An Ambulatory Monitoring Algorithm to Unify Diverse E-Textile Garments

by

Madison T. Blake

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Thomas L. Martin, Co-Chair

Mark T. Jones, Co-Chair

Cameron D. Patterson

February 12, 2014

Blacksburg, Virginia

Keywords: Activity Classification, Wearable Computing, User-independence

An Ambulatory Monitoring Algorithm to Unify Diverse E-Textile Garments

Madison T Blake

(ABSTRACT)

In this thesis, an activity classification algorithm is developed to support a human ambulatory monitoring system. This algorithm, to be deployed on an e-textile garment, represents the enabling step in creating a wide range of garments that can use the same classifier without having to re-train for different sensor types. This flexible operation is made possible by basing the classifier on an abstract model of the human body that is the same across all sensor types and subject bodies. In order to support low power devices inherent for wearable systems, the algorithm utilizes regular expressions along with a tree search during classification.

To validate the approach, a user study was conducted using video motion capture to record subjects performing a variety of activities. The subjects were randomly placed into two groups, one used to generate the activities known by the classifier and another to be used as observation to the classifier. These two sets were used to gain insight on the performance of the algorithm. The results of the study demonstrate that the algorithm can successfully classify observations, so as long as precautions are taken to prevent the activities known by the classifier to become too large. It is also shown that the tree search performed by the classification can be utilized to partially classify observations that would otherwise be rejected by the classifier. The user study additionally included subjects that performed activities purely used for observations to the classifier. With this set of recordings, it was demonstrated that the classifier does not over-fit and is capable of halting the classification of an observation.

Acknowledgments

I am indebted to many people for the completion of this work:

To the participants of the user study who endured performing many repetitive tasks with little quarrel.

To the CCM Lab for the many games of foosball games and Friday lunches.

To my Co-Advisor Dr. Martin for his support and refusal of the words “i’ll try”.

To my Co-Advisor Dr. Jones for the many long meetings about various forms of abstract math.

This material is based in part upon work supported by the National Science Foundation under Grant Number IIS-1116669. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

This thesis presents a user and sensor independent automatic activity classification scheme. The goal is to develop classifier that can be deployed with a set of garments for ambulatory medical monitoring without requiring re-training for each sensor and garment type. This approach is motivated by the need for ambulatory monitoring garments that can work across a wide range of patients without training on an individual and for developing a set of garments for a variety of medical monitoring applications without having to create a new classifier for each new type of garment.

Because it is not feasible for a doctor to fully assess a patient's health during a short visit to a clinic, and because many health events happen outside the clinic [1], ambulatory monitoring has emerged as an attractive and economical solution to remotely monitor patients with minimal or no distraction from the patient's daily living. The envisioned ambulatory monitoring solution is comprised of a set of wearable sensors which are used to monitor the

patient, and an activity classifier used to recognize the patient's physical activity. Activities for which such a system could be used include automatic creation of a patient's activity diary while wearing a Holter Monitor [2] and recognition of a patient's physical therapy routine [3]. In addition to medical applications, an activity classifier can be used in a variety of applications such as gesture recognition [4] and factory assembly lines [5].

This work is the enabling step in developing a broad range of ambulatory monitoring garments, each of which may be targeted toward a different form of ambulatory monitoring and physically implemented using different types of sensors. For such a system to be feasible, four desirable traits have been identified: sensor-independent, user-independent, scalable, and simple.

Sensor-independence characterizes the system's ability to operate on a range of garments, allowing the knowledge of activities to be shared across a wide range of possible sensor types. **User-independence** characterizes the system's ability to operate on a range of different people, allowing it to be given to the patient without having to be trained or configured for that specific patient. **Scalability** characterizes the system's ability to operate across a range of activities, allowing it to operate efficiently even as the number of activities to be classified grows. Finally, **simplicity** characterizes the system's computational performance requirements, allowing it to run on the low-power devices on which this work is targeted to operate. The work presented in this thesis describes an algorithm designed to take into account these four properties in order to construct an activity classification system suitable for an ambulatory monitoring garment.

