarding the lab floor's specific layout and size:

• i	sensor index: 0…127
• t	time index
• s_i	raw uncalibrated ADC value for sensor i: 0…1023
• z_i	zero offset term for sensor i: non-negative, variable over time
• a_i	calibration coefficient for sensor i: non-negative, constant over time
• w_i	calibrated weight value for sensor i in kilograms

Using a top-left origin approach, the sensor indexing i for the 128 sensors of the lab floor begins at the top left corner and ends at the bottom right corner. The floor sensor's data is collected at 25Hz, therefore each time index t represents 1/25 of a second. The embedded circuity and microcontroller samples each pressure sensor, performs a 10-bit analog-to-digital conversion (ADC), and returns a whole number integer value ranging from 0 to 1023. This raw sensor value $s_i^{(t)}$ is what must be calibrated and translated to known weight units (kg). Through experimentation the pressure sensor exhibited linear behavior when increasing, known weights were applied, and outputted a value of 0 when no weight was applied (0 kg).

A traditional linear approach for converting/calibrating a value of unknown units to one of known units (i.e. kg) is to use the linear equation of defining a line:

$$y = mx + b$$

Specifically converting ADC values (*s*) to weight in kg (*w*) for the lab floor:

$$w = as + b$$

where the b offset could account for constant weight on the sensor (i.e. tiles or furniture) and the a scalar explains the linear relationship between w and s. Simple linear regression techniques such as ordinary least squares (OLS) could be used to calculate the *a* and *b* values with only a few samples of ADC values *s* and known weights *w*.

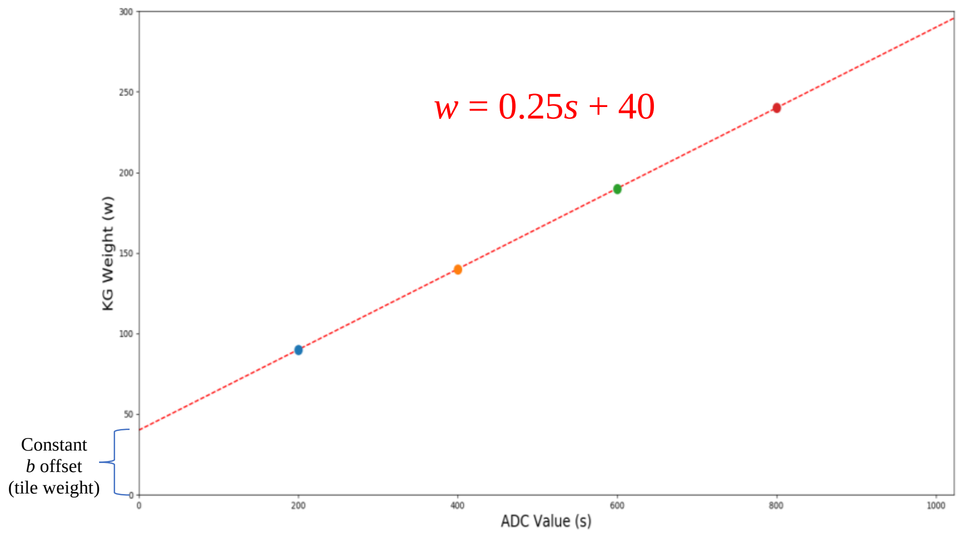


Figure 4.3: Theoretical Example of Translating Sensor Values *s* to Weight Values *w*

While this approach is feasible for calibrating standalone sensors independently it is too limited for an interconnected sensor grid underneath a shifting tile floor. The weight to be captured on top of the floor spreads across multiple tiles and sensors. Also, the tile coupling, shifting, and flexing over time in the x, y, or z directions can shift the constant tile weight offset *b* each sensor experiences. Constant recalibration would have to be applied to find the new updated *a* and *b* values to account for any day-to-day tile floor changes.

A more practical approach for this floor system would be to automatically remove the constant non-human *unloaded* weight seen by the sensor so only a single term *a* is required to be calculated. The term *unloaded* weight describes sources of weight we are not interested in knowing or capturing. This ever-changing weight could be tile, furniture, or temporary non-human objects. For example, if a 90 kg person's whole home is fitted with this floor tile sensor system, standing freely in the kitchen should yield the same weight calculation result as when they are sitting on a couch in the living room. The constant weight of the couch should be filtered out so only the person's weight is captured and converted to kg. To achieve this, a *z* offset term needs to be subtracted from the ADC sensor value *s* over time. This *z* value fluctuates to account for constant weight being added or removed on or near the sensor in question. Once the sensor's *s* value has had its *z* offset removed, linear regression in the form of OLS can be applied to learn each sensor's *a* value. The calibrated weight *w* experienced by the single sensor *i* at time *t* is expressed as:

$$w_i^{(t)} = a_i(s_i^{(t)} - z_i^{(t)})$$

Since no sensor is truly independent once it has been placed underneath the tile floor, the calibrated weight experienced by the entire lab floor at time *t* is expressed as:

$$\sum_{i=0}^{127} w_i^{(t)} = \sum_{i=0}^{127} a_i(s_i^{(t)} - z_i^{(t)}) = w_{floor}^{(t)}$$

Each sensor's *state* at any time *t* can be categorized as one of the following:

$$state_i^{(t)} \leftarrow \begin{cases} Loaded + & \text{if sensor value is significantly } \textbf{above} \text{ current } z \text{ offset} \\ Unloaded & \text{if sensor value is } \textbf{within } z \text{ offset threshold range} \\ Loaded - & \text{if sensor value is significantly } \textbf{below} \text{ current } z \text{ offset} \end{cases}$$

The *Unloaded* state can be interpreted as no person is contributing to the weight observed by the sensor, only static, non-human objects affect the sensor's value. *Loaded+* signifies a human weight has caused an increase in the sensor value *s* and should not be filtered out or lumped together with the *z* offset. *Loaded-* is a momentary state where the tile's coupling and the elasticity of the rubber surrounding the sensors causes the sensor to read a lower value than the current *z* offset level. These are seen as brief negative weights due to the impact forces of a person walking on the tile-covered smart floor. This state's importance is to more accurately calculate the *a* scalar during the linear regression process for each sensor during calibration and prevent under or over shooting of the final weight value *w*.

4.4 Calibration Process Overview

The smart floor sensor calibration process consists of three main steps:

1. Find each sensor's *unloaded* standard deviation σ_i used for removing zero offsets and determining *sensor states* during training and testing
2. Find each sensor's weight coefficient a_i that performs the linear transformation from undefined raw sensor values s_i to kg weight units
3. Use σ_i and a_i with new incoming floor data to calibrate and translate sensor values into w_i units

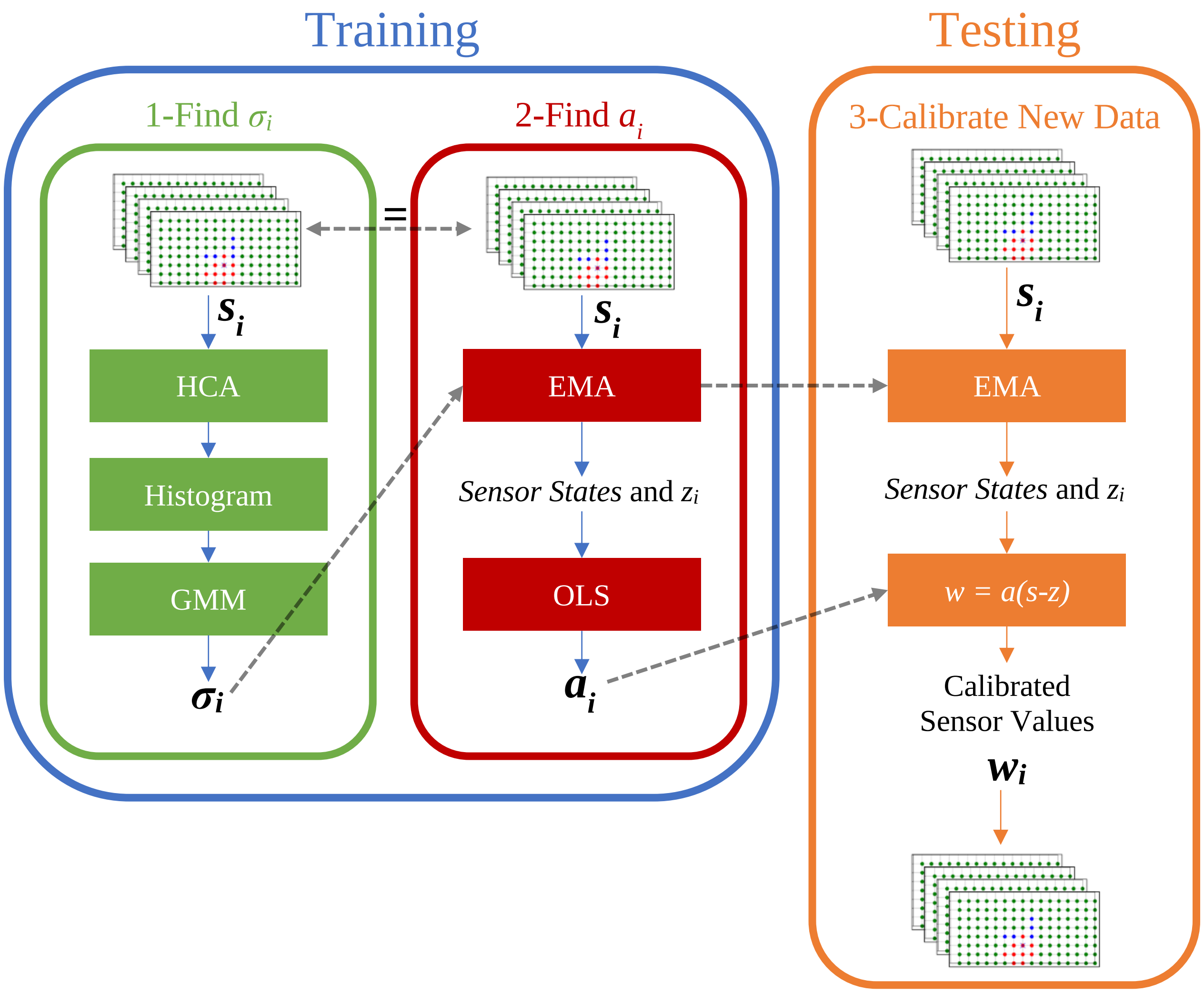


Figure 4.4: High-Level View of Calibration Process

4.5 Finding Each Sensor's Unloaded Standard Deviation

The first step is to discover how each sensor behaves under *unloaded* conditions. Specifically, how much does each sensor's value *s* fluctuate under normal conditions when the only present weight sources are the tile and non-human static objects on the floor. These fluctuations in the sensor value could be due to ADC noise, resolution, temperature, or nearby vibration. To describe this fluctuation range, each sensor's *unloaded* standard deviation σ_i is calculated. To find the σ_i for each sensor, a specific training dataset was recorded where a person of known weight walked across the entire lab floor and where each sensor experienced equal amounts of *loaded* and *unloaded* times.

4.5.1 Hierarchical Clustering Analysis

Hierarchical Clustering Analysis (HCA) was performed on all 128 sensors in order to split data into two groups: *Unloaded* or *Loaded*. The groupings are not the same sensor states described previously; those states will be calculated later. *Unloaded* refers to times when the sensor is under normal conditions and is not experiencing any human influence. *Loaded* refers to a sensor that is currently being influenced by human weight.

HCA in the form of Agglomerative Clustering was performed on each sensor's raw 1D s_i data with the parameters to find *two clusters* using *ward linkage*. Since the standard deviation of sensors is the desired result, *ward* was chosen as the linkage criterion because it minimizes the variance between data points when deciding whether they should be clustered. Of the two resulting clusters, the cluster with the smaller s_i mean was labeled *unloaded*. Any of the few data points incorrectly clustered, as evident in Figure 4.8, will be filtered out in the subsequent Gaussian Mixture Model step.

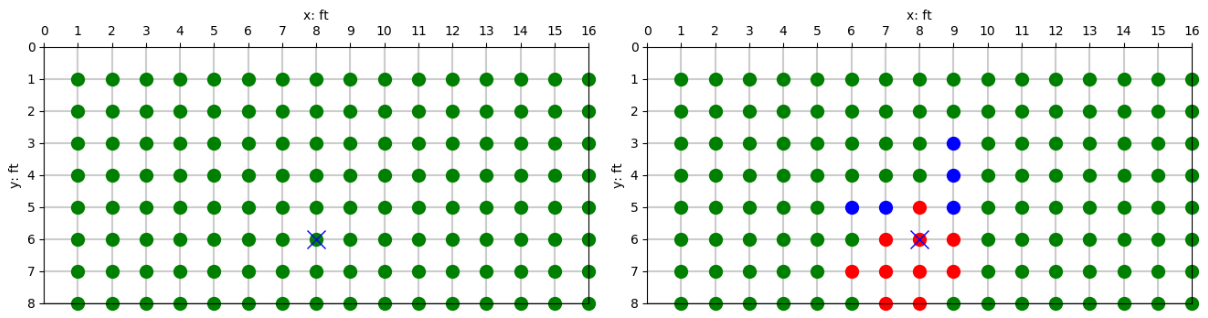


Figure 4.5: Sample Sensor (X) Under Unloaded Condition (left) and Loaded Condition (right)
Sensor Colors: Green = *Unloaded*, Red = *Loaded+*, Blue = *Loaded-*

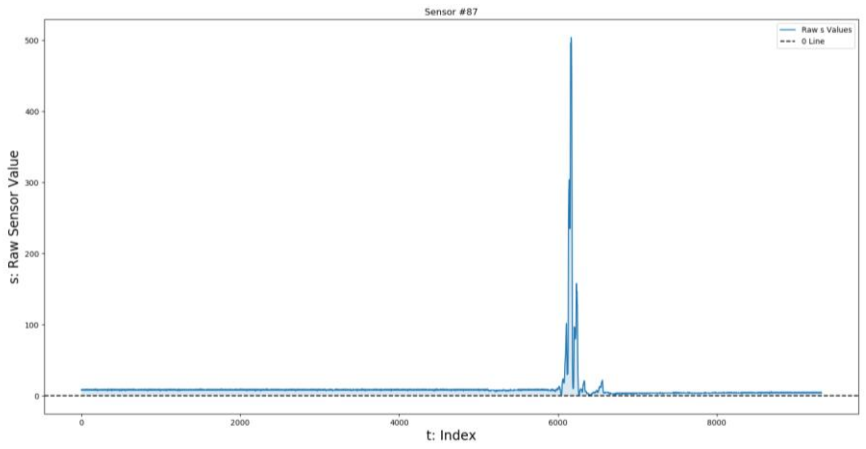


Figure 4.6: Sample Sensor Values Before Clustering

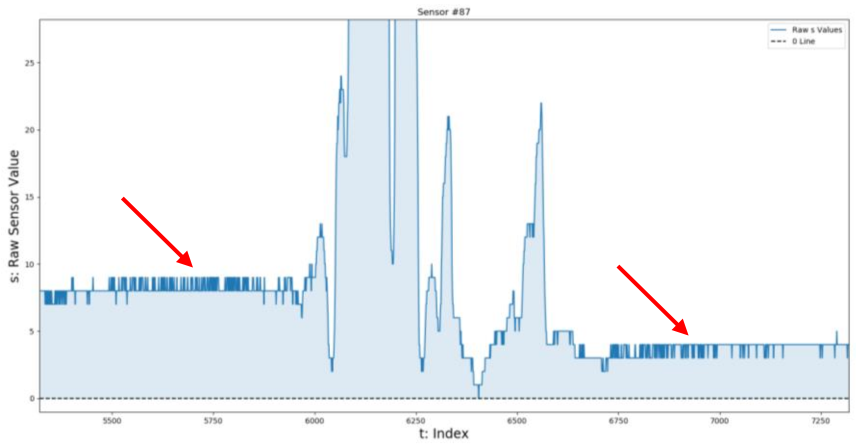


Figure 4.7: Zoomed Image to Show Zero-Line Shift After Human Presence

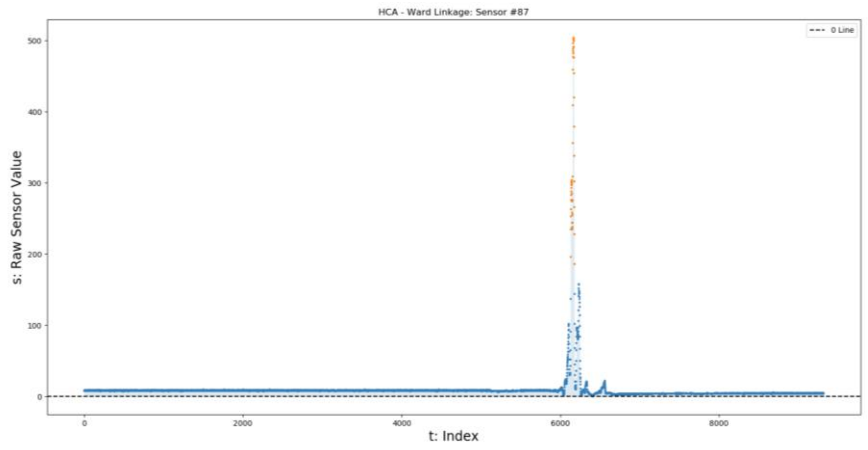


Figure 4.8: Sample Sensor Values After Clustering (cluster 1 - orange, cluster 2 - blue)

4.5.2 Gaussian Mixture Model

After experimentation and observation, an assumption was made that the *unloaded* sensor readings are normally distributed, and fluctuations resemble a Gaussian distribution. However, a single Gaussian cannot fully describe a sensor's activity when no human weight is present since the shifting tile and the addition of static weight might cause the zero-line to shift up or down. As shown in Figure 4.7, the zero-line shifted down after a human walked over the sensor area. The steady zero weight readings both before and after being *loaded* are valid and should be included in finding the *unloaded* standard deviation σ_i, hence the *unloaded* sensor times are seen as a mixture or combination of multiple Gaussian distributions when looking at the entire dataset.

To find the most common or frequent values each sensor outputs during *unloaded* times histogram data was generated for the smaller mean *unloaded* cluster. To make the integer discrete values of the histogram bin edges and values more continuous before performing a Gaussian Mixture Model (GMM) operation, "mini-gaussians" were calculated for each histogram bin to smooth out the data, essentially turning the stark histogram plot into a more continuous and smoother scatter plot.

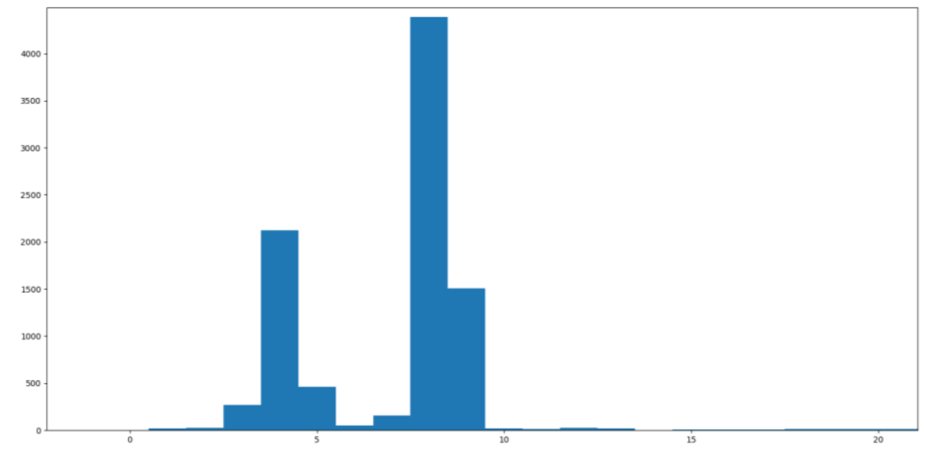


Figure 4.9: Histogram Data for Sample Sensor (Modes 4 and 8)

Using the perturbed histogram data, a GMM operation, using the Expectation-Maximization algorithm, was performed with the covariance type parameter set to *spherical* since each resulting Gaussian component's variance is deemed independent [27]. The GMM was run multiple times with different numbers of mixture components (1 through 5) to discover which number of Gaussians best explain the *unloaded* data. The GMM run with the best Akaike Information Criterion (AIC score) was deemed the best model [28].

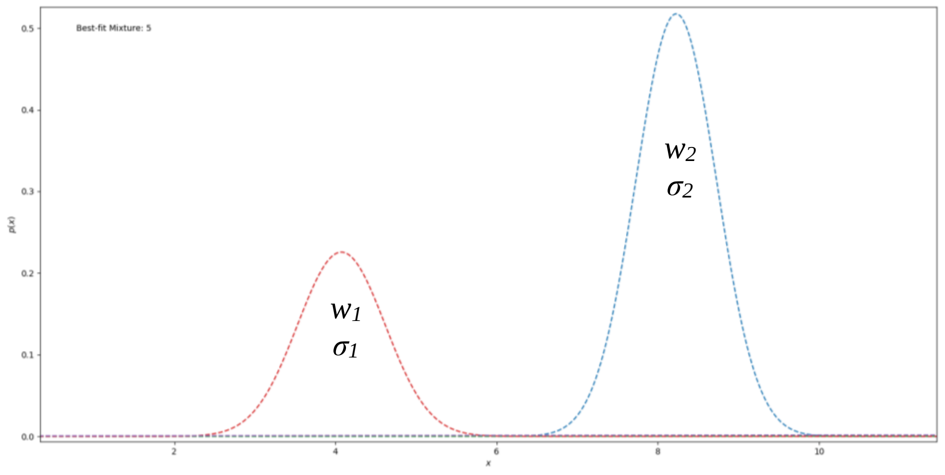


Figure 4.10: GMM Output for Sample Sensor (5 found, 2 kept)

Within the final GMM model any of the mixture models that contributed less than 10% to overall data were discarded. This removes outliers and any incorrectly labeled data points that occurred during HCA. The remaining mixture components' weights w_n were rescaled to sum to 1.0 and multiplied by their own standard deviations σ_n to compute the final *unloaded* standard deviation σ_i for every sensor.

$$\sigma_i = \sum w_n * \sigma_n$$

4.6 Determining Sensor State

Using the sensor calibration formula $w_i^{(t)} = a_i(s_i^{(t)} - z_i^{(t)})$, each sensor's zero offset term z_i needs to be continually updated to account for natural zero drift to ensure the proper amount of static zero weight is removed from the s_i term before applying the calibration coefficient a_i. The role of each sensor's *unloaded* standard deviation σ_i is to determine whether a potential new z_i value is likely a true *unloaded* moment or has deviated too far beyond normal zero ranges deeming it a *loaded* event. Each sensor's state over time $state_i^{(t)}$ must be determined using a moving average of sensor values to help smooth out any unforeseen short-term fluctuations or outliers.

4.6.1 Exponential Moving Average

Within the Exponential Moving Average algorithm (EMA) the first sensor reading is assumed *Unloaded* and is the initial starting point for the zero offset:

$$z_i^{(0)} \leftarrow s_i^{(0)}$$

Every subsequent sensor value is compared using its own σᵢ to determine the proper sensor *state*. If the current value is larger than 4 standard deviations σ_i above the most recent z_i, then this $s_i^{(t)}$value at time *t* is classified as *Loaded*+ and the new $z_i^{(t)}$ is just a copy of the previous $z_i^{(t-1)}$. If the current value is smaller than 4 standard deviations σ_i below the most recent z_i, then this $s_i^{(t)}$value at time *t* is classified as *Loaded*- and the new $z_i^{(t)}$ is also just a copy of the previous $z_i^{(t-1)}$. If the current value is less than 4 standard deviations σ_i from the most recent z_i in any direction, then this $s_i^{(t)}$ value at time *t* is classified as *Unloaded* and the new $z_i^{(t)}$ is updated using the standard EMA formula with $\alpha = 0.1$ (value found through experimentation).

$$state_i^{(t)} \leftarrow \begin{cases} Loaded + & if(\, s_i^{(t)} > z_i^{(t-1)} + 4 * \sigma_i) \\ Loaded - & elif(\, s_i^{(t)} < z_i^{(t-1)} - 4 * \sigma_i) \\ Unloaded & else \end{cases}$$

$$z_i^{(t)} \leftarrow \begin{cases} z_i^{(t-1)} & if\ state_i^{(t)} = Loaded + \\ z_i^{(t-1)} & if\ state_i^{(t)} = Loaded - \\ \alpha * s_i^{(t)} + (1-\alpha) * z_i^{(t-1)} & if\ state_i^{(t)} = Unloaded \end{cases}$$

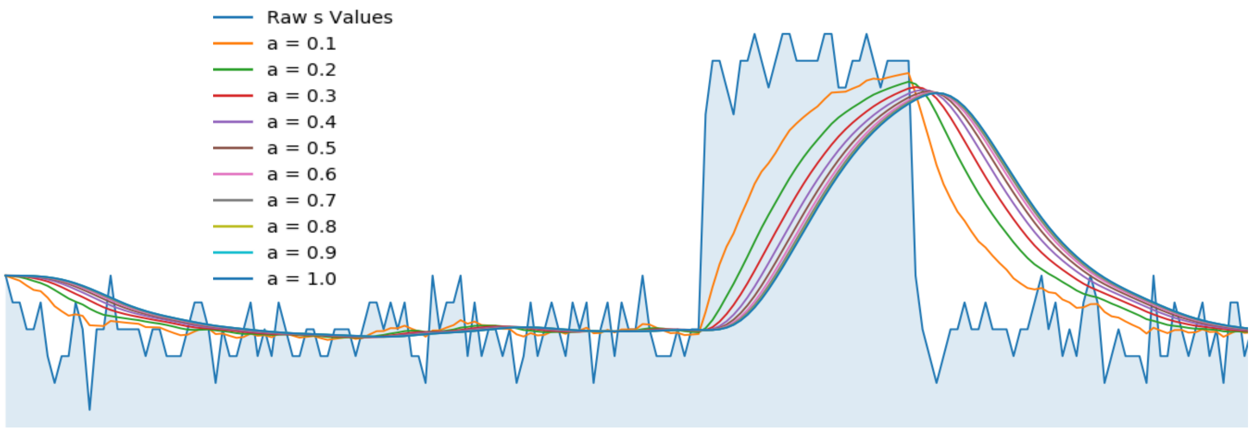


Figure 4.11: Different α Smoothing Factors on Sensor Data Assuming All Data Points are in the *Unloaded* State

There is a pitfall to solely using the EMA technique; a sensor can become "stuck" and have an incorrectly labeled state. The zero offset could naturally drift up or down over time and the EMA's smoothing effects might not catch it. After a person has walked on and left a sensor area, the EMA might forever label the sensor as *Loaded+/-* when there is truly zero human weight and it should be labeled as *Unloaded*. Below is an image of a stuck sensor condition, where the orange line is the EMA calculated zero offset over time $z_i^{(t)}$. The spikes were *Loaded+* times when a person walked near the sensor and the blue raw sensor values were significantly shifted up after the person left the area. The orange zero offset line never corrects and believes the sensor is *Loaded+* the remaining time when it should be classified as *Unloaded*.

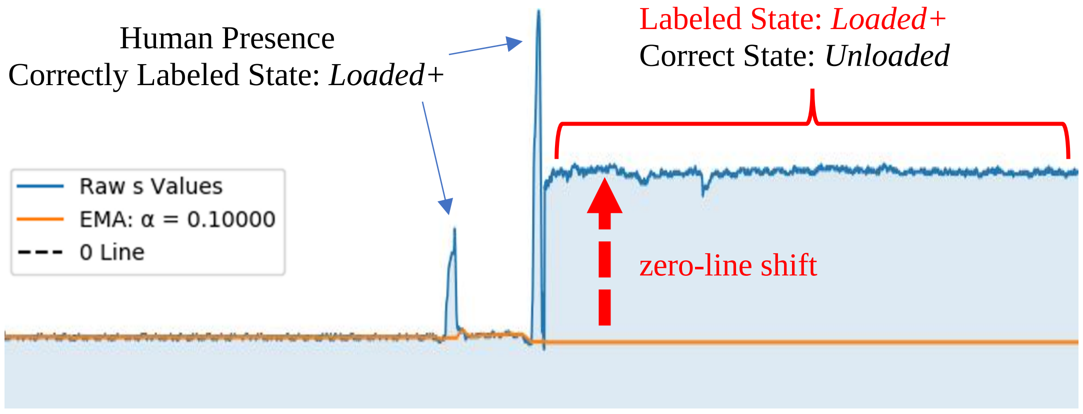


Figure 4.12: Stuck Sensor Condition

To overcome the possibility of the EMA missing a zero offset shift due to natural drift or tile coupling, a Simple Moving Average (SMA) operation is utilized within the larger calibration algorithm. After the current $z_i^{(t)}$ is calculated using the EMA, a moving window of 2 seconds (50 samples) is slid across the most recent raw sensor readings. If all 50 readings were labeled as *Loaded+* or *Loaded-* with no *Unloaded* gaps, then this buffer's standard deviation $\sigma_{sma\_i}^{(t)}$ is calculated. If it is within 100% of this sensor's zero weight standard deviation σ_i, its fluctuations could be explained completely by sensor noise and thus no evidence of weight shift on the sensor is detected. The algorithm uses this as an indication of the absence of a person and determines that a zero-line shift has occurred and should be fixed.

$$if\ \left(\left|\sigma_i - \sigma_{sma_i}^{(t)}\right| \leq \sigma_i\right) then:$$

$$z_i^{(t-1)} \leftarrow buffer\_mean_{sma_i}^{(t)}$$

Another problem arises when implementing this SMA fix. A person can stand still enough to cause the sliding window's standard deviation to be equal or even less than the sensor's σ_i. Figure 4.13 shows an example sensor reading profile of a person standing still being incorrectly labeled as *Unloaded* (zero-line did not shift and should not be redefined).

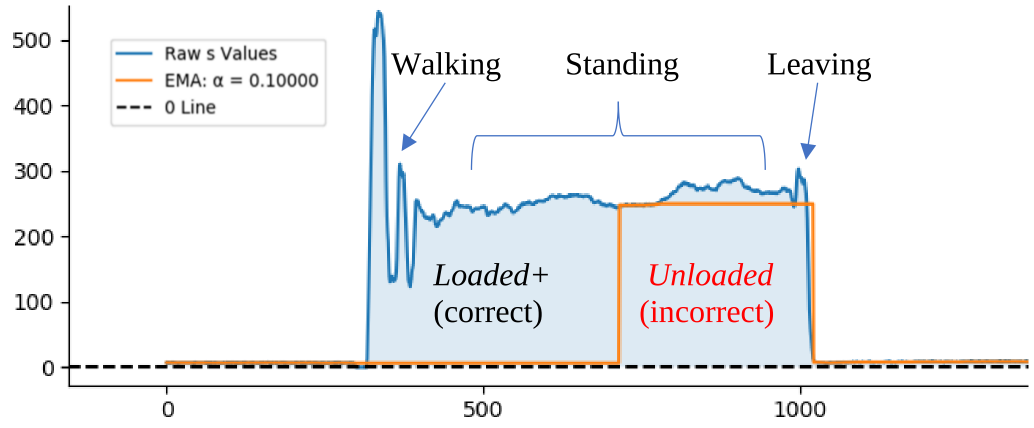


Figure 4.13: Incorrect State Labeling of a Still Person

To overcome the possibility of a person standing perfectly still and this algorithm labeling a *Loaded* time incorrectly as *Unloaded*, a threshold is applied within the SMA algorithm. The rationale for this threshold is the observation that a person standing perfectly still will generally need to be in a stable stance, applying significant weight on both feet, leading to high readings, while a sensor drift will usually be of a lower magnitude. This threshold is of the uncalibrated units of raw sensor values. After observing many cases of this error with people of different weights, a vertical window threshold of 10 was picked. Based off future a_i coefficient calculations, this raw sensor value s=10 amounts to 2.72 kg maximum.

If the difference between the sliding window's buffer mean and the current $z_i^{(t)}$ was greater than 10 then the SMA fix was not applied. If the difference is within the threshold, a fix is deemed necessary. Instead of correcting from this point t forward, the algorithm jumps back 50 time steps (size of sliding window) to adjust the zero mean at that point $z_i^{(t-50)}$ and runs the entire EMA/SMA algorithm again on those points so every sensor value within this SMA time window is labeled correctly:

$$if\left(\left|z_i^{(t)} - buffer\_mean_{sma_i}^{(t)}\right| \leq 10\right)\ then:$$

$$z_i^{(t-50)} \leftarrow buffer\_mean_{sma_i}^{(t)}$$

$$and\ restart\ EMA\ at\ t-50\ index$$

The resulting complete algorithm to determine the sensor load state and to dynamically adjust each sensor's zero load readings in shown in Algorithm 1.

Algorithm 1 Determine Sensor State Using EMA and SMA

```
input: set of raw sensor values $S$
       $unloaded$ standard deviation $\sigma$
output: set of zero offsets $Z$
        set of sensor states $States$

$\alpha \leftarrow 0.1$
$t \leftarrow 0$

while $t <$ length($S$) do
    $flag\_loaded \leftarrow$ false
    $s \leftarrow S[t]$

    if $t = 0$ then
        $\mu \leftarrow s$
        $States[t] \leftarrow$ unloaded
    else
        if $s > \mu + 4*\sigma$ then
            $flag\_loaded \leftarrow$ true
            $States[t] \leftarrow$ loaded+
        else if $s < u - 4*\sigma$ then
            $flag\_loaded \leftarrow$ true
            $States[t] \leftarrow$ loaded-
        else
            $\mu \leftarrow (\alpha*s) + (1 - \alpha)*\mu$

    $Z[t] \leftarrow \mu$

    if $flag\_loaded$ = true then
        if length($sma\_buffer$) = 50 then
            delete oldest element of $sma\_buffer$
            append $s$ to $sma\_buffer$
            $\sigma\_sma, \mu\_sma \leftarrow$ stdev($sma\_buffer$), mean($sma\_buffer$)

            if $|\sigma\_sma - \sigma| \leq \sigma$ then
                if $|\mu\_sma - \mu| \leq 10$ then
                    $\mu \leftarrow \mu\_sma$
                    clear $sma\_buffer$
                    $t \leftarrow t - 50 - 1$
        else
            append $s$ to $sma\_buffer$
    else
        clear $sma\_buffer$
    $t \leftarrow t + 1$

return $Z, states$
```

4.7 Linear Regression to Find Calibration Term

After each sensor's zero offset series is determined, simple Linear Regression is employed to learn the coefficient term a_i unique to each sensor. These values should not have to be relearned for long periods of time since they are inherent parameters of the sensor and thus not directly affected by the tile structure and static weights moved on top of the floor. To learn these parameters, we utilize the fact that while a person is walking on the floor, the total weight attributed to the person does not change. w_{floor} is a known person weight measured with two digital scales prior to the smart floor training walking sequence (in our experiments 91kg). Due to the weight of the person staying constant, the total floor weight measured from all sensors at each point in time in the calibration sequence can be written as the following linear equation:

$$\sum_{i=0}^{127} a_i(s_i^{(t)} - z_i^{(t)}) = w_{floor}^{(t)}$$

Ordinary Least Squares (OLS) is used to estimate the 128 a_i parameters for the lab floor with two alterations. If at time *t* a raw sensor value $s_i^{(t)}$ belongs to the *Unloaded* state (zero human weight), then it does not contribute to the large *P* matrix, a 0.0 is placed instead. Also, if an entire row in the large *P* matrix is only populated with zeros, the row is omitted. These alterations ensure that only *Loaded* sensors contribute to the calculation of the a_i parameters and efficiency is increased. The matrix notation below uses *k* to indicate the length of the walking sequence; in practice this value will be smaller than t_{max} *- 1* after the alteration removes unnecessary rows, i.e. *t* time steps with no person present, in the large *P* matrix. Below, an example known training weight of 91 kg is used in place of $w_{floor}^{(t)}$.

$$P\vec{a} = \vec{w}$$

$$\begin{pmatrix} (s_0^{(0)} - z_0^{(0)}) & \cdots & (s_{127}^{(0)} - z_{127}^{(0)}) \\ \cdots & \cdots & \cdots \\ (s_0^{(k-1)} - z_0^{(k-1)}) & \cdots & (s_{127}^{(k-1)} - z_{127}^{(k-1)}) \end{pmatrix} \begin{pmatrix} a_0 \\ . \\ . \\ . \\ . \\ . \\ . \\ a_{127} \end{pmatrix} = \begin{pmatrix} w_{floor}^{(0)} \\ . \\ . \\ . \\ . \\ . \\ . \\ w_{floor}^{(k-1)} \end{pmatrix} = \begin{pmatrix} 91.0 \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ 91.0 \end{pmatrix}$$

After all the 128 a_i and σ_i parameters are calculated, the training part of the calibration process is complete. The calibration coefficient and zero weight standard deviation values are stored in a separate .json file so future walking data does not need to perform these operations again. All future testing data will import these variables for their respective sensors and undergo the zero offset EMA/SMA process to classify each sensor's state for all times *t*.

If a sensor at time *t* is labeled *Loaded+* or *Loaded-*, its current zero offset $z_i^{(t)}$is subtracted from its raw $s_i^{(t)}$ value and then multiplied by its a_i coefficient to transform the raw sensor value into a known weight measurement $w_i^{(t)}$ in kg. If a sensor at time *t* is labeled *Unloaded*, $w_i^{(t)}$ is assigned a value of 0.0 kg human weight.

5 Contact Point Extraction

5.1 Overview of Model

Due to the low 1-sqft resolution of the smart floor, single contact point extraction of a person's footfalls cannot be achieved as straightforwardly as when using a sensor-dense high-resolution floor or mat. To achieve this extraction while considering the non-linear coupling properties of the floor, a Convolutional Autoencoder Neural Network model is used. The model's sequential design consists of 3 main steps:

1. Design and train the Decoder model which maps known location data and weight to a 4x4 square grid of calibrated sensor values in a supervised fashion. This can be thought of as the Tile Model and is trained with a very specific training dataset.
2. Design and train the Convolutional Autoencoder (CAE) model that incorporates 128 copies of the pre-trained Decoder (transfer learning). The convolutional parts of the model learn the non-linear relationships and interactions between sensors, tiles, and how weight spreads on the smart floor. The CAE is trained in a semi-supervised fashion and can be trained with a large amount of arbitrary calibrated walking data.
3. After the CAE is fully trained the frontend Encoder portion is extracted and becomes the resulting final model used to translate regular calibrated walking data into a series of individual contact points no longer limited by the low resolution of the smart floor.

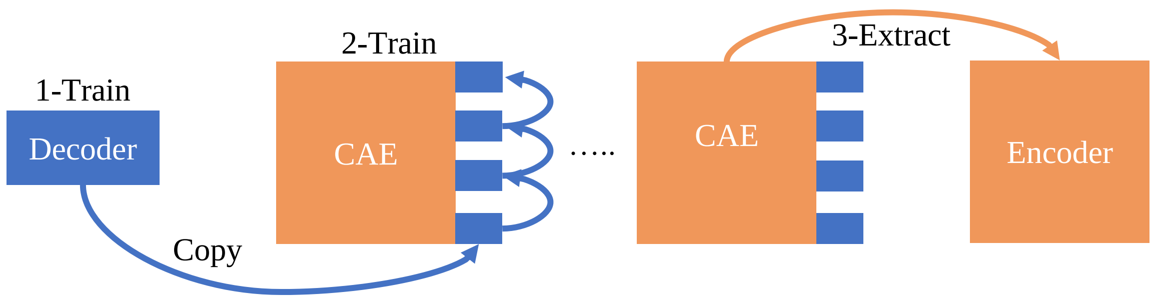


Figure 5.1: High-Level View of Model Iterations

The rationale behind this approach to separately train the Decoder and Encoder components addresses the requirement that we want to obtain contact location and weight data and thus the hidden representation of the autoencoder architecture has to be of a specific type which can only be achieved using some level of supervised data [29]. As opposed to supervised training of the Encoder which performs the ultimate contact point extraction task, however, the only supervised training performed here is of the Decoder, which requires significantly less and easier to obtain data. This stems from the observation that sensor pressure values from multiple weights on the floor combine purely additively and thus the sensor reading from multiple weights (contact points) on the floor can be computed by adding the values for each sensor for the Decoders sharing this sensor within their sensor region. This implies that it is possible to train the Decoder using only single contact point calibrated data which is significantly simpler to obtain than multi-contact data.

The Encoder, on the other hand, has the task to split sensor readings into sets of contact points, and thus requires multi-contact data. In the approach used here, the pre-training of the Decoder with single contact point data makes it possible to train the Encoder using unlabeled walking data which does not contain weight and contact point information and is thus significantly simpler to obtain. In addition, this procedure enables one to train the Decoder on a smaller part of the floor while training the encoder to capture the entire floor. Figure 5.1 shows the basic training sequence proposed here.

5.2 Training Data

Single contact point training data was recorded with the help of Sami Arshad under the supervision of Dr. Záruba and Dr. Huber. 1104 different location and weights were recorded on the lab smart floor, as shown in Figure 5.2, using a 1-inch diameter standoff where a person of known weight could steadily stand on to generate a single contact point pressure point. Locations were chosen every 4 inches in both x y directions for the left half of the floor covering tile centers, edges, and corners. For each data sample the absolute coordinates x_a and y_a were recorded along with the person's weight w_p. This training set is used to train the Decoder model directly and to aid in the CAE training as an auxiliary training target.

The main training dataset for the CAE requires no specific preprocessing or label creation since the Autoencoder nature of the CAE simply learns how to reconstruct full smart floor walking files (walking sequences). The only requirement is having the input and output walking data be fully calibrated as described in Chapter 4.

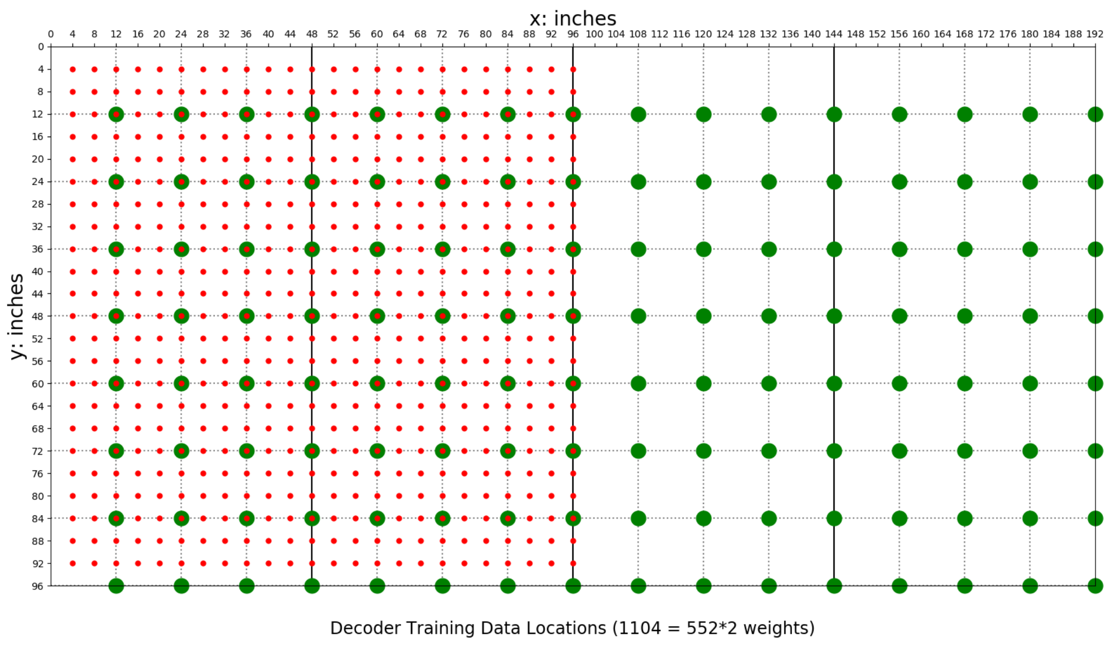


Figure 5.2: Single Contact Point Training Data Locations in Red

5.3 Decoder

The Decoder model is a standard feedforward neural network created and trained using Keras and Tensorflow [30] [31]. The architecture consists of 3 input nodes, 64 hidden nodes using a *sigmoid* activation function, and 16 output nodes using a *linear* activation function resulting in a total of 1,296 trainable weight and bias parameters. The standard mean squared error function was used as the training loss function with the Adam optimizer.

The 3 inputs are the relative x_p and y_p coordinates of the single contact point's location on a tile and the known weight w_p of the person generating the contact point's pressure. The 16 outputs represent the 4x4 calibrated sensor values w_1 ... w_{16} of the surrounding sensors of the tile in question, shown in Figure 5.3. The size of this surrounding area was picked based off observations which indicated that weights applied on a tile did not spread any further than this neighborhood.

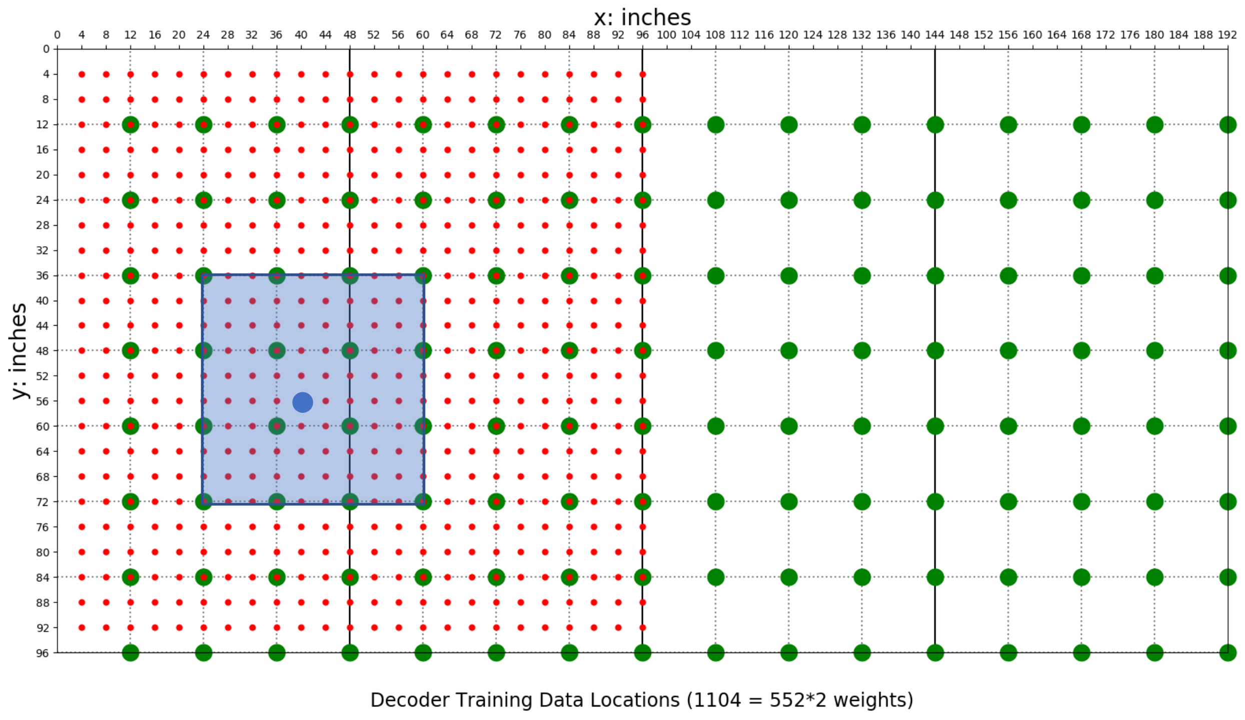


Figure 5.3: Decoder Training Data - 4x4 Sensor Area for Single Contact Point (blue circle)

The single contact point's absolute x_a and y_a coordinates (entire floor) are converted to the relative x_p and y_p coordinates (single tile) according to the lab smart floor's dimensions and layout. The range of the x_p and y_p values is [0…1] and w_p is also min-max normalized to the range [0…1].