1.2 Contributions

This thesis describes the Model-based User-independent Activity Classifier (MUAC), a classifier that utilizes a high-level model of the body, is user-independent, and is designed for low-power operation, all while retaining accurate classification on similar activities. The MUAC is based upon a common intermediate body model which describes the orientation of the subject's body and limbs. A recording is compared to a set of known activities to determine what, if any, activities it classifies to. These activities are derived from multiple users and are independent of the particular sensor platform or user.

This thesis makes the following contributions:

1. An algorithm to create a user-independent, sensor-independent activity representation that can be used to match an observation to an activity. To achieve sensor independence a body model is used as the mathematical space that MUAC operates on, as discussed in Chapter 3. This is in contrast to many previous wearable systems, as discussed in Chapter 2, where the classifier operates on sensor values. The method is carefully designed such that it can handle a large number of dimensions, in this work nine dimensions are used by the model. Chapter 4 presents the mathematical operations performed to construct the activity representation and the regular expression used to compute the match to an observed sequence. The use of a string matching approach is chosen because of its simplicity in matching two sequences, as shown in other works discussed in Chapter 2.

2. An algorithm to structure the set of activities into a hierarchy based on the order of similarity. Such a hierarchy allows for a tree search to be performed instead of a linear search. The algorithm to construct the hierarchical tree is given in Chapter 4.

3. A MATLAB implementation of the algorithm used to create the representation of an activity, and a C implementation of the string matching algorithm. The correctness of these implementations is shown in Chapter 5.

4. A testing framework which accepts either data from a video motion capture system[6] or the Microsoft Kinect[7] to generate body model sequences. This allows the use of publicly available motion capture data as well as the ability to perform user studies for specific activities. This framework is detailed in Chapter 5.

1.3 Thesis Organization

The remainder of this thesis is outlined as follows. Chapter 2 gives background and related work. Chapter 3 describes the model used to represent the human body and mathematical operations on the model. Chapter 4 presents the algorithms to create each activity's representation, construct the hierarchical tree, and generate an activity from a set of examples. Chapter 5 discusses experiments performed to test the quality of the contributions outlined above. Chapter 6 gives results and discussion of the experiments performed. Finally Chapter 7 discusses conclusions and provides directions for future work.

Chapter 2

Background

This chapter presents previous work which the MUAC relies on or improves upon. Section 2.1 describes user-independence and contrasts some classification schemes to achieve user-independence. Section 2.2 describes approximate string matching, an important concept used by this thesis. Finally Section 2.3 contrasts the MUAC to many existing wearable classification approaches.

2.1 User Independence

User-independence is an important characteristic of many classifiers because it often desired to give the classifier to users which it was not specifically trained for. User-independence is achieved when the classifier is able to classify the variations of the population performing the same activity [8]. Variations across the population has two causes:

1. **Anthropometric Variations** are differences between each subject's movement through three dimensional space. These differences may be due to a difference in body geom-

etry, or by a specific body part that is non-essential to the activity. For example, it has been shown that a subject's arms can cause error in classifying the activity of walking [9].

2. **Performance Speed Variations** are differences between each subject's movement through time. Subjects performing an activity will not only perform each activity somewhat differently, but will also perform the activity at different speeds [8].

A method that is often used to account for performance speed variations is a technique called dynamic time warping [10][11]. Dynamic time warping (DTW) takes two sequences of length m and n and finds the path through an $m \times n$ matrix, which results in a nonlinear mapping of one sequence onto the other [12]. With DTW, the number of adjacent steps allowed is restricted, which ultimately allows for variations in execution speeds. The MUAC uses a specific form of DTW that does not allow insertions to perform approximate string matching. Insertions are not needed because instead of relying on performing DTW on each exemplar in the training data, MUAC constructs a single regular expression to an observation to.

A common approach to detect variations is to give the classifier multiple examples of subjects performing the same activity. These multiple examples make up the activity's example set, also called training set, which is used to create a user-independent classifier. Ideally the example set will contain every possible variation for that activity; realistically, the example set is constructed such that there can be some confidence that it contains a sufficient amount of variations. The MUAC uses such an example set to find variations in an activity, which it uses to create a user-independent representation of that activity.