When the tile ownership of a point comes into question as a contact point is on the edge of multiple tiles, multiple samples are created to give all tiles involved a fair, unbiased data sample during the Decoder's training process. If a contact point lies on the edge of two tiles two sets of relative points are created. Also, if a contact point lies on a corner four sets are created.

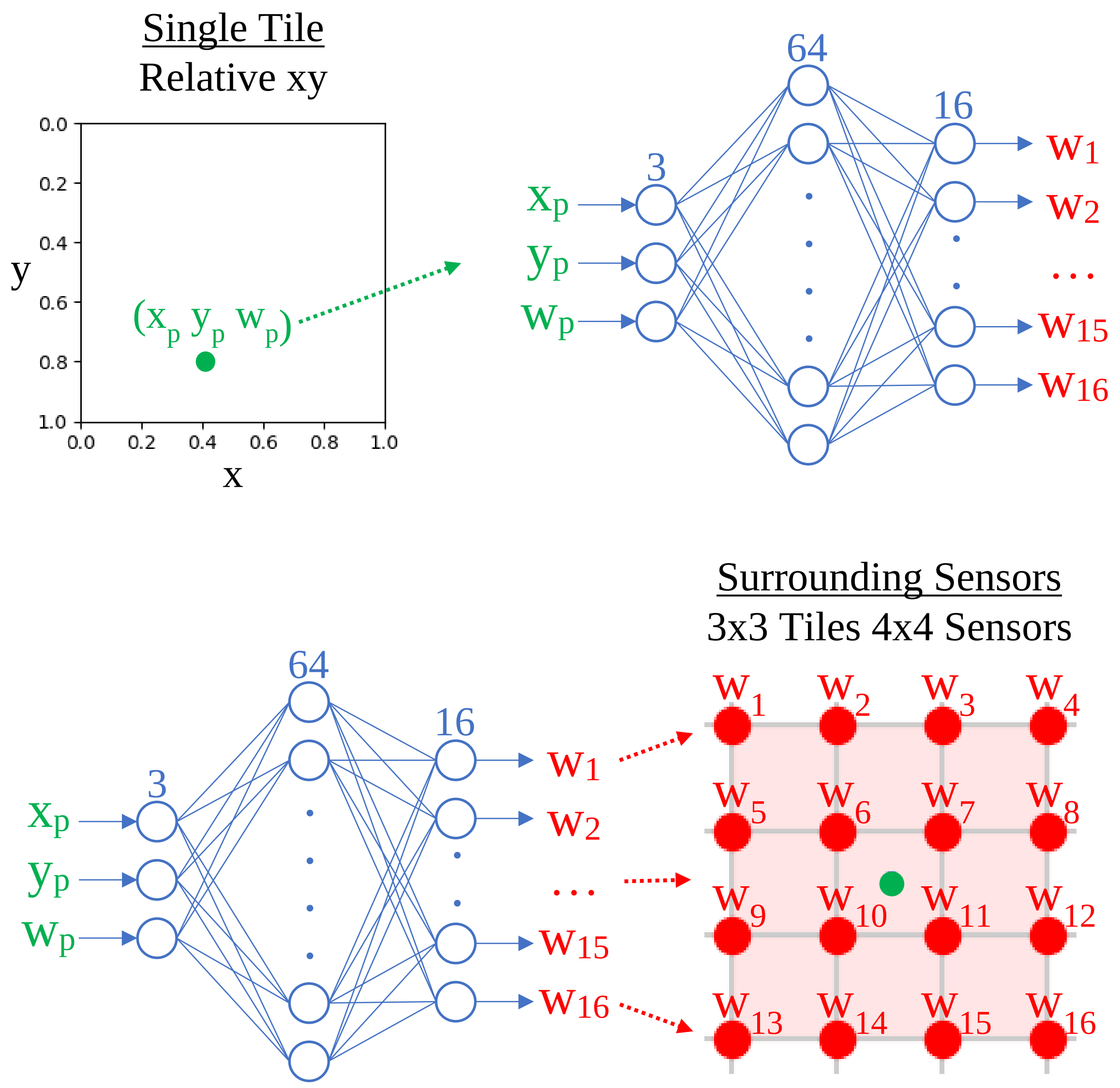
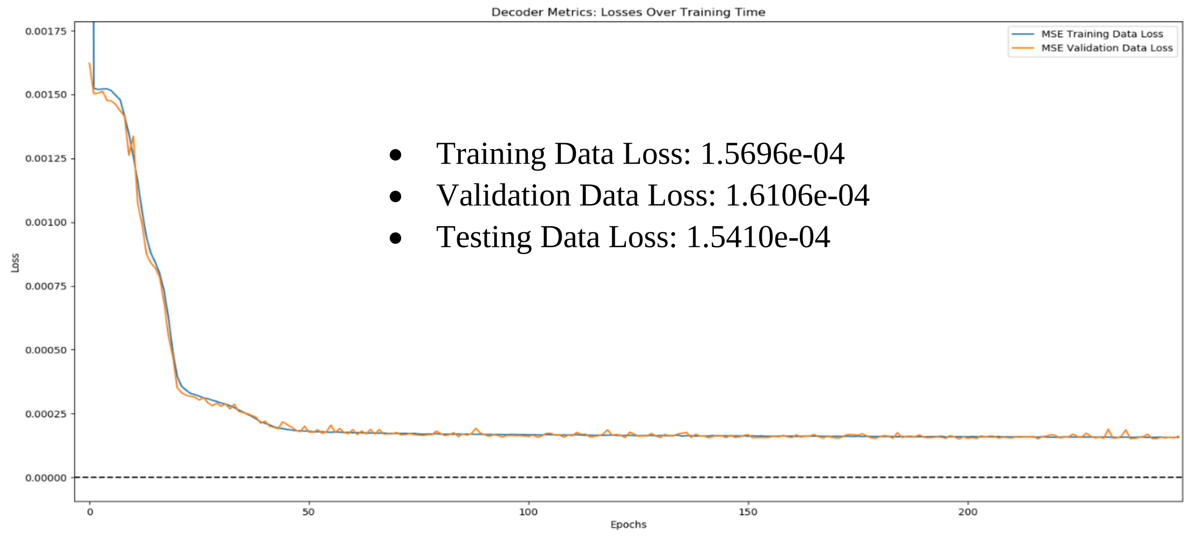


Figure 5.4: Decoder Model Input and Output

Figure 5.5 shows the learning curve for the training of the Decoder which resulted in the following losses for the fully trained Decoder, illustrating the successful performance and indication that the chosen architecture can accurately represent the sensor activations resulting from the calibration weights without overfitting.



- Training Data Loss: 1.5696e-04
- Validation Data Loss: 1.6106e-04
- Testing Data Loss: 1.5410e-04

Figure 5.5: Decoder Training Losses Over Time

5.4 Convolutional Autoencoder

Once the Decoder is trained, it can be integrated into a Convolutional Autoencoder architecture (CAE) while effectively serving as a distal teacher for the encoder portion [32]. The CAE consists of two main parts: the convolutional Encoder portion and the 128 pretrained Decoders, one for each tile. The overlapping Decoders' outputs for each sensor are added, effectively having the Decoders form a deconvolution transforming multiple contact points to corresponding sensor readings. The weight and bias parameters for the 128 identical Decoder models are frozen and untrainable during the CAE training process. The CAE's input data and output target data are entire floor snapshots of calibrated sensor data of various walking data from different people. Since the CAE's main goal is to reconstruct the input data as perfectly as possible, the bottleneck between the two portions forces the CAE, during training, to learn an encoding that adopts the contact point properties embedded within the Decoder.

The CAE slides or convolves 4x4 sensor (3x3 tile) kernels over the entire floor, as illustrated in Figure 5.6, and learns how neighboring sections of the floor interact, couple, and affect one another while also trying to recreate the same input pressure profile of the entire floor on its output in adherence to the contact point profile learned in the Decoder's training process. The stacking with other convolutional layers with differing kernel sizes allows the CAE to see larger segments of the floor and discover more complex non-linear relationships within the smart floor. In other words, the CAE takes a single entire floor snapshot and the Encoder portion outputs an (x_p y_p w_p) for every tile with the w_p serving as a probability of sorts. The complete CAE architecture used here uses 3 convolutional layers and additional padding masks to account for the sensor-less edges of the floor and is shown in Figure 5.7.

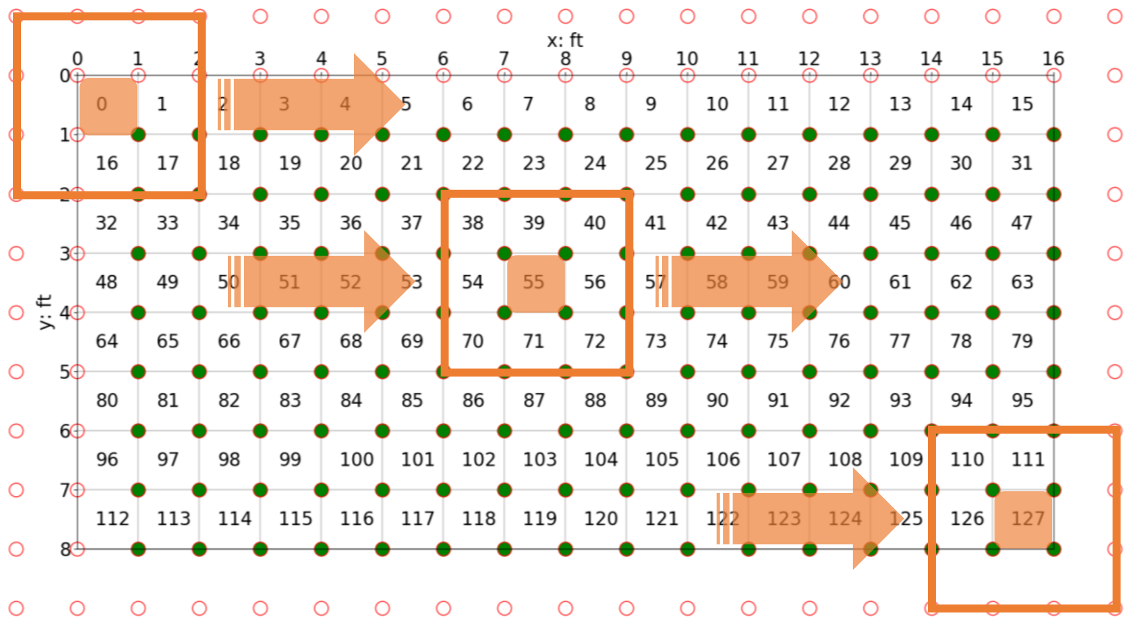


Figure 5.6: 4x4 Sensor Kernel Convolving Over the Entire Floor (hollow red circles are padding)

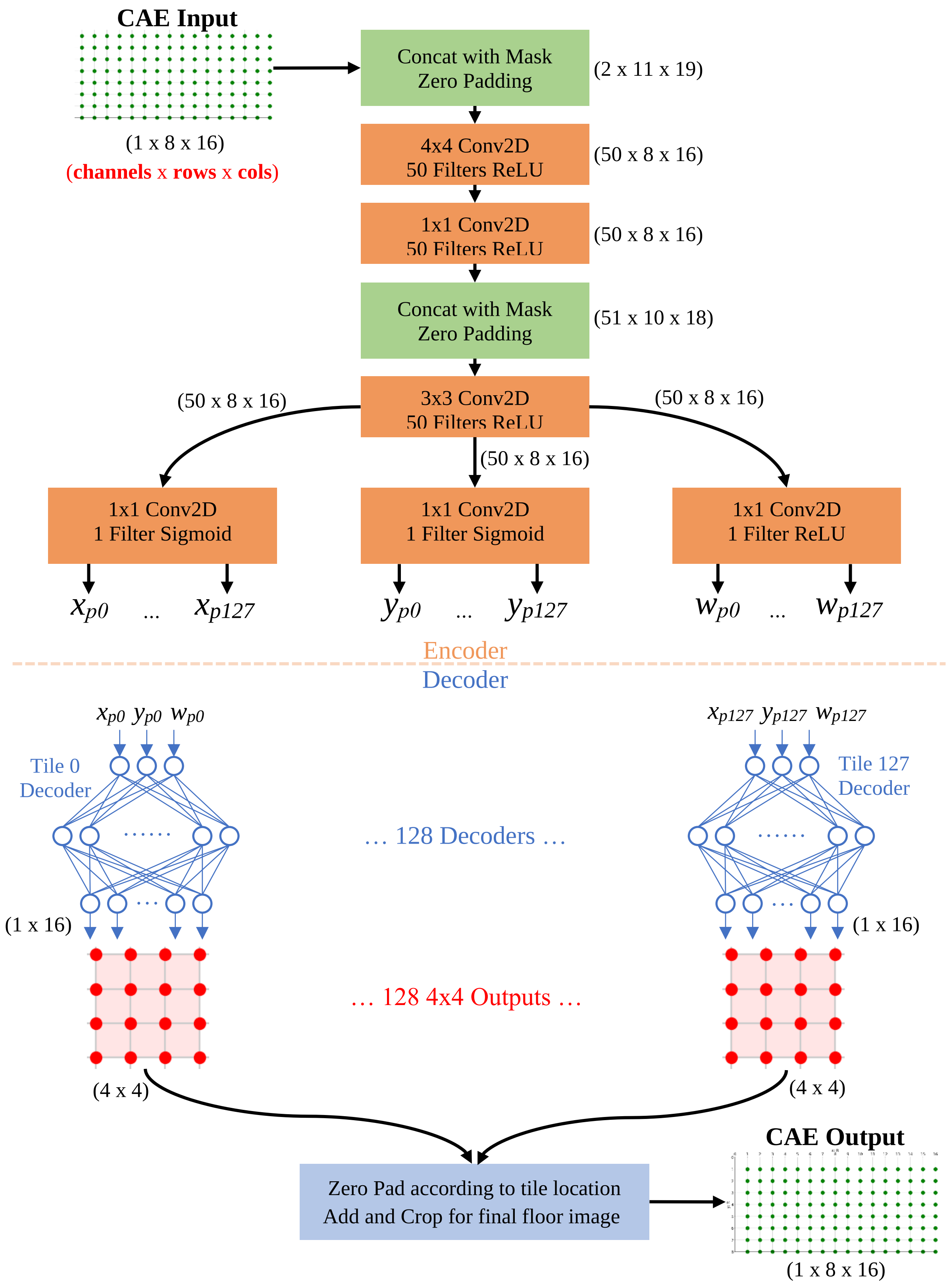


Figure 5.7: Detailed Architecture of the CAE

The CAE architecture layers are as follows:

1. Single-channel input layer of 2D dimension 8x16
2. Concatenation with Binary Mask Channel and Zero-padding
3. Convolutional Layer: 4x4 kernel, 50 filters, ReLU activation
4. Convolutional Layer: 1x1 kernel, 50 filters, ReLU activation
5. Concatenation with Binary Mask Channel and Zero-padding
6. Convolutional Layer: 3x3 kernel, 50 filters, ReLU activation
7. Branch x_p:
    a. Convolutional Layer: 1x1 kernel, 1 filter, sigmoid activation
8. Branch y_p:
    a. Convolutional Layer: 1x1 kernel, 1 filter, sigmoid activation
9. Branch w_p:
    a. Convolutional Layer: 1x1 kernel, 1 filter, ReLU activation
10. Concatenation of the 3 branches
11. 3 Element Slice in the 3D axis for each of the 128 tiles
    a. Auxiliary output layer for x_p y_p w_p values
12. Flatten (end of Encoder portion)
13. 128 copies of the pre-trained Decoder model
14. Zero-padding and Add Layer
15. Single-channel output layer of 2D dimension 8x16 (same as input)

* all convolutional layers used a stride of (1,1) with valid padding

*Total Weight and Bias Parameters: 193,241
*Trainable Parameters: 27,353
*Non-trainable Parameters: 165,888 (128 frozen Decoder models)

The input training data and the output targets are the same since the role of an autoencoder is to reconstruct data input and learn encodings. Each data sample was an 8x16 full floor snapshot of 128 calibrated smart floor sensors taken from various files of different people walking over the entire floor. All calibrated sensor values for the input and output were normalized to range [0…1] but using the same min-max scaling factors established during Decoder training. The output reconstruction loss function for the CAE was the standard mean squared error function with a special condition to prevent zero weight biasing due to most sensors reading being ~0kg the majority of the time. During training, if both the true and predicted values of a calibrated floor sensor are within +/- 1kg around zero weight then its contribution to the mean squared error function is reduced to 0.01%. The rationale here is that the CAE has already correctly converged on predicting 0kg for that sensor and should amplify the error in predicting other sensors that actually have significant weight.

Activity regularization was applied to the output of the w_p branch convolutional layer to encourage sparsity amongst the generated w_p values in the aim of having fewer tiles explain the pressure profile and data reconstruction (encourage single contact points over multiple points spread across multiple tiles). The reason for this is that due to the linear combination of sensors, a contact point on or close to a tile boundary can be represented either as a single contact point or as the result of multiple contact points that are on the edges of adjacent tiles with minimal differences in the sensor readings. In these situations, we would prefer the system to provide us with a single contact point to explain the pressure readings. A regularization loss function utilizing the l_0-norm would have been ideal since it solely penalizes a higher count of non-zero values and therefore minimizes the number of final contact points, but implementing a true l_0-norm within the Tensorflow framework proved too challenging. Instead the regularization loss function was the mean of the $l_{0.5}$-norm with a λ regularization factor of 3x10^{-5}. The regularization factor was found through experimentation to equally balance its contribution in comparison to the other losses.

$$\lambda * \frac{1}{128}\left(\sum_{i=0}^{127}\left|w_{p\,(i)}\right|^{0.5}\right)^{2}$$

To counteract the undesirable effect of the sparsity regularization driving all w_p values to 0.0, the CAE model is reinforced with an auxiliary training target making the overall model single input multiple output (SIMO). The large CAE training dataset of various walking files was interlaced with the original single contact point data used in Decoder training. The output labels x_p, y_p, and w_p and their predicted values contributed to the overall loss with their own standard mean squared error loss function with a weight factor of 0.2. The Encoder MSE loss only contributes 20%, found through experimentation, to balance its contribution to the overall loss function without overwhelming the CAE's MSE loss. With w_p being normalized in the same fashion as before.

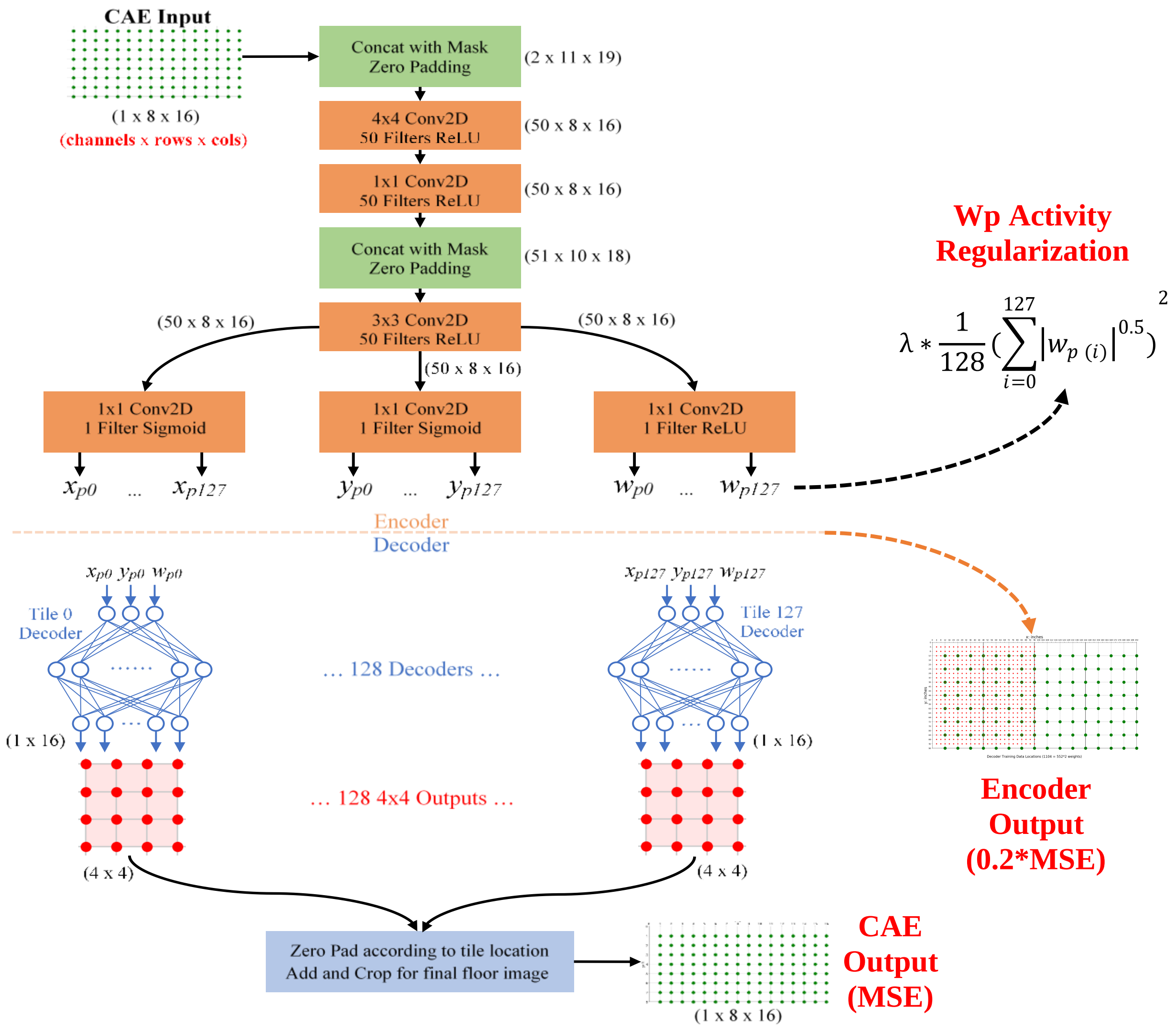

CAE Input
(1 x 8 x 16)
(channels x rows x cols)
Concat with Mask Zero Padding
(2 x 11 x 19)
4x4 Conv2D 50 Filters ReLU
(50 x 8 x 16)
1x1 Conv2D 50 Filters ReLU
(50 x 8 x 16)
Concat with Mask Zero Padding
(51 x 10 x 18)
3x3 Conv2D 50 Filters ReLU
(50 x 8 x 16)
(50 x 8 x 16)
(50 x 8 x 16)
1x1 Conv2D 1 Filter Sigmoid
1x1 Conv2D 1 Filter Sigmoid
1x1 Conv2D 1 Filter ReLU
x_{p0} ... x_{p127}
y_{p0} ... y_{p127}
w_{p0} ... w_{p127}
Wp Activity Regularization
$\lambda * \frac{1}{128}\left(\sum_{i=0}^{127} \left|w_{p\,(i)}\right|^{0.5}\right)^2$
Encoder
Decoder
x_{p0} y_{p0} w_{p0}
Tile 0 Decoder
x_{p127} y_{p127} w_{p127}
Tile 127 Decoder
… 128 Decoders …
(1 x 16)
(1 x 16)
… 128 4x4 Outputs …
(4 x 4)
(4 x 4)
Encoder Output (0.2*MSE)
Zero Pad according to tile location Add and Crop for final floor image
(1 x 8 x 16)
CAE Output (MSE)