What makes the MUAC unique is *how* it overcomes the variations in an activity. Many activity classifiers use a large training set that contains many examples with some metric to compute a statistic of similarity between an observation and each item in the example

set [11][8][13]. To match the observation to an activity, the observation is compared to *all* the examples of the activity in the training set. The MUAC instead uses a single representation of the activity, as opposed to others because it is computationally intensive and is at odds with the goal of efficient operation.

One proposed camera-based approach takes into account both forms of variations by using a metric similar to DTW, and a matrix of weights to represent anthropometric variations [4]. This approach also uses a similar concept of a model to express the human body and aims to classify similar activities. The approach uses a training set, examples of an activity being performed, and a heuristic metric of similarity, similar to Dynamic Time Warping, to determine similarity within the example set. Mahalanobis distance is used to measure the distance between an observed sequence and a single "centroid" sequence, where the centroid sequence is chosen as the training set instance most similar to all the others. In this approach, the anthropometric differences are resolved using the Mahalanobis distance and performance speed variations are resolved using the similarity measurement. The MUAC uses a metric related to the Mahalanobis distance; however, the MUAC uses all examples to compute the "centroid" activity, instead of relying on a sufficiently "central" example being present in the example set. Additionally, instead of using statistical variances to compute the variation parameters, the MUAC uses a minimal bounding circle algorithm. As shown in Chapter 5, this difference results in a more predictable distance metric because it is not affected by a repetitively occurring aspect of an activity.

2.2 Approximate String Matching

One major concern for low-power operation of activity classifiers is the computational complexity required to match two sequences of data. String matching has been shown to be an

approach that can efficiently line up sequences of data, and has been used with success in computational biology, signal processing, and even wearable computing [14][5]. Some key definitions used by string matching are the following:

- $\boldsymbol{\Sigma}$- A finite alphabet of characters.
- $\boldsymbol{P} \in \boldsymbol{\Sigma}^*$- A set of characters that represents the pattern to match to.
- $\boldsymbol{T} \in \boldsymbol{\Sigma}^*$- A text to match to the pattern.
- ε- An empty string.
- $\boldsymbol{d} : \boldsymbol{\Sigma}^* \times \boldsymbol{\Sigma}^* \rightarrow \mathbb{R}$- A distance function between two characters in the alphabet.

Utilizing regular expressions, performance speed variations can be compensated for by formally defining how many times a specific character in a given text is allowed to match a character in the pattern [15]. This approach is used by MUAC during real time operation to efficiently compute a match while also taking into account performance speed variations.

2.3 Wearable Systems

There have been many systems that aim to do activity classification on a wearable device similar to the target device of this thesis. One class of wearable activity classifiers employs a single inertial sensor located on the trunk of the user. These techniques use a variety of classifier algorithms and have been shown to be effective at classifying general locomotive motions including walking, jogging, bicycling on a stationary bike, and sitting [16][17]. Such an approach, however, cannot distinguish the very similar everyday activities such as Brushing Teeth and Eating because a single inertial sensor on the trunk will not be able to give any accurate information on a limb's position.

One approach uses an inertial sensor held by the subject to perform one hand gesture recognition such as drawing shapes or playing card movements [18]. This work benefited from knowing that the motion sensed by the system is important to the gesture. This benefit does not apply to the activities used in this thesis. Some activities contain movement that is not important for that activity, for example the arms while taking a step. The MUAC is designed to recognize unimportant dimensions and ignore them while classifying observations.

It is known that a garment moves relative to the skin [19] and it has been shown that movement of sensors can negatively affect the accuracy of a classifier [20]. When using the MUAC, the component that generates the body model from the sensed values is responsible for compensating for any sensor drift. The MUAC operates under the assumption that if two subjects are in the same body position with sensors drifted into different positions, the same body model will still be generated.