Figure 5.8: SIMO CAE Architecture with Regularization and Multiple Output Targets

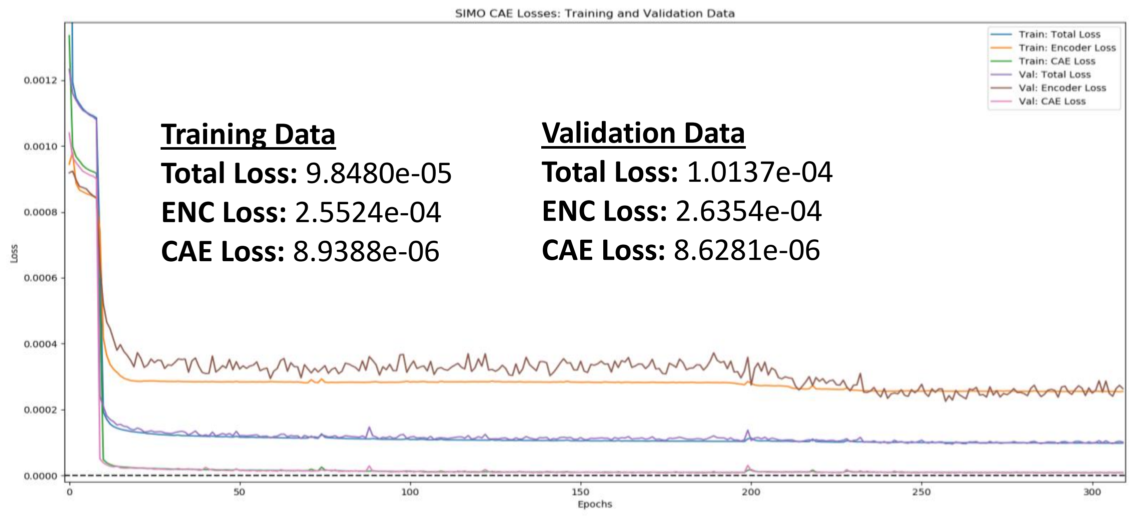
The SIMO CAE’s complete loss function is expressed as:

$$Total\ Loss\ =\ CAE\ Loss\ +\ Wp\ Regularization\ +\ Encoder\ Loss$$

- $CAE\ Loss\ =\ MSE(floor\_true, floor\_prediction)$
- $Wp\ Regularization\ =\ (3 * 10^{-5}) * \frac{1}{128}\left(\sum_{i=0}^{127} \left|w_{p\ (i)}\right|^{0.5}\right)^{2}$
- $Encoder\ Loss\ =\ 0.2 * MSE(xyw\_true, xyw\_prediction)$

To prevent zero-weight biasing, the *CAE Loss* single sensor contribution is reduced to 0.01% within the mean squared error function if both true and prediction weight targets are within +/- 1kg of zero weight. The *CAE Loss* indicates how well the model can reconstruct the entire floor weight. The *Wp Regularization* acts as a penalty to encourage more single contact point explanations. The *Encoder Loss* signifies how well the model’s bottleneck can generate accurate contact points per tile. Finally, the *Total Loss* describes how well the overall model is learning.

The behavior of the loss and regularization functions is shown in Figure 5.9 and resulted in the following losses, illustrating the CAE’s ability to learn a good representation without overfitting the training data.



Training Data		**Validation Data**	
Total Loss:	9.8480e-05	**Total Loss:**	1.0137e-04
ENC Loss:	2.5524e-04	**ENC Loss:**	2.6354e-04
CAE Loss:	8.9388e-06	**CAE Loss:**	8.6281e-06

Figure 5.9: SIMO CAE Training Losses Over Time

In the loss function figure, we can see that all loss functions decrease rapidly initially and then slowly converge to a minimum over the course of 300 epochs. It is important to note that during training a tradeoff between the autoencoder loss and the regularization loss functions is made, which results in the Encoder preferring to underestimate weights slightly which we correct for when estimating a person's weight.

5.5 Encoder

After the CAE is fully trained, the Encoder portion is kept as the final standalone model in predicting individual contact points per tile and across the entire floor. When a series of entire floor snapshots are given to the input of the Encoder, each output is of the shape (1 x 384) which holds the (x_p y_p w_p) values for each of the 128 tiles on the floor. If a tile's w_p value is close to 0kg then that tile's (x_p y_p w_p) is discarded and only tiles with weight activity are kept. Each final contact point is prepended with timestamp information given from the original series of input data resulting in (t x_p y_p w_p) to describe each contact point. The final Encoder model is shown below in Figure 5.10.

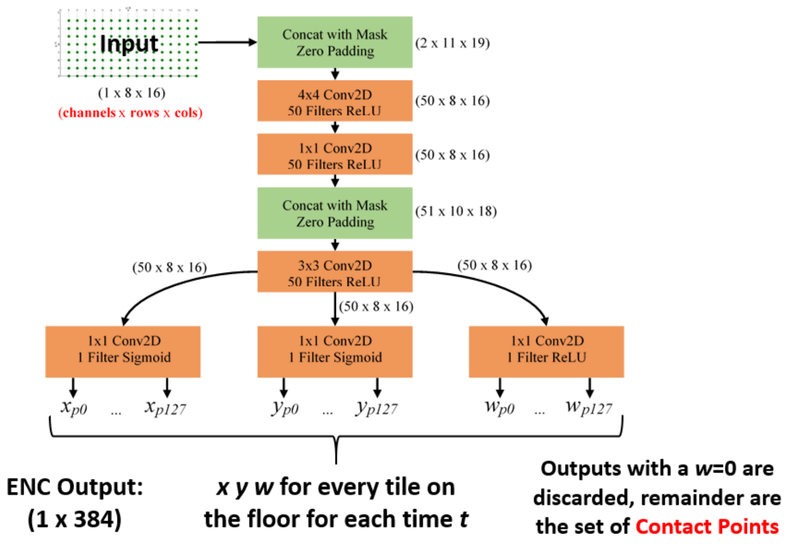
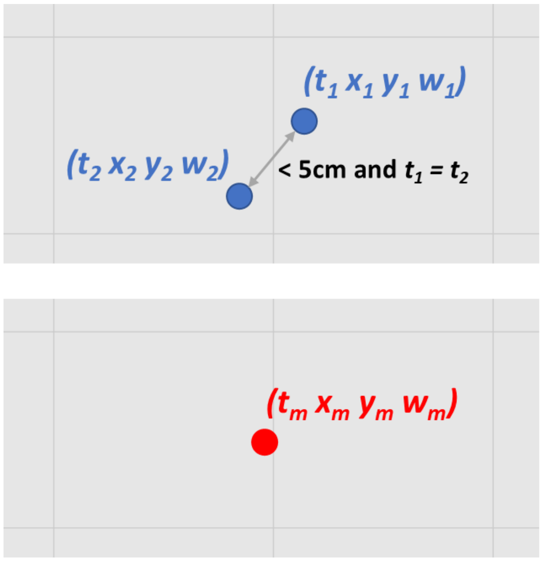

Input
(1 x 8 x 16)
(channels x rows x cols)
Concat with Mask Zero Padding
(2 x 11 x 19)
4x4 Conv2D 50 Filters ReLU
(50 x 8 x 16)
1x1 Conv2D 50 Filters ReLU
(50 x 8 x 16)
Concat with Mask Zero Padding
(51 x 10 x 18)
3x3 Conv2D 50 Filters ReLU
(50 x 8 x 16)
(50 x 8 x 16)
(50 x 8 x 16)
1x1 Conv2D 1 Filter Sigmoid
1x1 Conv2D 1 Filter Sigmoid
1x1 Conv2D 1 Filter ReLU
x_{p0} ... x_{p127}
y_{p0} ... y_{p127}
w_{p0} ... w_{p127}
ENC Output: (1 x 384)
x y w for every tile on the floor for each time t
Outputs with a w=0 are discarded, remainder are the set of Contact Points

Figure 5.10: Detailed Architecture of the Extracted Encoder Model

5.5.1 Converting Encoder Output to Contact Points

Once only relevant tile contact points remain, they must be converted from relative coordinates to absolute floor coordinate units according to their tile's location while retaining their original timestamps t and weight values w_p, $(t\ x_p\ y_p\ w_p) \longrightarrow (t\ x_a\ y_a\ w_a)$. If there exist multiple contact points at the same time t and if their Euclidean distance is less than 5cm then it is assumed the Encoder incorrectly split contact points at tile boundaries and they should be recombined. The new combined $x_a\ y_a$ location is chosen based off each points w_p weight value instead of merely splitting the difference.



$$t_{merged} \leftarrow t_1 \vee t_2$$

$$x_{merged} \leftarrow \frac{w_1 * x_1 + w_2 * x_2}{w_1 + w_2}$$

$$y_{merged} \leftarrow \frac{w_1 * y_1 + w_2 * y_2}{w_1 + w_2}$$

$$w_{merged} \leftarrow w_1 + w_2$$

Figure 5.11: Merging Incorrectly Split Contact Points from Encoder

Figure 5.12 shows an example of the contact points predicted from a person walking across the smart floor and the corresponding true footfall locations extracted with a high-resolution pressure mat, described in Chapter 7, placed on top of the floor. This illustrates the approach's ability to successfully extract contact points even in multi-contact situations without the need for any multi-contact training data.

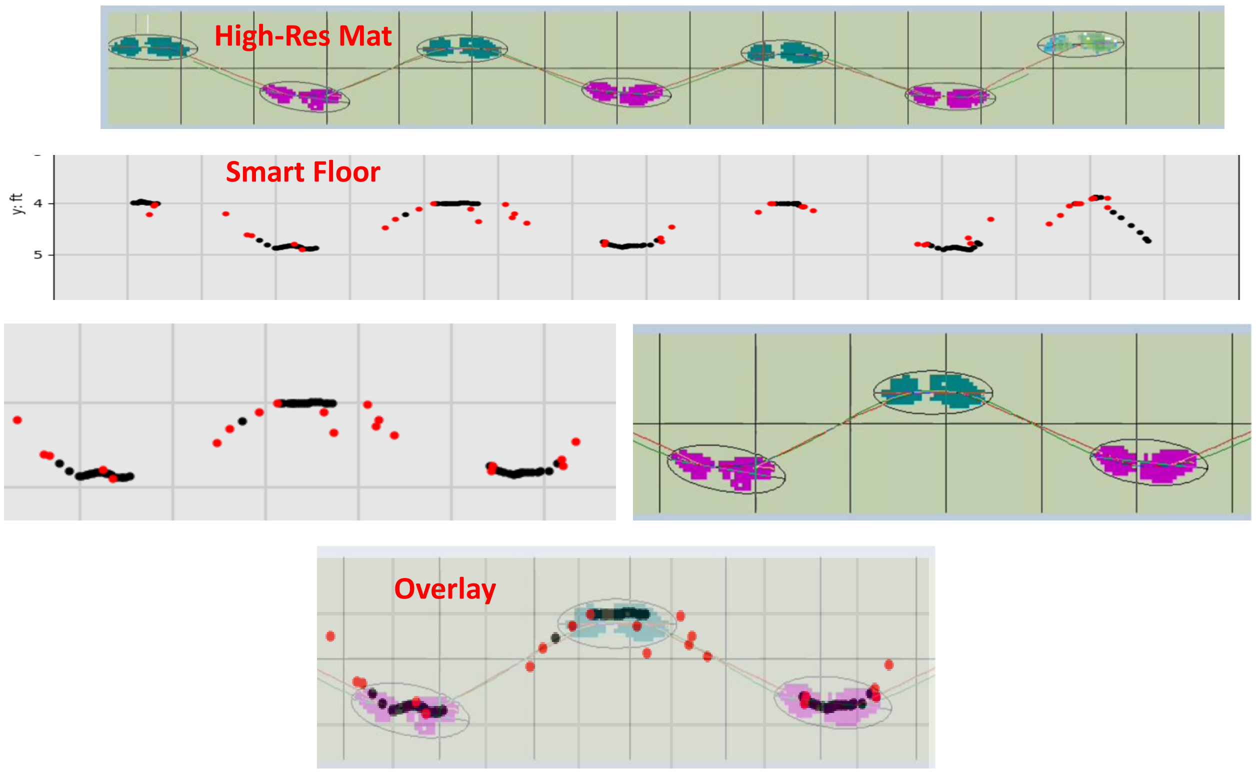


Figure 5.12: Smart Floor vs High-Resolution Mat Contact Points

6 Clustering and Gait Analysis

6.1 Footfall Clustering

After the absolute contact points are successfully extracted using the Encoder model, they are clustered to segment individual footfalls of a person while also removing outliers. A time series of *k* contact points is expressed as $CP = \{cp^{(0)}, cp^{(1)}, \ldots , cp^{(k-1)}\}$ where $cp^k = (t^k, x^k, y^k, w^k, \beta^k)$:

- t = timestamp index of original walking data
- x = absolute floor coordinate along the x-axis in cm or ft
- y = absolute floor coordinate along the y-axis in cm or ft
- w = calibrated weight values of the contact point in kg
- β = stance type of contact point (1 = single, 2 = dual, ≥ 3 = dual+)

The 25Hz sample rate of the floor is fast enough to capture the moment two feet are in contact with the floor at the same time even at a brisk walk. Stance type β is a variable to help describe which support phase (single or double) a contact point belongs to when a person is walking. Labeling contact points as *dual stance type* can also support segmenting and signifying the start or end of an individual footfall. Contact point data of *dual+ stance type* was deemed Encoder output outliers and were discarded before foot clustering began. Stance type was determined by counting the number of shared contact points at timestamp *t* within the *CP* data. Figure 6.1 below shows an entire walking sequence of multiple passes over the entire floor with single stance points in black, dual in red, and dual+ in cyan while Figure 6.2 shows the same for a single walking pass.

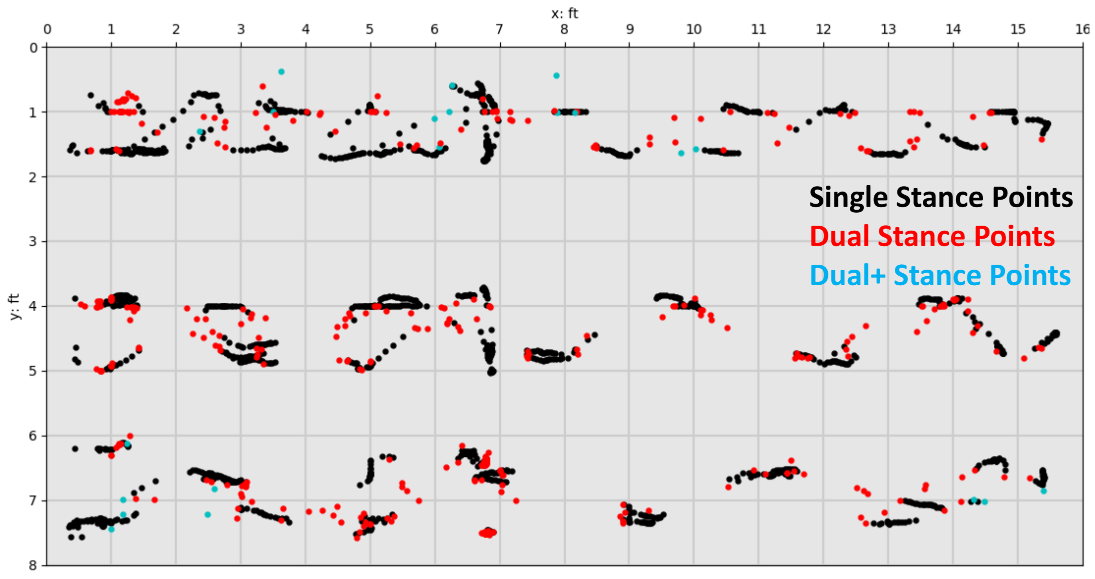


Figure 6.1: Unclustered Contact Points of an Entire Walking Sequence

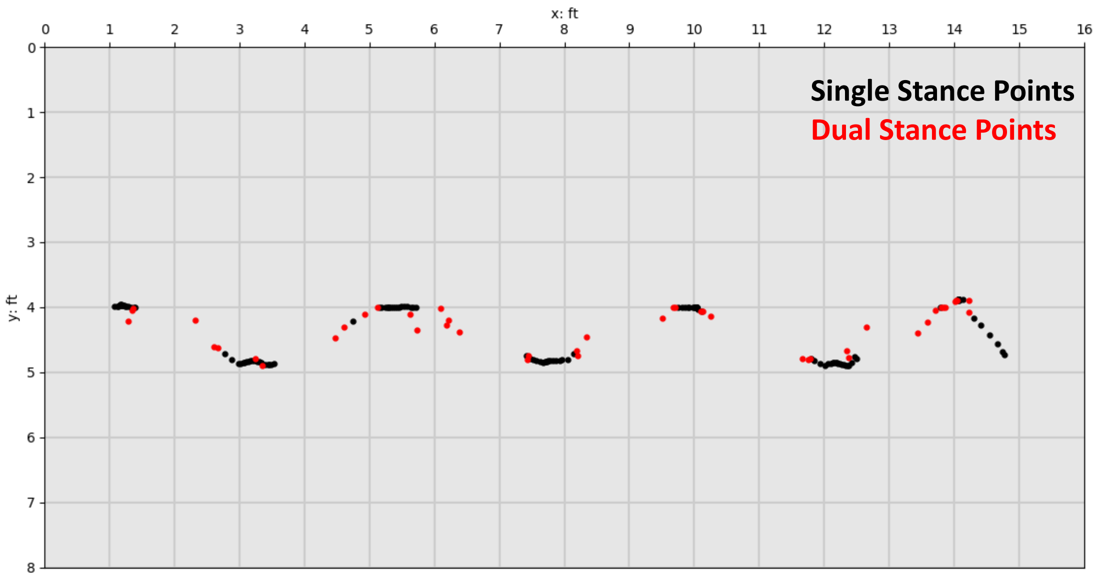


Figure 6.2: Unclustered Contact Points of a Single Walking Pass

6.1.1 Normalizing Contact Point Features

Before Hierarchical Clustering Analysis (HCA) is performed to determine the clusters representing individual footfalls, contact point features are normalized. The stance type β is not used for HCA and therefore not normalized here. The *CP* features *t*, *x*, and *y* are scaled according to each feature's median rate of change considering the time between two consecutive floor samples taken at 25Hz (1/25Hz = 0.04sec). Additionally, location features *x* and *y* are jointly normalized using the Euclidean distance of their sequential differences to avoid axis bias in case one direction was spanned differently than the other during an entire walking sequence, as shown in Figure 6.3. Weight feature *w* is normalized with the standard min-max method [0…1]. Below, *diff()* represents finding the sequential n-th order discrete difference along the feature column.

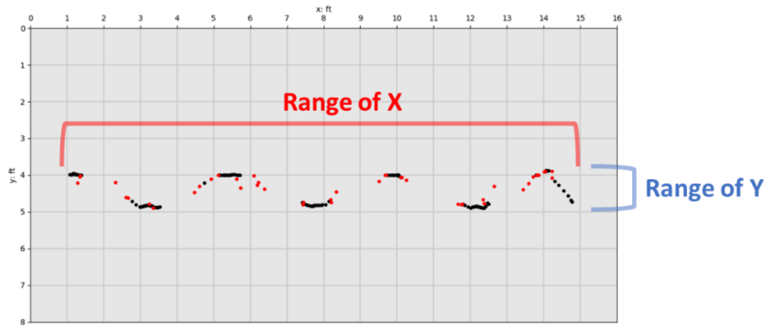


Figure 6.3: Contact Point Span Across *X* and *Y* Axes

$$t' \leftarrow \frac{t}{median(diff(t)) * 25Hz}$$

$$x' \leftarrow \frac{x}{median(diff(dist_{L2}(x,y))) * 25Hz}$$

$$y' \leftarrow \frac{y}{median(diff(dist_{L2}(x,y))) * 25Hz}$$

$$w' \leftarrow \frac{w - min(w)}{max(w) - min(w)}$$

6.1.2 Recursive Hierarchical Clustering Analysis

Using the normalized feature vectors of all contact points, HCA (agglomerative method) is performed to segment and cluster points together into individual foot falls using minimum linkage and Euclidean distance for the distance metric. When viewing the results of an HCA as a dendrogram, the middle of the largest vertical distance between a cluster merge is picked as the criterion to “best cut” the tree and decide how many clusters is optimal. In practice this is achieved by finding the largest jump in the computed linkage matrix. Since a single global “best cut” spanning the entire dendrogram does not guarantee finding each individual footfall cluster, local “best cuts” must be found within groupings of groupings deep within the tree. The recursive element of the HCA foot clustering algorithm will continue to further split clusters with the same “best cut” method until the max Euclidean distance within a cluster is under a maximum “foot length” threshold of 35.56 cm (14 in) while also discarding clusters that contain fewer than 5 individual contact points (single observations).

To illustrate the performance of this recursive hierarchical clustering approach, Figure 6.4 shows the contact points of a sample pass across the floor, Figure 6.5 shows the resulting dendrogram as well as the first cutoff point, before Figure 6.6 shows the resulting final footfall clusters.

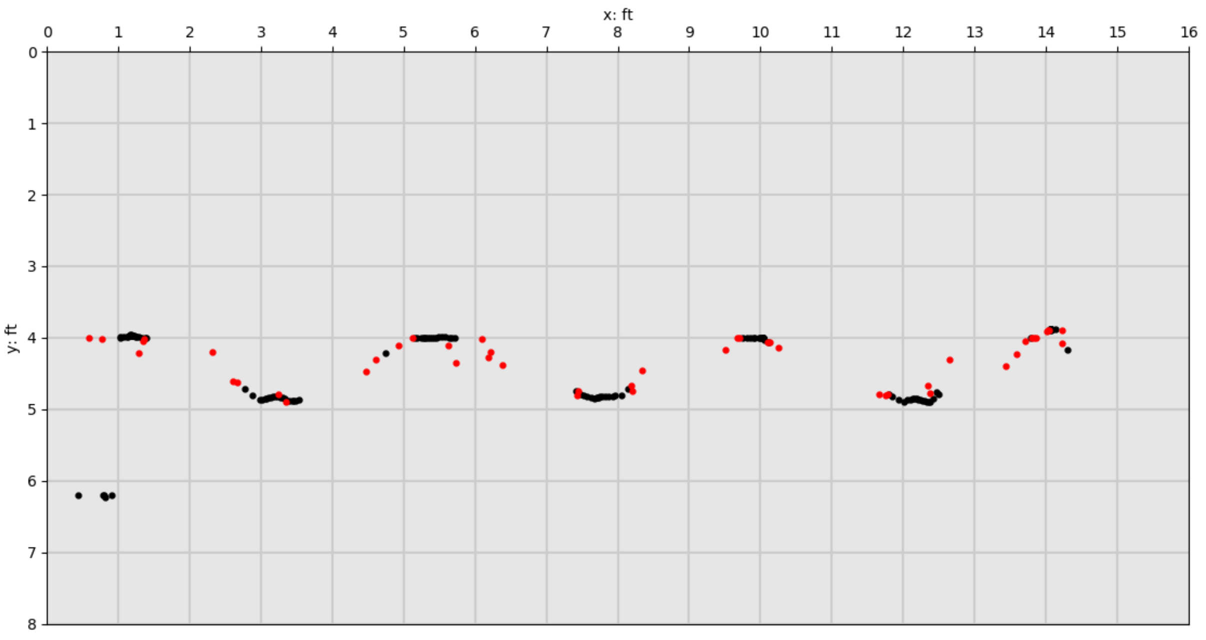


Figure 6.4: Unclustered Contact Points of a Single Walking Pass

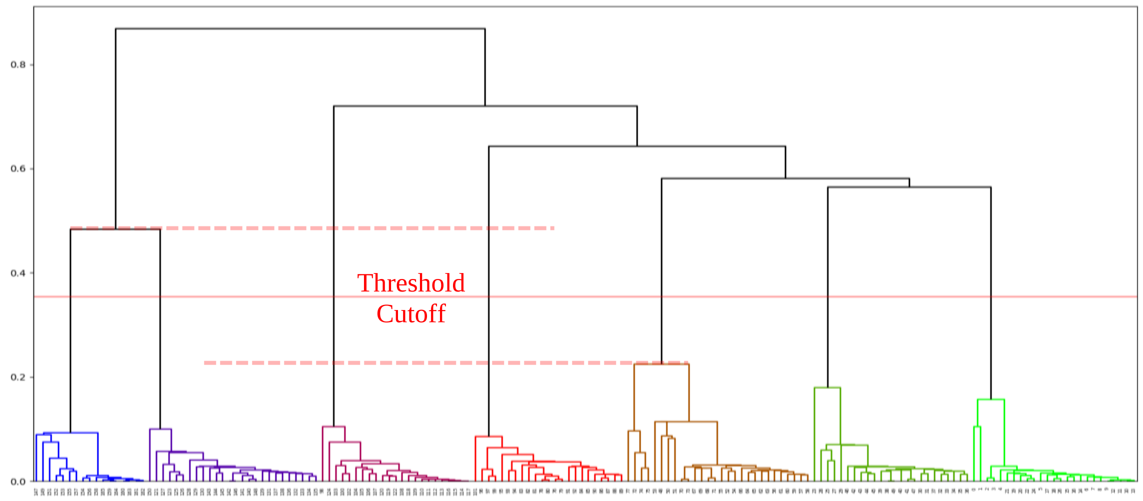


Figure 6.5: Dendrogram of HCA Results

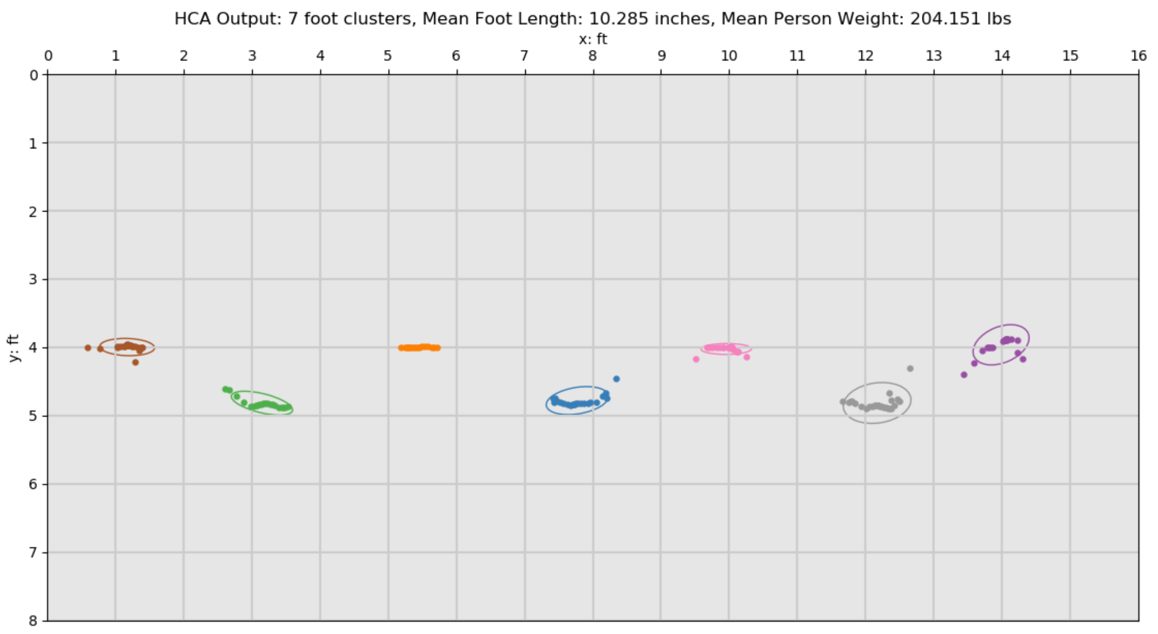
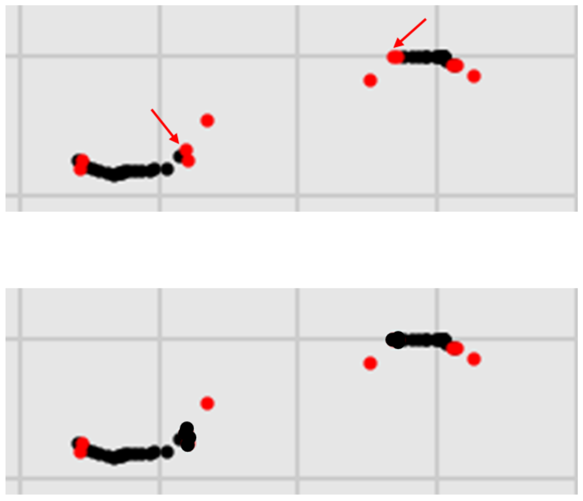


Figure 6.6: Footfall Clusters After HCA (7 Found)

6.1.3 Merging Dual Stance Contact Points

After a set of *m* foot clusters are found ($FC = \{fc^{(0)}, fc^{(1)}, \dots, fc^{(m-1)}\}$ where $fc^{m} = \{cp^{(0)}, cp^{(1)}, \dots, cp^{(n-1)}\}$ and $cp^{n} = (t^{n}, x^{n}, y^{n}, w^{n}, \beta^{n})$), if there exists any *dual stance* points who's pair is within the same cluster they are merged together and become *single stance* type. If a *dual stance* point's pair belongs to another foot cluster they are left unchanged.



$$t_{merged} \leftarrow t_1 \vee t_2$$

$$x_{merged} \leftarrow \frac{w_1 * x_1 + w_2 * x_2}{w_1 + w_2}$$

$$y_{merged} \leftarrow \frac{w_1 * y_1 + w_2 * y_2}{w_1 + w_2}$$

$$w_{merged} \leftarrow w_1 + w_2$$

$$\beta_{merged} \leftarrow 1$$

Figure 6.7: Merging Dual Stance Points within Single Footfall Clusters

Algorithm 2 shows a high-level view of the foot-cluster extraction process and Algorithm 3 shows in more detail the recursive hierarchical clustering approach.

Algorithm 2 High-Level Extraction of Foot Clusters

```
input:  set of raw contact points CP_raw : {cp^(0), … , cp^(k-1)} where cp^k = (t^k, x^k, y^k, w^k, β^k)
output: set of foot clusters FC : {fc^(0), … , fc^(m-1)} where fc^m = {cp^(0), cp^(1), … , cp^(n-1)}

function extractFootClusters(CP_raw)
global foot_cluster_number ← 0
global max_foot_length_threshold ← 14        // inches
global FC ← empty set

for cp in CP_raw do
        cp.β ← count(cp.t = CP_raw.t)         // determine stance type by
                                              // counting common timestamps
HCA(CP_raw)

return FC
```

Algorithm 3 Recursive HCA

```
input: set of contact points $CP$ : $\{cp^{(0)}, \ldots, cp^{(k-1)}\}$ where $cp^k = (t^k, x^k, y^k, w^k, \beta^k)$

function HCA($CP$)
for $cp$ in $CP$ do
    $cp.t \leftarrow cp.t$ / $median(diff(CP.t))$*25Hz    // normalize features
    $cp.x \leftarrow cp.x$ / $median(diff(dist_{L2}(CP.x,CP.y)))$*25Hz
    $cp.y \leftarrow cp.y$ / $median(diff(dist_{L2}(CP.x,CP.y)))$*25Hz
    $cp.w \leftarrow (cp.w - min(CP.w)) / (max(CP.w) - min(CP.w))$

$linkage\_matrix \leftarrow computeLinkageMatrix(CP$, single_linkage, euclidean)

$min\_cutoff \leftarrow linkage\_matrix[argmax(diff(linkage\_matrix))]$
$max\_cutoff \leftarrow linkage\_matrix[argmax(diff(linkage\_matrix)) + 1]$
$cutoff \leftarrow ((max\_cutoff - min\_cutoff) / 2) + min\_cutoff$

$clusters \leftarrow agglomerativeClustering(CP$, single_linkage, euclidean, $cutoff)$
$clusters \leftarrow denormalizeFeatures(clusters)$    // revert to true $xy$ coordinates

for $cluster$ in $clusters$ do
        if $cluster.num\_points > 5$ then
                $extreme\_points[0] \leftarrow cluster[argmax(cluster.x)]$
                $extreme\_points[1] \leftarrow cluster[argmin(cluster.x)]$
                $extreme\_points[2] \leftarrow cluster[argmax(cluster.y)]$
                $extreme\_points[3] \leftarrow cluster[argmin(cluster.y)]$

                $max\_euc\_dist \leftarrow 0$

                for $point\_1$ in $extreme\_points$ do
                        for $point\_2$ in $extreme\_points$ do
                                $curr\_euc\_dist \leftarrow euclideanDistance(point\_1, point\_2)$

                                if $curr\_euc\_dist > max\_euc\_dist$ then
                                        $max\_euc\_dist \leftarrow curr\_euc\_dist$

                if $max\_euc\_dist < max\_foot\_length\_threshold$ then
                        append $cluster$ to $FC$ with $foot\_cluster\_number$        // valid foot cluster
                        $foot\_cluster\_number \leftarrow foot\_cluster\_number + 1$
                else
                        HCA($cluster$)    // invalid foot cluster, recursive call
```

6.1.4 Segmenting into Gait Sequences

Once foot clusters are found, gait sequences can be constructed for follow-up gait parameter extraction. For this, all foot clusters are sorted according to the mean *t* of each cluster's set of contact points. The entire set of foot clusters *FC* must be segmented into a set of gait sequences where reliable gait parameters can be extracted. This involves separating gait sequences if a person physically leaves the floor and returns and when a walking person is making a turn. Extracting step, stride, and other gait parameters during a turning sequence is deemed unreliable and only gait sequences where a person walks semi-straight are to be used.

If there exists a time gap of at least 1.5 secs in the sequential set of foot clusters, it is assumed the person has left the smart floor and the foot clusters are separated into different gait sequences. Within each gait sequence, a local line of progression (LOP) is calculated for each stride using sets of 3 sequential footfalls $fc^{(n-1)}$, $fc^{(n)}$, and $fc^{(n+1)}$. The *LOP vector* is drawn between the *xy* mean of the 1st and 3rd foot clusters as they should represent the same foot. The *LOP angle* for the middle foot cluster $fc^{(n)}$ is calculated by applying the *arctan2()* function on the *LOP vector*. The *LOP angles* for the first and last foot cluster within a gait sequence are merely the copy of the closest neighbor since those feet do not exist within the "middle" of a complete stride. Figure 6.8 shows the LOP and foot angles for a sequence of 3 foot clusters.

$$\overrightarrow{LOP\ Vector}^{(n)} \leftarrow \begin{pmatrix} x_{mean}^{(n+1)} - x_{mean}^{(n-1)} \\ y_{mean}^{(n+1)} - y_{mean}^{(n-1)} \end{pmatrix}$$

$$LOP\ Angle^{(n)} \leftarrow arctan2(\overrightarrow{LOP\ Vector_y}^{(n)}, \overrightarrow{LOP\ Vector_x}^{(n)})$$

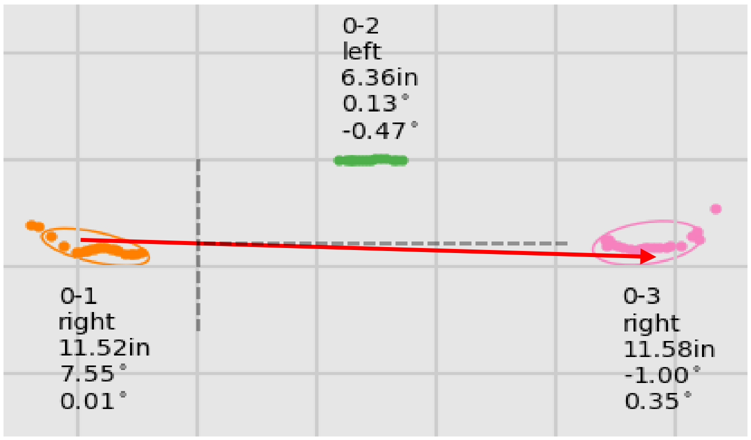


Figure 6.8: LOP Vector and Angle Visualization

The *LOP angles* of each foot cluster are used to determine whether a person is walking semi-straight or turning on the smart floor. Iterating sequentially over the set of foot clusters in pairs, if the absolute difference between each foot cluster's *LOP angle* is greater than 10 degrees, then the latter of the two is labeled a turn. This comparison is performed for all foot clusters and the gait sequence is further split into more gait sequences wherever turns are found. The turn data is not thrown out since it offers location data that could be useful for tracking, but they are not used in the subsequent gait analysis. Figure 6.9 shows sample data of an initial lone gait sequence split into two after turns were discovered within the walking path, indicated in red. During this time and turn splitting process, if a resulting gait sequence contains less than 3 total foot clusters then it is discarded since meaningful gait parameters require many sequential footfalls to calculate.

Figure 6.10 shows an example of the foot clusters of a person walking across the floor in multiple passes with turns before gait sequence extraction and identification of turns. Based on these clusters, Figure 6.11 shows the classification of foot clusters into straight and turn types, clearly identifying the straight line passes and the separating turns between them.

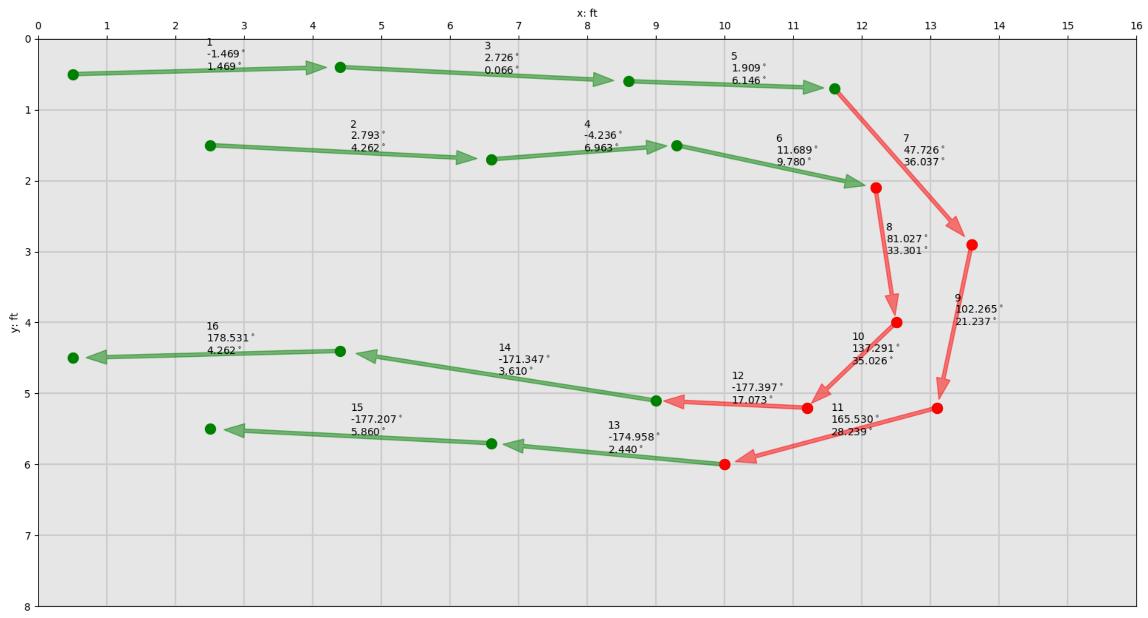


Figure 6.9: Gait Sequence of Straight and Turn Data

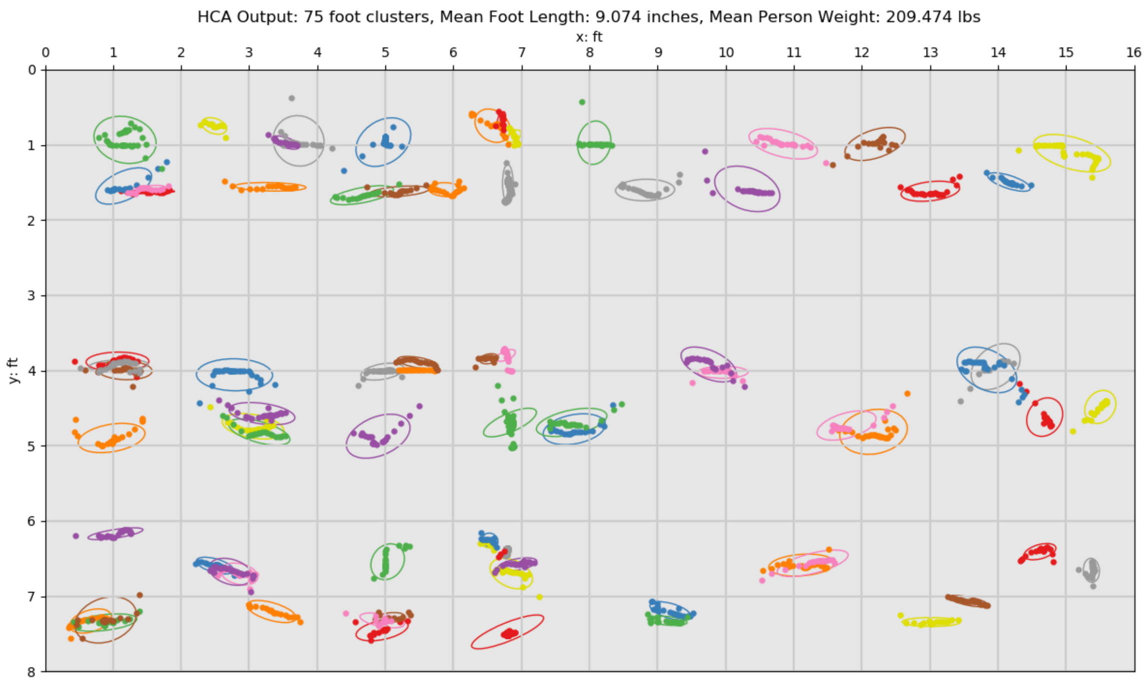


Figure 6.10: Foot Clusters before Gait Sequence Segmentation

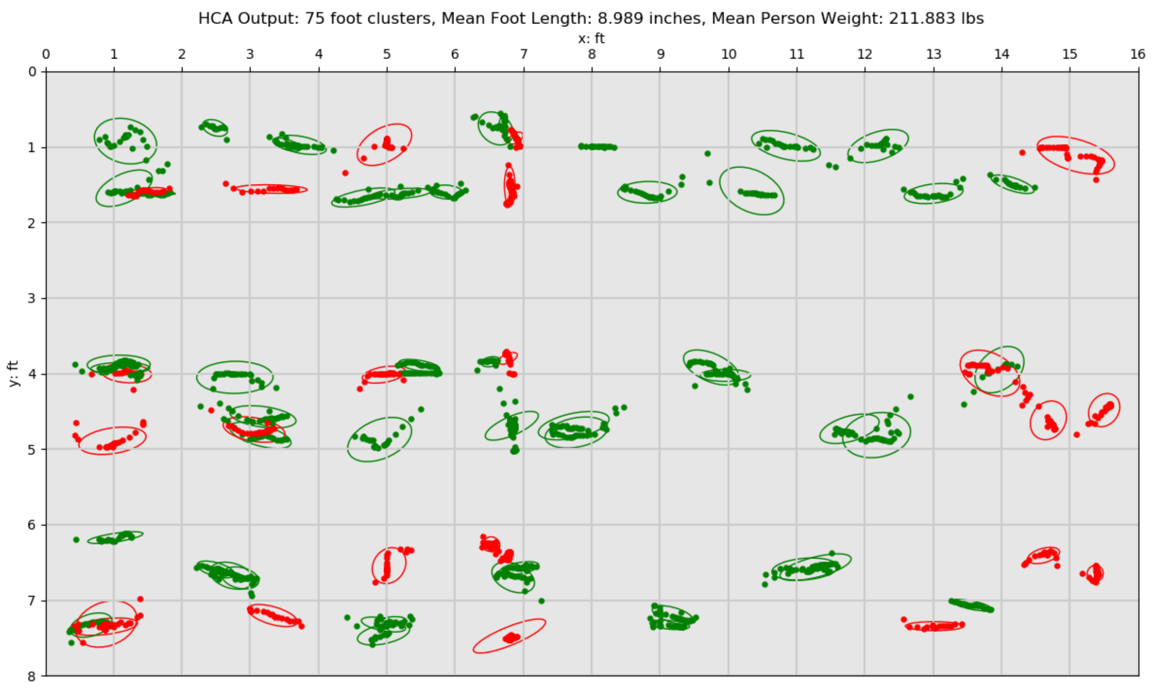


Figure 6.11: Walking Sequence after Gait Sequence Segmentation (green = kept, red = discarded turns)

After each gait sequence's foot clusters are finalized, each foot cluster is labeled as right or left. This process assumes the person is walking forward. Using sets of 3 sequential foot clusters $fc^{(n-1)}$, $fc^{(n)}$, and $fc^{(n+1)}$ the middle foot cluster's left/right foot type is determined by using the sign of the cross product of the vectors from $fc^{(n-1)}$ to $fc^{(n+1)}$ and $fc^{(n-1)}$ to $fc^{(n)}$. The vectors span to the *xy* mean of each foot cluster. Using known data and intuition, a positive sign indicated the middle $fc^{(n)}$ is a right foot and negative indicated left. The small algorithm also labels $fc^{(n-1)}$ to $fc^{(n+1)}$ the opposite of the found $fc^{(n)}$ label. This process iterates over the entire gait sequence until all foot clusters are labeled with a foot type, resulting in the identification illustrated in Figure 6.12.

$$footType^{n} \leftarrow \begin{cases} right & if((\overrightarrow{fc^{n-1}fc^{n+1}} \times \overrightarrow{fc^{n-1}fc^{n}}) > 0) \\ left & else \end{cases}$$

$$footType^{n-1} \leftarrow \begin{cases} right & if(footType^{n} = left) \\ left & else \end{cases}$$

$$footType^{n+1} \leftarrow \begin{cases} right & if(footType^{n} = left) \\ left & else \end{cases}$$

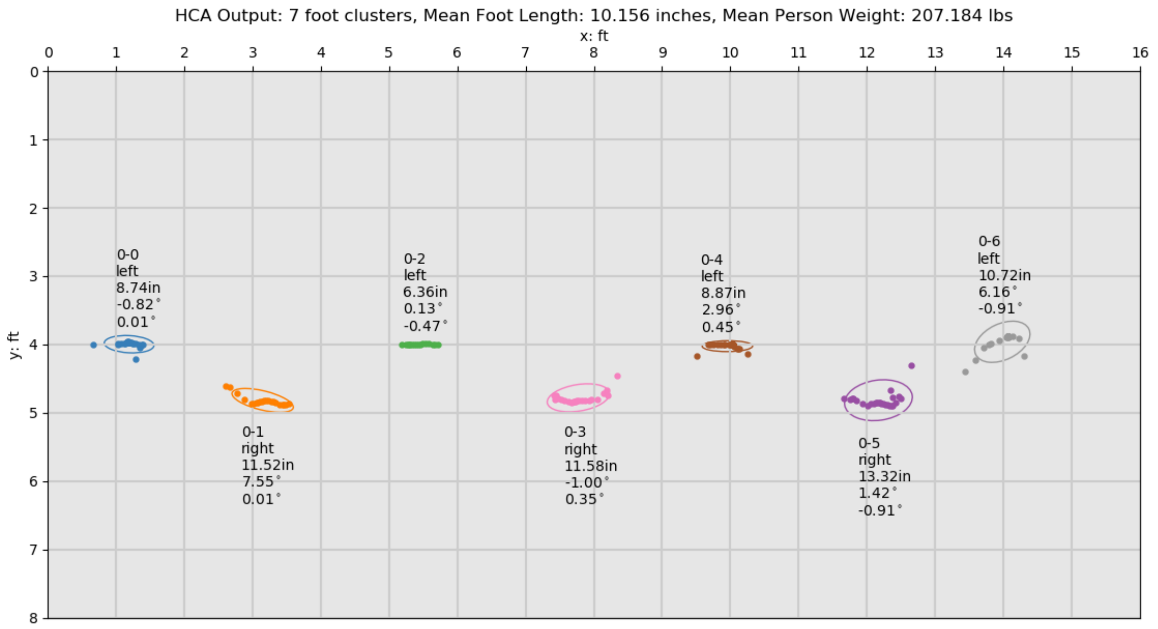


Figure 6.12: Person Walking Left to Right with Foot Type Labels

6.2 Gait Analysis

After each gait sequence only contains semi-straight walking segments without turns and each foot cluster's *footType* and *LOP angle* are calculated, gait analysis can be performed. Each gait sequence will generate 20 total gait parameters (10 left, 10 right): foot weight, foot length, foot angle, step length, stride length, step width, step time, step speed, stride time, and stride speed.