The relationship between sensor values and body model distinguishes the MUAC from many approaches that use inertial measurement sensors placed around the human body. In these approaches, the recorded sensor data of a subject performing an activity is used to train the system for that activity, using a particular classification algorithm. Algorithms such as Bayesian classifiers [21], decision tree [22], support vector machines [23], string matching [5], and Dynamic Time Warping [11] have been used to achieve a similar accuracy performance to the work in this thesis. In each of these approaches, it is required that the same sensors that were used to collect training data are used to observe the subject. The overarching difference between each of these approaches is the MUAC's use of a body model to represent the human body. As to be described in Chapter 3, each sensor type translates its sensor values into an abstract model of the human body. The body model relieves the need to re-collect data and re-train the classifier when the sensor types are changed. This approach is different than translating from one sensor type to another sensor type as previous work

has done between inertial measurement units and Kinect points [24]. As long as each sensor type is able to create the same body model given the same orientation of the human body, the sensor types can be used interchangeably by the MUAC.

Chapter 3

Model

This chapter discusses the model used by this thesis to represent the human body and introduces the basic mathematical operations required by the MUAC. Overall, a model of the human body based upon the rotation of segments is used to represent the torso, arms, and legs of a person. Section 3.1 discusses segments, dimensions of lengths used to represent human limbs. Section 3.2 discusses the rotation of segments and how it is expressed. Finally, Section 3.3 combines the segments and rotations to introduce the mathematical space which this thesis uses to describe the human body.

3.1 Segments

In this thesis, an eight-segment model of the human body, the left/right upper arm, left/right forearm, left/right thigh, and left/right shin. Each segment extends from a joint which it rotates about. The segment labels and originating joints are given in Table 3.1. Additionally, each segment has a length based on the normalized human body; these lengths are discussed in Appendix A, and do not effect the operation of MUAC.

Label	Description	Originating Joint
LUA	Left Upper Arm	Left Shoulder
LFA	Left ForeArm	Left Upper Arm
RUA	Right Upper Arm	Right Shoulder
RFA	Right ForeArm	Right Upper Arm
LT	Left Thigh	Left Hip
LS	Left Shin	Left Thigh
RT	Right Thigh	Right Hip
RS	Right Shin	Right Thigh

Table 3.1: Segments used to express the human body

3.2 Rotation

The upper segments' (LUA, RUA, LT, RT) coordinate system is defined by the x axis protruding out of the shoulder/hip, the z axis pointing toward the head, and the y axis pointing the direction the torso is facing. Each segment on the model is put into position by applying a rotation onto the vector [1 0 0] extending out of the joint. Therefore, the fore segments' (LFA, RFA, LS, RS) coordinate system is the upper segments' coordinate system rotated by the upper segments' rotation. The fore segment's position is dependent on the rotation of the upper segment that it is connected to. If a subject were to extend both their left upper arm and left fore arm straight out in line with the shoulders, both the left upper arm and left forearm would have the same rotation. If the subject were to raise their arm, keeping the forearm fully extended, only the left upper arm's rotation would be changed, while the forearm's rotation will remain unchanged.

The segment's coordinate system is designed such that rotation is defined relative to what the segment is attached to. If the upper segment's rotation is changing while the fore segment remains still, with respect to the upper segment, the fore segment's rotation will not be changing. Additionally, in this approach, a vertically symmetric body will have identical rotations for both the left and right side.

The rotation of a segment is represented by a quaternion, a hypercomplex number system