6.2.1 Foot Weight

Each foot cluster's weight (kg) is calculated by finding the mean of the middle 50% of the sorted single contact point calibrated weight values in kg. Only computing the mean weight on the middle subset helps to exclude any valid *dual stance* points who usually exhibit a smaller weight since they are paired with another contact point also in contact with the floor at the same time. This also helps to remove heel strike and toe off moments where weight spikes might occur due to downward forces while walking.

6.2.2 Foot Length

A cluster's foot length (cm) is calculated by finding the maximum Euclidean distance between the set of the most extreme *xy* points within the cluster.

6.2.3 Foot Angle

A cluster's foot angle (degrees) is calculated using the *LOP vector* of the cluster and the principal component of the contact points within the foot cluster. Principal Component Analysis (PCA) is performed on the set of *xy* contact points minus any dual stance points detected at the start or end of the foot cluster. These points are not included so the predominant contributing factor to the principal component, and ultimately the foot angle, is the stable region during the mid-stance portion of the gait cycle when the foot is most flat with the floor and gives the most accurate description of direction and angle. After PCA is performed on the *xy* data, the unit vector of the principal component *PC1* is compared against the *LOP vector* to calculate the foot cluster's angle.

$$\overrightarrow{PC1}\ Angle \leftarrow arctan2(\overrightarrow{PC1}_y, \overrightarrow{PC1}_x)$$

$$\overrightarrow{LOP}\ Angle \leftarrow arctan2(\overrightarrow{LOP}_y, \overrightarrow{LOP}_x)$$

$$Foot\ Angle \leftarrow \overrightarrow{PC1}\ Angle - \overrightarrow{LOP}\ Angle$$

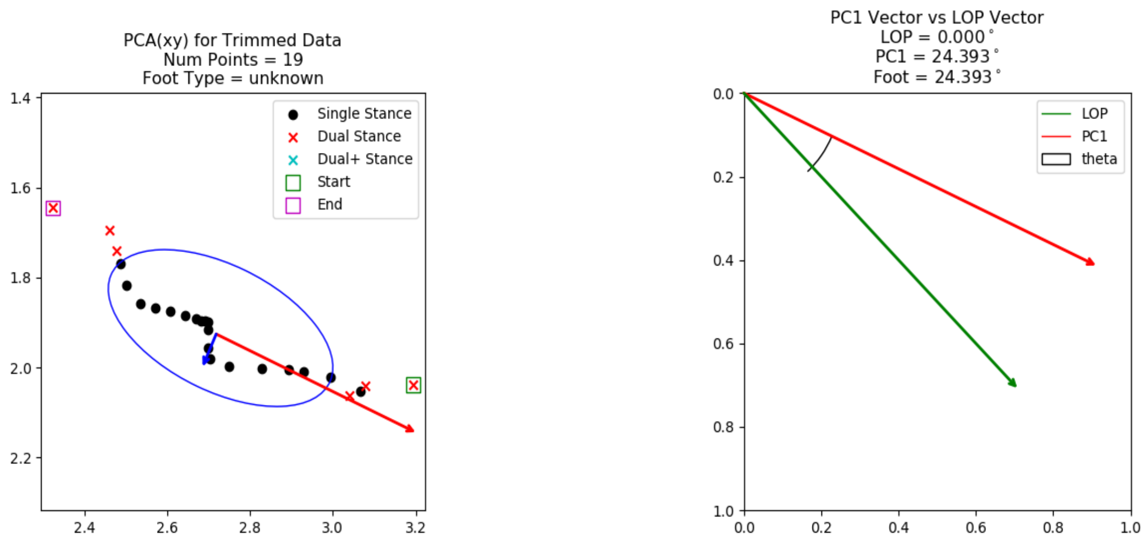


Figure 6.13: PCA and Foot Angle (theta) Calculations

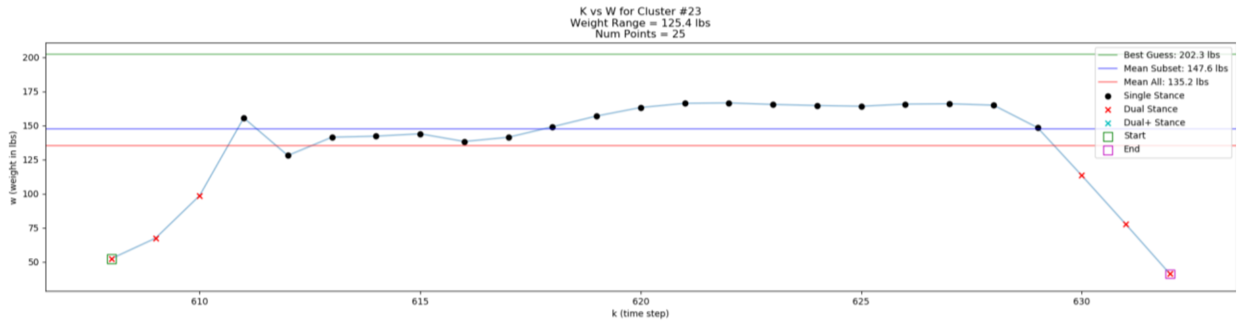


Figure 6.14: Sequential Pressure Profile of a Foot Cluster's Contact Points over Time

6.2.4 Step Length

Two sequential foot clusters are required to calculate a step length (cm), $fc^{(n)}$ and $fc^{(n+1)}$. The toe-off *xy* points of both clusters and the *LOP vector* from the second cluster are needed. The toe-off points are determined to be the last *xy* coordinate in the time-sorted set of contact points per foot cluster. Step length is not measured by absolute *xy* Euclidean distance of the two toe-off points, it is measured along the line of progression (LOP) the person is walking. The toe-off points of both clusters are projected on to the *LOP line* found using the *LOP vector's* slope and y-intercept. The *LOP vector* is found using cluster *xy* means, not toe-offs so both have to be projected. The final step length is the Euclidean distance between the projected toe-off points of both clusters. The step length's left/right label matches the *footType* label of the $fc^{(n+1)}$ cluster, the foot that travelled the distance to complete the step.

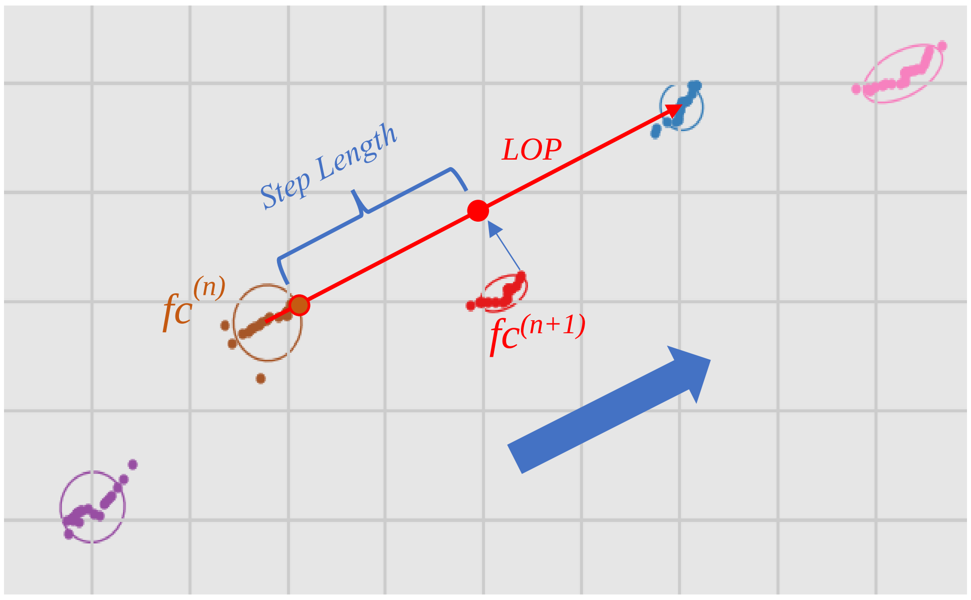


Figure 6.15: Step Length Calculation

6.2.5 Stride Length

Two toe-off points of sequential foot clusters of the same type, left or right, are required to calculate the stride length (cm), $fc^{(n)}$ and $fc^{(n+2)}$. The *LOP vector* of the middle cluster $fc^{(n+1)}$ is used for projecting the toe-off points. With these values the stride length calculation follows the same procedure as step length.

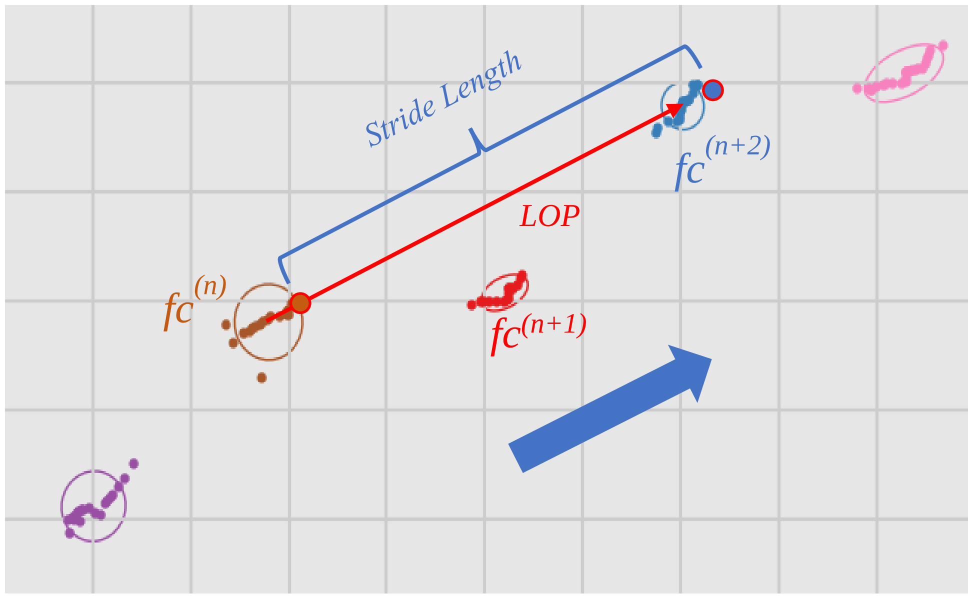


Figure 6.16: Stride Length Calculation

6.2.6 Step Width

Step widths (cm) are calculated by iterating over every foot cluster one-by-one excluding the first and last. Each foot cluster's *xy* mean point is projected onto its own *LOP vector* and the projected point's travel distance is the step width. The step width's left/right label matches that of the foot cluster in question.

6.2.7 Step Time

Step time (seconds) is calculated by using the same foot clusters for calculating step length. The step time is merely the difference in timestamp *t* values between the two toe-off contact points divided by the smart floor's data sampling rate of 25Hz.

6.2.8 Step Speed

Step speed (cm per sec) is calculated by iterating over each step length and dividing by its respective step time pair.

6.2.9 Stride Time

Stride time (seconds) is calculated by using the same foot clusters for calculating stride length. The stride time is merely the difference in timestamp *t* values between the two toe-off contact points divided by the smart floor's data sampling rate of 25Hz.

6.2.10 Stride Speed

Stride speed (cm per sec) is calculated by iterating over each stride length and dividing by its respective stride time pair.

6.2.11 Gait Parameter Means and Variances

If the walking sequence is sufficiently large means and variances over the entire dataset of gait values are calculated. For each of the 20 gait features, the middle 50% of the sorted feature vector is used to calculate the mean and variance to help remove any small or large outliers.

Figure 6.17 and Figure 6.18 show examples of the calculated gait features for a single pass and a set of passes, respectively.

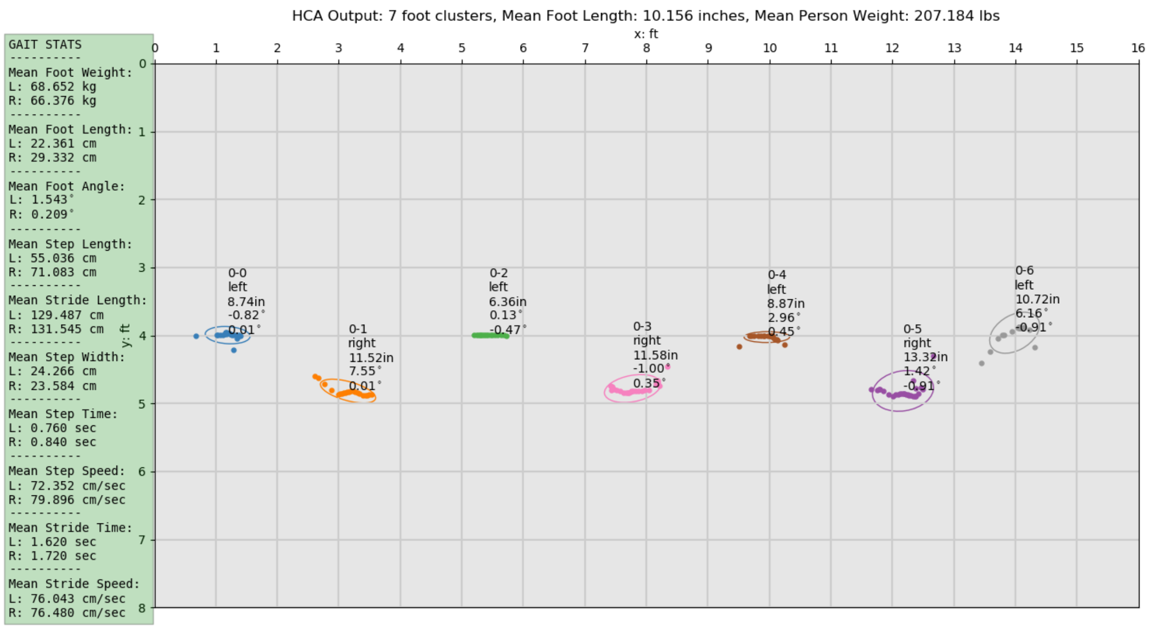


Figure 6.17: Single Pass Showing Gait Means

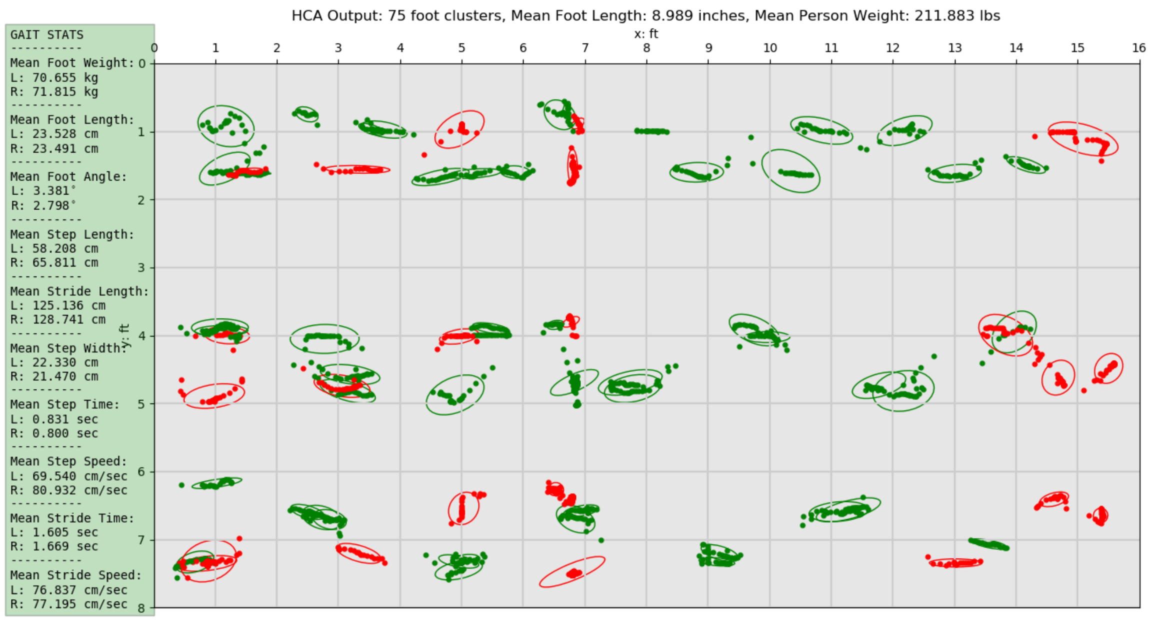


Figure 6.18: Entire Walking Sequence (with red turns removed) Showing Gait Means

6.3 Future Work

All the work described in this chapter is under the assumption that only a single person is walking on the smart floor at any given time. While the recursive HCA foot clustering can successfully segment different people's footfalls, it does not possess the framework to account for overlapping timestamps and assign footfalls and gait sequences to Person A, B, …, etc. A potential method to successfully segment multiple people would involve tracking possibly in the form of a Kalman Filter to track and anticipate where and when the next footfall of a person should take place.

When multiple people are not only walking on the floor simultaneously but have very near footfall locations special consideration to prevent incorrect footfall mergers needs to be accounted for. If the pressure profiles of the first few steps can be established (time vs. weight of contact points) then possibly Dynamic Time Warping (DTW) could be used to calculate a likelihood that a new footfall belongs to Person A, Person B, etc. This research work is a crucial next step in order to make this established work applicable to the SmartCare apartment's smart floor.

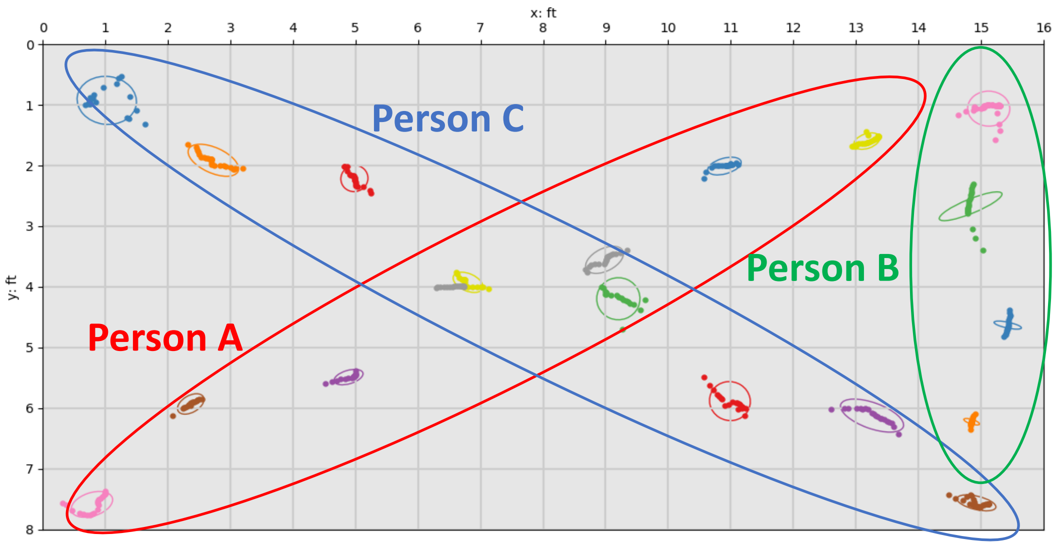


Figure 6.19: Future Person Separation and Identification

7 High-Resolution Mat Study

7.1 Overview

Over the Summer of 2019, an IRB study (protocol number 2019-0291) was performed with the lab's smart floor and a high-resolution floor mat (*ProtoKinetics 16ft. Zeno Walkway* [33] and *PKMAS software* [34]). The IRB protocol was titled *A Smart Floor Study to Measure Weight, Gait, and Activity*. The study recruited 10 subjects to perform simple walking patterns across our smart floor and the *ProtoKinetics* mat. Data was recorded simultaneously for the floor and mat. The *ProtoKinetics* mat offers very high-resolution pressure data with sensors spaced 0.5in (1.27cm) apart compared to our floor's resolution of 1sqft. Each subject's age, height, weight, gender, and foot length were recorded prior to each session. Video footage was also recorded for reference when examining walking data (no subject's face will ever appear in any SmartCare publication).

The study's motivation was to acquire ground truth walking data for comparison when examining our smart floor's data and writing software. The *ProtoKinetics* mat offers very fine resolution location information and can extract individual footsteps which will be used to test the contact point extraction software of our smart floor. The mat also calculates many gait parameters: foot center and heel locations, step length, stride length and width, step time, stride time, stride velocity, gait cycle time, center of pressure, center of mass, cadence, and foot fall counts.

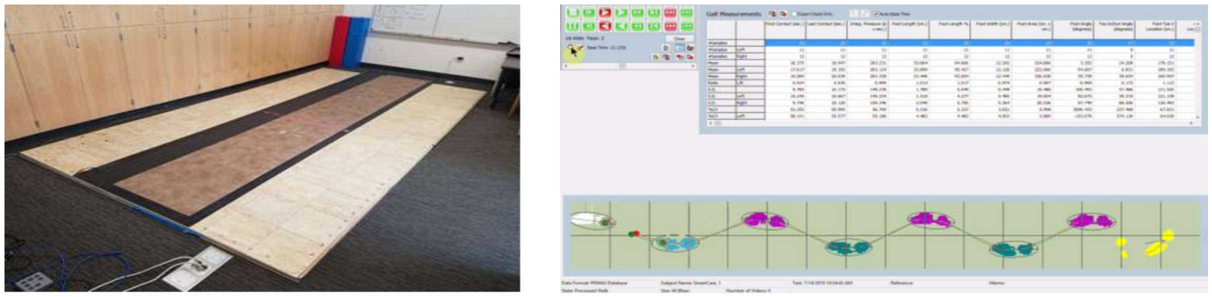


Figure 7.1: ProtoKinetics Zeno Walkway Mat and PKMAS Software

7.2 Walking Procedure

The subjects performed a total of 6 walking sequences on various segments of the lab's smart floor. The final two taking place on the high-resolution mat placed on top of the smart floor. The even numbered sequences in Figure 7.2 involved a ~5 second standing phase in the middle of the floor before returning off the floor.

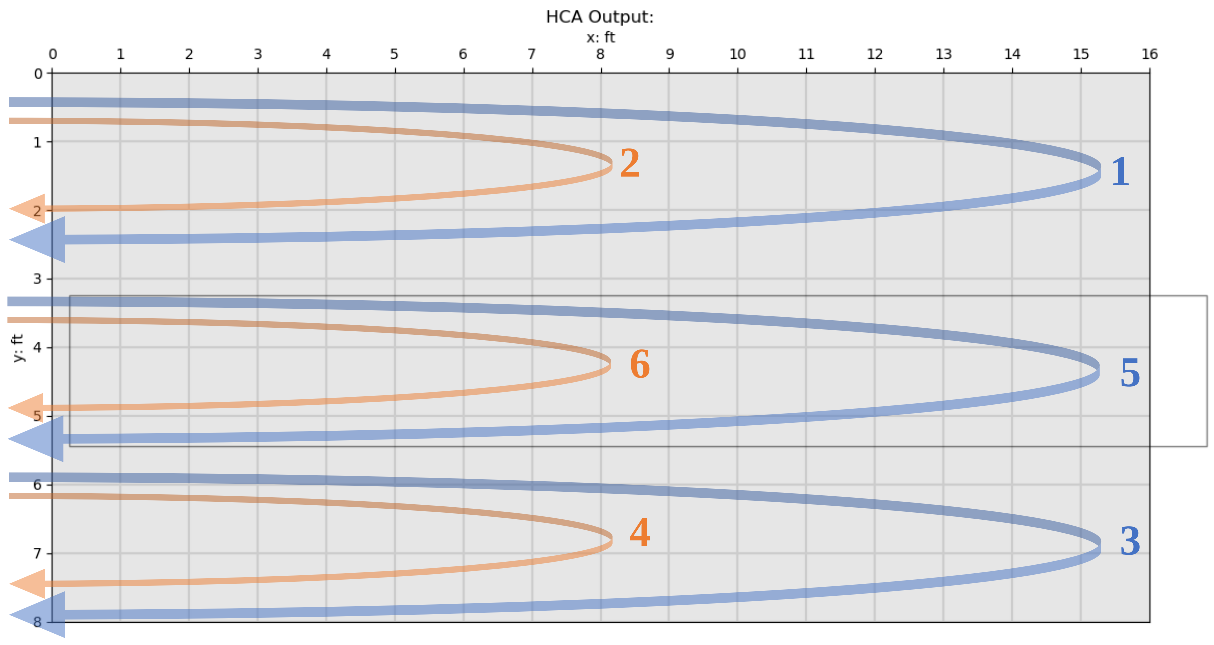


Figure 7.2: Walking Sequences on Smart Floor with Outline of High-Resolution Mat

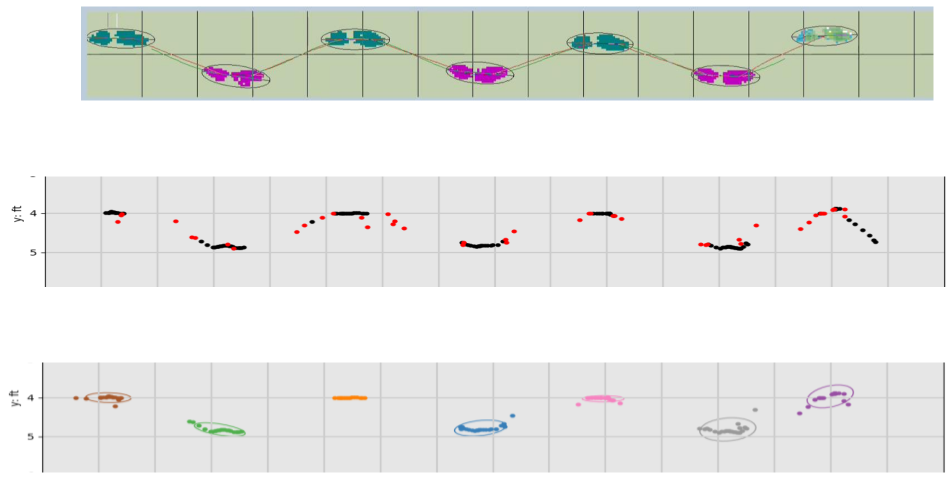


Figure 7.3: Sample of Smart Floor vs High-Resolution Mat Contact Points and Foot Clusters

7.3 Results of High-Resolution Mat vs. Smart Floor

The PKMAS gait means in the tables below were computed using the clean first pass of walking sequence 5 in Figure 7.2 without turns. Smart floor data in the tables shows both entire walking sequence of the entire floor and trimmed data of sequence 5 to show a direct comparison to the results of the high-resolution mat. Single pass data for the mat, the floor, and their comparisons are shown in Table 7-1, Table 7-2, and Table 7-3, respectively, while equivalent data for the floor and comparisons for all passes is shown in Table 7-5 and Table 7-6. While there are 11 columns of data below, only 10 people generated the data. A person walked twice on different days; subjects 0 and 8 are the same person.

PKMAS Mat - Means of 1st pass of Walking Sequence 5											
Subject:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LEFT - Foot Length (cm)	34.678	30.627	29.976	29.422	29.353	27.052	25.915	28.545	35.393	27.340	27.498
RIGHT - Foot Length (cm)	35.390	30.699	30.720	29.057	28.811	25.884	26.663	28.468	34.503	27.463	27.011
LEFT - Foot Angle (degrees)	-0.528	4.943	0.828	6.514	5.245	5.887	10.976	0.680	2.218	-2.099	5.495
RIGHT - Foot Angle (degrees)	-6.657	-8.095	-3.141	-18.034	-7.538	-0.198	-14.689	-2.095	-2.418	-2.526	-9.320
LEFT - Step Length (cm)	62.821	51.013	57.769	46.829	50.987	61.720	54.413	63.896	67.472	64.869	50.752
RIGHT - Step Length (cm)	69.092	58.431	57.158	55.000	51.862	61.817	55.260	71.665	73.972	65.090	50.933
LEFT - Stride Length (cm)	130.097	109.408	112.563	99.736	102.803	123.828	108.313	135.212	141.618	131.115	102.857
RIGHT - Stride Length (cm)	135.313	110.660	115.209	107.365	102.909	124.693	109.623	138.690	143.050	129.815	100.795
LEFT - Step Width (cm)	22.869	17.856	8.202	14.037	10.796	10.028	8.027	11.953	20.222	7.650	7.561
RIGHT - Step Width (cm)	21.586	17.771	6.240	9.829	10.543	10.239	8.340	9.637	20.733	7.810	8.294
LEFT - Step Time (secs)	0.889	0.591	0.689	0.748	0.911	0.567	0.683	0.596	0.754	0.667	0.628
RIGHT - Step Time (secs)	0.846	0.597	0.706	0.748	0.903	0.606	0.728	0.621	0.738	0.697	0.671
LEFT - Step Speed (cm/sec)	72.899	92.061	79.675	66.719	56.686	105.222	75.084	111.132	94.913	96.305	80.025
RIGHT - Step Speed (cm/sec)	77.902	87.856	82.667	70.927	55.457	102.024	77.689	108.362	92.664	92.970	76.238
LEFT - Stride Time (secs)	1.789	1.189	1.419	1.496	1.814	1.179	1.447	1.217	1.492	1.363	1.286
RIGHT - Stride Time (secs)	1.738	1.280	1.394	1.519	1.859	1.231	1.411	1.283	1.546	1.397	1.329
LEFT - Stride Speed (cm/sec)	72.899	92.061	79.675	66.719	56.686	105.222	75.084	111.132	94.913	96.305	80.025
RIGHT - Stride Speed (cm/sec)	77.902	87.856	82.667	70.927	55.457	102.024	77.689	108.362	92.664	92.970	76.238

Table 7-1: PKMAS Mat Gait Means (single pass)

Smart Floor - Means of 1st pass of Walking Sequence 5

Subject:	0	1	2	3	4	5	6	7	8	9	10
LEFT - Foot Length (cm)	22.361	19.828	23.167	15.277	24.980	23.981	22.453	17.053	20.043	16.616	13.976
RIGHT - Foot Length (cm)	29.332	24.083	32.465	26.354	20.823	26.590	28.393	27.411	23.027	16.320	18.943
LEFT - Foot Angle (degrees)	-0.346	0.996	4.864	-11.385	-3.785	-4.668	-15.935	-8.277	-3.444	2.612	0.706
RIGHT - Foot Angle (degrees)	0.274	8.799	7.476	7.663	1.198	-2.972	15.700	4.516	-5.787	0.310	8.056
LEFT - Step Length (cm)	52.600	56.933	52.396	49.662	51.141	63.066	49.968	71.022	65.052	65.611	50.133
RIGHT - Step Length (cm)	71.077	50.661	56.891	61.382	44.843	58.578	60.597	55.402	59.641	60.337	52.669
LEFT - Stride Length (cm)	127.025	103.771	109.425	109.852	94.443	111.716	108.411	131.142	130.148	121.621	102.145
RIGHT - Stride Length (cm)	131.545	107.034	107.147	103.864	92.018	122.700	105.889	126.518	135.651	132.149	99.750
LEFT - Step Width (cm)	24.266	21.818	10.432	20.783	14.901	10.561	13.370	12.703	22.072	5.130	13.121
RIGHT - Step Width (cm)	23.584	22.246	9.658	20.831	16.205	11.800	12.771	11.068	22.414	5.007	14.479
LEFT - Step Time (secs)	0.720	0.600	0.640	0.680	0.860	0.580	0.640	0.580	0.720	0.640	0.640
RIGHT - Step Time (secs)	0.840	0.520	0.680	0.740	0.940	0.520	0.700	0.600	0.640	0.640	0.680
LEFT - Step Speed (cm/sec)	72.992	91.985	76.983	72.460	55.938	103.951	74.859	111.102	85.096	92.391	74.717
RIGHT - Step Speed (cm/sec)	79.889	97.556	83.398	78.924	51.335	104.409	83.448	84.601	78.475	88.731	80.404
LEFT - Stride Time (secs)	1.580	1.140	1.300	1.420	1.720	1.060	1.340	1.240	1.480	1.360	1.240
RIGHT - Stride Time (secs)	1.720	1.140	1.300	1.420	1.800	1.140	1.340	1.180	1.360	1.240	1.300
LEFT - Stride Speed (cm/sec)	76.397	91.046	80.649	75.316	53.704	103.577	79.062	105.760	87.938	89.427	77.658
RIGHT - Stride Speed (cm/sec)	76.480	94.070	79.079	71.130	56.212	106.166	75.135	97.189	84.782	91.770	77.573

Table 7-2: Smart Floor Gait Means (single pass)

Absolute Errors of Means of 1st pass of Walking Sequence 5 for PKMAS Mat and Smart Floor

Subject:	0	1	2	3	4	5	6	7	8	9	10	Mean
LEFT - Foot Length (cm)	12.317	10.799	6.809	14.145	4.373	3.071	3.462	11.492	15.350	10.724	13.522	**9.642**
RIGHT - Foot Length (cm)	6.058	6.616	1.745	2.703	7.988	0.706	1.730	1.057	11.476	11.143	8.068	**5.390**
LEFT - Foot Angle (degrees)	0.182	3.947	4.036	17.899	9.030	10.555	26.911	8.957	5.662	4.711	4.789	**8.789**
RIGHT - Foot Angle (degrees)	6.931	16.894	10.617	25.697	8.736	2.775	30.389	6.611	3.369	2.836	17.376	**12.021**
LEFT - Step Length (cm)	10.221	5.920	5.373	2.833	0.154	1.347	4.445	7.127	2.420	0.742	0.619	**3.745**
RIGHT - Step Length (cm)	1.986	7.770	0.267	6.382	7.019	3.239	5.337	16.263	14.331	4.753	1.737	**6.280**
LEFT - Stride Length (cm)	3.072	5.637	3.138	10.117	8.360	12.112	0.098	4.070	11.470	9.494	0.712	**6.207**
RIGHT - Stride Length (cm)	3.768	3.626	8.062	3.501	10.891	1.993	3.734	12.172	7.398	2.334	1.045	**5.320**
LEFT - Step Width (cm)	1.397	3.962	2.230	6.746	4.105	0.534	5.343	0.750	1.850	2.520	5.560	**3.181**
RIGHT - Step Width (cm)	1.999	4.475	3.418	11.003	5.662	1.561	4.431	1.432	1.682	2.803	6.185	**4.059**
LEFT - Step Time (secs)	0.169	0.009	0.049	0.068	0.051	0.014	0.043	0.016	0.034	0.027	0.012	**0.045**
RIGHT - Step Time (secs)	0.005	0.077	0.026	0.008	0.037	0.086	0.028	0.021	0.098	0.057	0.009	**0.041**
LEFT - Step Speed (cm/sec)	0.093	0.076	2.692	5.741	0.748	1.271	0.225	0.029	9.816	3.914	5.308	**2.720**
RIGHT - Step Speed (cm/sec)	1.987	9.700	0.731	7.997	4.122	2.385	5.759	23.761	14.189	4.239	4.166	**7.185**
LEFT - Stride Time (secs)	0.209	0.049	0.119	0.076	0.094	0.119	0.107	0.024	0.012	0.002	0.046	**0.078**
RIGHT - Stride Time (secs)	0.017	0.140	0.094	0.099	0.059	0.091	0.071	0.103	0.186	0.157	0.029	**0.095**
LEFT - Stride Speed (cm/sec)	3.498	1.015	0.974	8.597	2.982	1.644	3.978	5.372	6.974	6.878	2.367	**4.025**
RIGHT - Stride Speed (cm/sec)	1.422	6.214	3.588	0.203	0.755	4.142	2.554	11.173	7.882	1.200	1.335	**3.679**
												4.584

Table 7-3: Absolute Errors Between PKMAS Mat and Smart Floor (single pass vs. single pass)

	MAE			MAE
RIGHT - Foot Angle (degrees)	**12.021**		**LEFT - Stride Speed (cm/sec)**	**4.025**
LEFT - Foot Length (cm)	**9.642**		**LEFT - Step Length (cm)**	**3.745**
LEFT - Foot Angle (degrees)	**8.789**		**RIGHT - Stride Speed (cm/sec)**	**3.679**
RIGHT - Step Speed (cm/sec)	**7.185**		**LEFT - Step Width (cm)**	**3.181**
RIGHT - Step Length (cm)	**6.280**		**LEFT - Step Speed (cm/sec)**	**2.720**
LEFT - Stride Length (cm)	**6.207**		**RIGHT - Stride Time (secs)**	**0.095**
RIGHT - Foot Length (cm)	**5.390**		**LEFT - Stride Time (secs)**	**0.078**
RIGHT - Stride Length (cm)	**5.320**		**LEFT - Step Time (secs)**	**0.045**
RIGHT - Step Width (cm)	**4.059**		**RIGHT - Step Time (secs)**	**0.041**

Table 7-4: Average MAE for Each Gait Parameter (single pass vs. single pass)

As shown in Table 7-4 above, using only the exact same trimmed walking sequence for the high-resolution mat and smart floor, across all subjects, step and stride times were the most similar with a maximum MAE of 0.095 seconds. Foot angles were the most dissimilar with the right foot angle's MAE being 12.021 degrees. This is possibly due to using different methods for calculating the foot angle between the PKMAS software and our method described in 6.2.3.

Smart Floor - Means of Entire Walking Sequence											
Subject:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LEFT - Foot Length (cm)	23.528	19.556	21.396	20.758	21.307	25.200	17.503	20.038	19.659	19.286	18.340
RIGHT - Foot Length (cm)	23.491	24.126	24.629	27.106	23.210	26.481	22.204	21.259	22.380	16.895	23.569
LEFT - Foot Angle (degrees)	3.381	-0.143	4.287	-11.122	0.340	-3.114	-7.703	-0.832	-0.013	0.974	0.207
RIGHT - Foot Angle (degrees)	2.798	-0.164	3.868	4.518	0.622	-1.831	10.681	4.383	5.609	-0.630	2.509
LEFT - Step Length (cm)	58.208	48.824	58.795	55.290	49.124	59.358	51.023	63.498	66.068	62.989	46.445
RIGHT - Step Length (cm)	65.811	57.957	58.597	59.420	51.526	62.290	56.461	66.657	71.360	62.957	47.510
LEFT - Stride Length (cm)	125.136	109.416	118.043	110.543	98.259	121.204	105.127	133.048	141.361	125.301	95.334
RIGHT - Stride Length (cm)	128.741	107.002	111.407	112.200	98.910	122.653	108.437	129.961	139.094	122.750	97.058
LEFT - Step Width (cm)	22.330	22.149	7.612	19.756	15.041	12.515	11.350	10.479	21.462	6.863	11.616
RIGHT - Step Width (cm)	21.470	21.674	7.809	20.411	15.809	11.638	10.947	10.725	19.845	6.460	12.061
LEFT - Step Time (secs)	0.831	0.620	0.664	0.750	0.840	0.590	0.723	0.630	0.740	0.667	0.643
RIGHT - Step Time (secs)	0.800	0.617	0.672	0.772	0.885	0.585	0.710	0.610	0.760	0.653	0.689
LEFT - Step Speed (cm/sec)	69.540	79.473	86.935	72.129	57.171	98.678	70.206	101.105	84.622	93.558	72.729
RIGHT - Step Speed (cm/sec)	80.932	94.383	85.164	74.115	56.576	104.601	77.241	106.997	90.419	95.299	72.059
LEFT - Stride Time (secs)	1.605	1.204	1.320	1.515	1.713	1.160	1.408	1.247	1.486	1.320	1.309
RIGHT - Stride Time (secs)	1.669	1.228	1.325	1.540	1.733	1.180	1.392	1.227	1.533	1.280	1.356
LEFT - Stride Speed (cm/sec)	76.837	91.409	88.855	74.434	57.855	103.345	73.978	106.121	90.688	94.970	72.907
RIGHT - Stride Speed (cm/sec)	77.195	87.077	83.134	70.984	57.858	105.441	76.332	105.369	88.431	94.298	72.083

Table 7-5: Smart Floor Gait Means (entire floor)

Absolute Errors of Means of 1st pass of Walking Sequence 5 for PKMAS Mat and Entire Walking Sequence for Smart Floor												
Subject:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**
LEFT - Foot Length (cm)	11.150	11.071	8.580	8.664	8.046	1.852	8.412	8.507	15.734	8.054	9.158	**9.021**
RIGHT - Foot Length (cm)	11.899	6.573	6.091	1.951	5.601	0.597	4.459	7.209	12.123	10.568	3.442	**6.410**
LEFT - Foot Angle (degrees)	3.909	5.086	3.459	17.636	4.905	9.001	18.679	1.512	2.231	3.073	5.288	**6.798**
RIGHT - Foot Angle (degrees)	9.455	7.931	7.009	22.552	8.160	1.634	25.370	6.478	8.027	1.896	11.829	**10.031**
LEFT - Step Length (cm)	4.613	2.189	1.026	8.461	1.863	2.362	3.390	0.398	1.404	1.880	4.307	**2.899**
RIGHT - Step Length (cm)	3.280	0.474	1.439	4.420	0.336	0.473	1.201	5.008	2.611	2.133	3.423	**2.254**
LEFT - Stride Length (cm)	4.961	0.008	5.480	10.808	4.544	2.624	3.186	2.164	0.257	5.814	7.523	**4.306**
RIGHT - Stride Length (cm)	6.572	3.658	3.802	4.836	3.999	2.040	1.186	8.729	3.956	7.065	3.737	**4.507**
LEFT - Step Width (cm)	0.539	4.293	0.590	5.719	4.245	2.488	3.323	1.474	1.240	0.787	4.055	**2.614**
RIGHT - Step Width (cm)	0.116	3.903	1.569	10.583	5.266	1.399	2.607	1.089	0.888	1.350	3.767	**2.958**
LEFT - Step Time (secs)	0.058	0.029	0.025	0.002	0.071	0.024	0.040	0.034	0.014	0.000	0.015	**0.028**
RIGHT - Step Time (secs)	0.045	0.020	0.034	0.024	0.018	0.021	0.018	0.011	0.023	0.044	0.018	**0.025**
LEFT - Step Speed (cm/sec)	3.359	12.588	7.260	5.410	0.485	6.543	4.878	10.027	10.291	2.747	7.296	**6.444**
RIGHT - Step Speed (cm/sec)	3.030	6.527	2.497	3.188	1.119	2.577	0.448	1.365	2.245	2.329	4.179	**2.682**
LEFT - Stride Time (secs)	0.184	0.015	0.099	0.019	0.101	0.019	0.039	0.031	0.006	0.042	0.023	**0.053**
RIGHT - Stride Time (secs)	0.068	0.052	0.069	0.021	0.126	0.051	0.019	0.056	0.013	0.117	0.027	**0.056**
LEFT - Stride Speed (cm/sec)	3.938	0.652	9.180	7.715	1.169	1.876	1.106	5.011	4.224	1.335	7.118	**3.939**
RIGHT - Stride Speed (cm/sec)	0.707	0.779	0.467	0.057	2.401	3.417	1.357	2.993	4.233	1.328	4.155	**1.990**
												3.723

Table 7-6: Absolute Errors Between PKMAS Mat and Smart Floor (single pass vs. entire floor)

	MAE			**MAE**
RIGHT - Foot Angle (degrees)	**10.031**		**LEFT - Step Length (cm)**	**2.899**
LEFT - Foot Length (cm)	**9.021**		**RIGHT - Step Speed (cm/sec)**	**2.682**
LEFT - Foot Angle (degrees)	**6.798**		**LEFT - Step Width (cm)**	**2.614**
LEFT - Step Speed (cm/sec)	**6.444**		**RIGHT - Step Length (cm)**	**2.254**
RIGHT - Foot Length (cm)	**6.410**		**RIGHT - Stride Speed (cm/sec)**	**1.990**
RIGHT - Stride Length (cm)	**4.507**		**RIGHT - Stride Time (secs)**	**0.056**
LEFT - Stride Length (cm)	**4.306**		**LEFT - Stride Time (secs)**	**0.053**
LEFT - Stride Speed (cm/sec)	**3.939**		**LEFT - Step Time (secs)**	**0.028**
RIGHT - Step Width (cm)	**2.958**		**RIGHT - Step Time (secs)**	**0.025**

Table 7-7: Average MAE for Each Gait Parameter (single pass vs. entire floor)

As shown in Table 7-7 above, when using the entire floor's walking sequence captured by the smart floor in comparison to the single pass data of the high-resolution mat, MAE differences were similar to Table 7-4Table 7-3 and followed the same general trend. However, there is an overall lower MAE when using the semi-long form data for smart floor gait calculations. By capturing more footfalls and gait segments on the smart floor, a clearer picture of a person's walking style appeared. The gait calculations appear to be closer to the "ground truth" data gathered by the high-resolution mat. Table 7-8 and Table 7-9 show the variances of the extracted parameters from the smart floor for the single pass and multiple pass scenarios. When calculating the variances over the full walking sequence across the entire smart floor, step and stride times varied the least and were the most consistent gait parameter of all the subjects. Step speeds, foot angles, foot lengths, step lengths, and stride lengths varied the most.

Smart Floor - Variances of 1st pass of Walking Sequence 5

Subject:	0	1	2	3	4	5	6	7	8	9	10
LEFT - Foot Length (cm)	0.025	0.047	49.136	0.108	38.173	4.969	17.643	1.727	8.277	4.427	0.642
RIGHT - Foot Length (cm)	0.006	3.856	0.032	2.742	29.931	3.220	2.642	1.074	13.504	0.035	2.637
LEFT - Foot Angle (degrees)	0.227	0.504	1.657	81.072	6.639	0.713	0.269	17.406	0.071	5.328	0.028
RIGHT - Foot Angle (degrees)	1.625	9.077	5.349	34.484	29.908	50.395	1.165	0.448	0.199	102.674	37.207
LEFT - Step Length (cm)	39.100	1.386	98.635	3.829	22.978	6.825	13.555	9.630	60.381	17.652	13.369
RIGHT - Step Length (cm)	5.215	9.049	2.992	5.794	2.515	8.280	0.007	3.599	0.000	0.000	0.179
LEFT - Stride Length (cm)	69.858	0.372	61.155	16.669	1.348	82.556	2.833	0.000	0.000	0.000	0.422
RIGHT - Stride Length (cm)	0.000	3.013	97.755	53.595	22.070	45.191	51.811	0.867	0.000	0.000	8.451
LEFT - Step Width (cm)	0.637	0.009	0.705	0.098	1.372	4.318	4.090	0.000	0.000	0.000	0.034
RIGHT - Step Width (cm)	0.000	2.120	0.054	0.279	3.753	1.998	0.019	0.080	0.000	0.000	0.104
LEFT - Step Time (secs)	0.006	0.000	0.002	0.002	0.000	0.000	0.006	0.000	0.000	0.002	0.000
RIGHT - Step Time (secs)	0.006	0.002	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.002
LEFT - Step Speed (cm/sec)	0.330	23.679	38.551	38.472	5.417	10.205	60.779	4.389	30.675	4.335	4.393
RIGHT - Step Speed (cm/sec)	0.022	2.957	7.796	8.910	0.756	7.334	33.520	313.981	0.000	0.000	6.509
LEFT - Stride Time (secs)	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.000	0.000	0.000	0.014
RIGHT - Stride Time (secs)	0.000	0.004	0.004	0.004	0.014	0.000	0.010	0.000	0.000	0.000	0.004
LEFT - Stride Speed (cm/sec)	0.108	1.128	1.613	3.480	3.535	23.956	48.285	0.000	0.000	0.000	4.301
RIGHT - Stride Speed (cm/sec)	0.000	11.753	0.415	0.020	17.940	6.839	19.848	122.659	0.000	0.000	10.869

Table 7-8: Smart Floor Gait Variances (single pass only)

Smart Floor - Variances of Entire Walking Sequence

Subject:	0	1	2	3	4	5	6	7	8	9	10
LEFT - Foot Length (cm)	18.906	17.066	24.879	20.923	9.693	4.857	8.725	13.684	7.703	2.805	20.095
RIGHT - Foot Length (cm)	12.889	12.379	12.144	4.476	25.933	6.245	12.545	7.590	2.748	2.605	23.833
LEFT - Foot Angle (degrees)	23.319	16.827	38.300	59.209	7.746	18.270	63.056	7.324	16.446	7.015	7.285
RIGHT - Foot Angle (degrees)	17.771	22.981	11.501	68.792	15.353	8.466	51.216	6.247	24.778	5.500	50.816
LEFT - Step Length (cm)	23.743	56.882	6.431	9.064	22.845	2.626	25.267	6.103	9.367	14.272	22.320
RIGHT - Step Length (cm)	5.939	13.791	3.579	8.318	12.966	2.491	13.258	19.055	18.279	6.024	19.870
LEFT - Stride Length (cm)	7.034	22.930	12.491	32.011	17.007	3.834	20.505	36.123	31.911	15.680	24.152
RIGHT - Stride Length (cm)	31.361	100.145	45.747	24.392	20.199	3.217	40.393	16.960	4.137	16.291	9.299
LEFT - Step Width (cm)	3.729	3.832	1.579	1.415	2.139	0.797	1.542	2.019	2.710	2.683	1.380
RIGHT - Step Width (cm)	2.578	1.338	1.799	1.909	0.528	1.787	2.088	1.321	2.794	0.746	1.624
LEFT - Step Time (secs)	0.002	0.001	0.000	0.001	0.001	0.000	0.001	0.001	0.001	0.001	0.001
RIGHT - Step Time (secs)	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.002	0.000	0.001
LEFT - Step Speed (cm/sec)	45.212	60.698	11.207	10.684	14.600	24.406	73.287	39.151	29.704	18.347	31.248
RIGHT - Step Speed (cm/sec)	22.064	79.699	20.909	29.781	10.081	8.038	25.517	22.980	39.731	29.868	47.820
LEFT - Stride Time (secs)	0.002	0.002	0.001	0.001	0.002	0.000	0.001	0.001	0.003	0.002	0.002
RIGHT - Stride Time (secs)	0.003	0.002	0.001	0.000	0.006	0.000	0.001	0.001	0.008	0.001	0.001
LEFT - Stride Speed (cm/sec)	6.248	7.192	4.369	1.981	8.099	13.656	21.860	30.894	30.090	7.033	25.172
RIGHT - Stride Speed (cm/sec)	7.540	68.825	7.586	1.850	8.060	4.403	35.146	9.250	26.149	2.401	8.245

Table 7-9: Smart Floor Gait Variances (entire floor)

7.4 Preliminary Person Identification

Another experiment was performed to evaluate the utility of the extracted gait parameters and their ability to distinguish different gait patterns. In particular, a person identification classifier was trained using only the extracted gait parameters from the smart floor data. For this, *k*-nearest neighbors (*k*-NN) classification was applied to the smart floor's gait parameter calculations from the subjects' data in the walking study. This preliminary effort of person identification based solely off gait parameters used standard *k*-NN techniques without special weighting schemes.

Each of the 10 unique individuals generated 4 to 13 valid gait segments during their full walking sequence across the entire floor. A total of 70 valid gait segments across all subjects were generated. A valid gait segment contains at least four footfalls after gait segmentation due to turns and leaving the floor. The minimum requirement of four footfalls ensures that every gait parameter has enough data for a calculation (i.e. left and right strides). The feature space for each sample is the 20 gait parameter means from each valid gait segment.

Note that previously established subject 0 is the same person as subject 8, and the two datasets have been combined and labeled solely as 8. The *k*-NN class labels are the 10 subject numbers (IDs).

10 Subjects	70 Total Gait Segments
1	9
2	6
3	6
4	4
5	6
6	8
7	6
8	13
9	6
10	6

Table 7-10: *k*-NN Data Samples and Labels

Figure 7.4 shows the 20-dimensional feature space of each gait segment sample reduced to 2 dimensions using t-distributed stochastic neighbor embedding (t-SNE). It is purely for visualizing the high-dimensional data and is not used in the *k*-NN algorithm. A perplexity of 5 was used for the t-SNE plot.

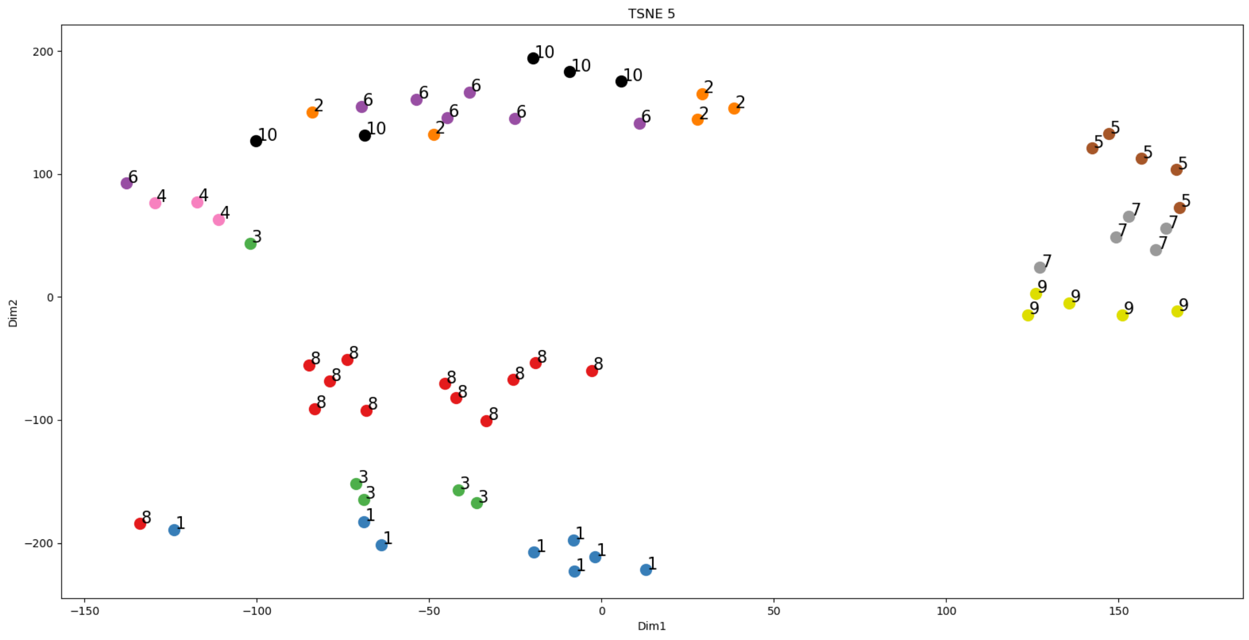


Figure 7.4: t-SNE Plot of *k*-NN Training Samples with Subject ID Labels

For each subject, 1 of their gait segments was randomly selected as holdout test data and the remaining were used as the larger training set resulting in a training sample size of 60 gait segments with class labels (subject IDs) and a test set of 10. The classification was run 1000 times for each *k* ranging from 1 to 29 to generate an average classification accuracy on the holdout test data. The score metric is the mean accuracy of all 10 test data samples with 1.0 being the best and 0.0 the worst. The entire training and testing runs were performed 3 times using different data scaling methods: raw features, min-max scaling 0 to 1, and standardization.

With a training sample size of 60 with 10 different class labels a maximum accuracy of 0.87 was achieved. The optimal *k*-neighbors value was 5 across all feature scaling methods. Table 7-11 below shows the accuracy results omitting *k* values above 15 since accuracy only decreased as *k* grew larger as illustrated in Figure 7.5.

k	Raw Data	MinMax Scaler	Std Scaler
1	0.81	0.77	0.75
2	0.76	0.80	0.75
3	0.78	0.84	0.79
4	0.81	0.84	0.83
5	**0.84**	**0.87**	**0.85**
6	0.83	0.80	0.79
7	0.78	0.77	0.77
8	0.70	0.75	0.77
9	0.64	0.70	0.71
10	0.64	0.68	0.65
11	0.61	0.63	0.63
12	0.60	0.59	0.60
13	0.55	0.58	0.57
14	0.50	0.56	0.56
15	0.47	0.53	0.55

Table 7-11: *k*-NN Classification Results

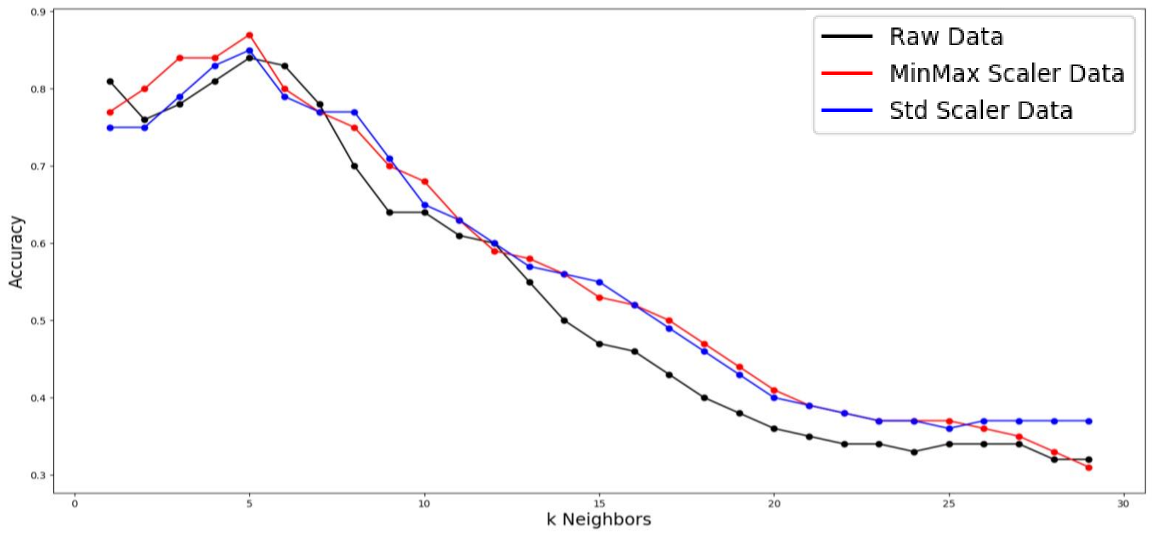


Figure 7.5: *k*-NN Classification Results Graph

8 Conclusions and Future Work

8.1 Summary

This dissertation has developed methods for extracting meaningful walking data from a custom-built smart floor using automatic calibration, machine learning, and clustering techniques. The following sections review the contributions of this research and propose future work and applications.

8.2 Automatic Floor Sensor Calibration

Having a specialized automatic method for calibrating the smart floor over time is critical when the sensors are no longer directly and independently accessible after being deployed under the tile surface. The calibration algorithm described in Chapter 4 allows sensors to be jointly calibrated and converted to kg units while accounting for static tile weight. By learning how each sensor behaves when experiencing no human activity, the algorithm also accounts for and removes any source of static weight which is useful for a smart floor within a person's home containing furniture and other household objects.

The calibration process does not require any specialized hardware or obtrusive techniques, merely the known weight of the person/resident is required and different sensors can be targeted and calibrated over time by simply walking over the smart floor in a normal fashion.

8.3 Contact Point Extraction using Machine Learning Techniques

To overcome the low 1-sqft resolution of the smart floor and capture meaningful single contact points of a person's footfalls, a three-step process of building and training neural network models is detailed in Chapter 5. Using a specific training set of known single contact point location and weight labels paired with calibrated sensor values, a mapping and pressure profile is learned that can be applied over the entire smart floor.

Using this learned model of how a tile subsection responds to a single contact point of weight, it can be used with generic walking data in a larger convolutional model that learns the larger non-linear relationships of how the entire floor disperses and responds to weight. The convolutional model, during training, learns how to account for physical differences in the tile coupling, sensorless rubber edges, static weight, and other inconsistencies.

The encoder portion of this model can be extracted and used to predict a location and weight value for every tile of the smart floor for every time sample. These predicted values are used as single contact points of a person's walking pattern over time. They can be used for tracking, clustering, identification, deviation detection, and gait analysis.

8.4 Footfall Clustering and Gait Analysis

The extracted contact point features of time, location, and weight are used in a recursive Hierarchical Clustering Analysis algorithm explained in Chapter 6 that determines which points likely belong to which footfall. The unique recursive element of the method ensures that any incorrectly grouped footfalls will be further split again and again until only single foot images remain. The algorithm also automatically identifies and removes turning data so only semi-straight reliable moments of a person's walking sequence are used to generate gait measurements.

Once the set of footfalls is properly segmented, gait analysis can be performed to measure how well a person is walking on the smart floor. A total of 20 gait parameter measurements can currently be calculated on the smart floor walking data described in Section 6.2. The gait results were comparable to those captured by a high-resolution gait mat. The smart floor's gait analysis can be performed on normal everyday in-home walking data of a person living within the SmartCare apartment without the need for expensive high-resolution mats or the resident having to visit a dedicated clinic and perform a specific routine.

In the future, this clustering algorithm should account of multiple people walking on the floor simultaneously and near one another. Possible methods to explore would involve tracking using a Kalman Filter and Dynamic Time Warping to learn a model or profile of an individual's footfalls.

8.5 Adapt Models for Apartment Floor Data

All smart floor methods described in this dissertation were constructed using a smaller version of the floor within a laboratory setting. A critical next step is to adapt and apply the calibration methods, models, and clustering techniques for application with the SmartCare apartment's larger smart floor. The general principles of these methods hold true for the apartment floor, but special consideration would have to be taken into account for the larger number of sensors, different spatial arrangement, and gaps of the apartment floor.

Semi-long form data from the apartment has already been collected from the volunteers living in the apartment for different periods of time. The volunteers' walking data is an abundant source of new training data that will be used in calibration, model training, and clustering for the apartment smart floor.

8.6 Incorporate Other Sensor Data from Apartment

The smart floor is not the only source of data collection within the SmartCare apartment. Embedded sensing technologies also collect electricity usage, water usage, IR locations, temperature, humidity, luminosity, and drawer/cabinet activity. These features could be used separately or together with floor data to learn a resident's daily activity and identify deviations and irregularities. They could also be used to help label and separate different people to aid in tracking if multiple people are living in the apartment.

9 References

[1] A. S. Alharthi, S. U. Ozanyan and K. B. Yunas, "Deep Learning for Monitoring of Human Gait: A Review," *IEEE Sensors Journal,* vol. 19, no. 21, pp. 9575-9591, 2019.

[2] "AgingMO - Tiger Place," [Online]. Available: https://agingmo.com/tiger-place-institute/.

[3] "CASAS," [Online]. Available: http://casas.wsu.edu/about.

[4] "Tiger Place Research Projects," [Online]. Available: https://agingmo.com/research-projects/aging-in-place/tigerplace/.

[5] "CASAS Research Projects," [Online]. Available: http://casas.wsu.edu/publications/.

[6] G. Qian, J. Zhan and A. Kidané, "People Identification Using Floor Pressure Sensing and Analysis," *IEEE Sensors Journal,* vol. 10, no. 9, pp. 1447-1460, 2010.

[7] O. Costilla-Reyes, R. Vera-Rodriguez, P. Scully and K. B. Ozanyan, "Analysis of Spatio-Temporal Representations for Robust Footstep Recognition with Deep Residual Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 41, no. 2, pp. 285-296, 2019.

[8] R. Vera-Rodriguez, J. S. D. Mason, J. Fierrez and J. Ortega-Garcia, "Comparative Analysis and Fusion of Spatiotemporal Information for Footstep Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 4, pp. 823-834, 2013.

[9] A. Bränzel, C. Holz, D. Hoffmann, D. Schmidt, M. Knaust, P. Lühne and P. Baudisch, "GravitySpace: tracking users and their poses in a smart room using a pressure-sensing floor," in *SIGCHI Conference on Human Factors in Computing Systems,* 2013.

[10] A. Steinhage, C. Lauterbach and A. Techmer, "2-Large-Area Wireless Sensor System for Ambient Assisted Living," in *SENSOR 2013,* 2013.

[11] P. Srinivasan, D. Birchfield, G. Qian and A. Kidane, "Design of a pressure sensitive floor for multimodal sensing," in *IEEE - Information Visualization*, 2005.

[12] J. Suutala and J. Röning, "Towards the adaptive identification of walkers: Automated feature selection of footsteps using distinction sensitive LVQ," in *Workshop on Processing Sensory Information for Proactive Systems,* 2004.

[13] J. Suutala and J. Röning, "Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option," *Information Fusion,* no. 9, pp. 21-40, 2008.

[14] S. Reddy, "Person Identification and Anomaly Detection using Gait Parameters Extracted from Time Series Data," The University of Texas at Arlington, Arlington, 2017.

[15] O. Oluwadare, "Gait Analysis on a Smart Floor for Health Monitoring," The University of Texas at Arlington, Arlington, 2015.

[16] A. Alamdari and V. Krovi, "A Review of Computational Musculoskeletal Analysis of Human Lower Extremities," *Human Modelling for Bio-Inspired Robotics,* 2017.

[17] J. Perry and J. M. Burnfield, Gait Analysis: Normal and Pathological Function, New York: Slack Inc, 2010.

[18] "The Gait Cycle: Phases, Parameters to Evaluate & Technology," Tekscan, Inc., [Online]. Available: https://www.tekscan.com/blog/medical/gait-cycle-phases-parameters-evaluate-technology. [Accessed 2020].

[19] S. Raschka and V. Mirjalili, Python Machine Learning - Second Edition: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow, Birmingham: Packt Publishing, 2017.

[20] scikit-learn, "Agglomerative Clustering Using sklearn," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html. [Accessed 2020].

[21] G. J. Szekely and M. L. Rizzo, "Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method," *Journal of Classification,* no. 22, pp. 151-183, 2005.

[22] A. Burkov, The Hundred-Page Machine Learning Book, Quebec City: Andriy Burkov, 2019.

[23] G. Záruba, M. Huber, K. Daniel and N. B. Burns, "SmartCare - An Introduction," in *PerCom*, 2017.

[24] K. Daniel, M. Huber, G. Záruba, N. B. Burns and P. Sassaman, "PESTO: Data Integration for Visualization and Device Control in the SmartCare Project," in *PerCom*, 2016.

[25] "FlexiForce Load/Force Sensors and Systems," Tekscan, [Online]. Available: https://www.tekscan.com/flexiforce-load-force-sensors-and-systems. [Accessed 2020].

[26] "BeagleBone Black," Texas Instruments, [Online]. Available: https://beagleboard.org/black. [Accessed 2020].

[27] scikit-learn, "Gaussian Mixture Models," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html. [Accessed 2020].

[28] K. P. Burnham and D. R. Anderson, Model Selection and Multimodel Inference: A practical information-theoretic approach, Springer-Verlag.

[29] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, Sebastopol: O'Reilly Media, 2017.

[30] Chollet, Francois and Others, "Keras," 2015. [Online]. Available: https://keras.io/.

[31] "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: http://tensorflow.org/.

[32] M. I. Jordan and D. E. Rumelhart, "Forward Models: Supervised Learning with a Distal Teacher," *Cognitive Science*, no. 16, pp. 307-354, 1992.

[33] "ProtoKinetics," ProtoKinetics, LLC, [Online]. Available: https://www.protokinetics.com/.

[34] *PKMAS: ProtoKinetics Motion Analysis Software User Guide,* Peekskill, NY, USA: Zenometrics, LLC, 2012.