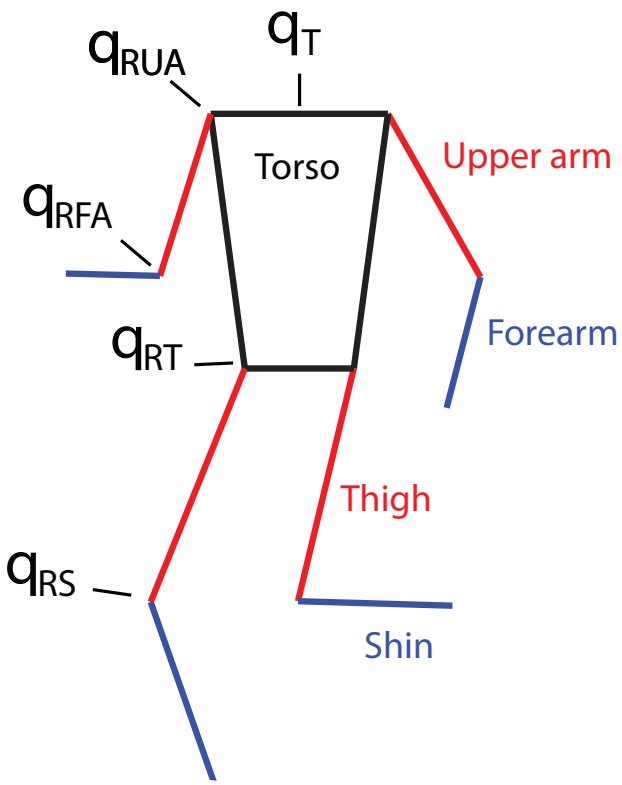

Figure 3.1: A nine-segment body model, with the segments labeled on one side and quaternions for the joints shown on the other. This body model is used for the experimental results in this thesis, but the method is not limited to a specific model.

in $\mathbb{H}$ space. Quaternions are useful because each quaternion provides a unique rotation to a point in space, they are more efficient than other rotation representation schemes [25], and it is simple to interpolate between two quaternions. Appendix B provides a detailed discussion on how to convert a segment's vector into a quaternion and vice versa.

3.3 Pose

Segments and rotations are used together to define the **pose**, an instantaneous measurement in the particular way which a human subject orients its body and limbs. The pose is constructed using a set of nine quaternions; one quaternion expressing the rotation of the torso in the environment, and eight quaternions expressing the rotation of each segment. The pose, $\mathbf{p} \in \mathbb{H}^9$, is written as

$$\mathbf{p} = [q_T \; q_{LUA} \; q_{LFA} \; q_{RUA} \; q_{RFA} \; q_{LT} \; q_{LS} \; q_{RT} \; q_{RS}] \, . \tag{3.1}$$

Figure 3.1 shows graphically how the segments and quaternions together construct the pose. The torso quaternion, q_T, is a rotation that is applied to all other quaternions in the pose. It is used to rotate the coordinate system of body onto the coordinate system of the environment. If two poses were only different in how the subject positioned their body in the environment then it would be expected that all other quaternions in the pose would be identical. Refer to Appendix C for a more detailed discussion of how to determine the subject's spatial orientation from a pose.

The distance between two quaternion in a pose, q_1 and q_2, is expressed as the function

$$\theta : [q_1, \; q_2] \rightarrow \mathbb{R}, \tag{3.2}$$

and defined as

$$\theta \, (q_1, \; q_2) = \sqrt{\overline{(q_1 - q_2)}(q_1 - q_2)}. \tag{3.3}$$

A size N vector of distances between two poses, $\mathbf{p_1}$ and $\mathbf{p_2}$, is computed by the function

$$\mathbf{d} : [\mathbf{p_1}, \mathbf{p_2}] \rightarrow \mathbb{R}^N. \tag{3.4}$$

Element i of $\mathbf{d}$ is computed by

$$d_i\left(\mathbf{p}_1, \mathbf{p}_2\right) = \theta\left(p_{1_i}, p_{2_i}\right) \tag{3.5}$$

where p_{1_i} is the i^{th} quaternion of the p_1 pose.

The use of the pose as the basis for classification is pivotal towards the sensor-independence characteristic of MUAC. Instead of using raw sensor data that could be different across sensor types, the raw data is converted to the pose representation introduced in this section. Since the MUAC operates purely on poses rather than raw sensor values, two distinct sensor types can be used interchangeability; observations from one sensor type can be matched to the activity representations created from another sensor type. The use as a pose also allows for the MUAC to be user-independent. Each subject is fitted to the same normalize body model, regardless of body size, shape, or sex. Since each subject is fitted to the same pose, their pose sequences can also be used interchangeably among other subjects.

Chapter 4

Method

This chapter outlines MUAC's methodology to generate the user-independent representation of activities that is used to compute an observation's match in both a scalable and simple manor. Section 4.1 introduces how to compute and use the activity's **archetype**, the user-independent representation of the activity built from performances of the activity. Section 4.2 introduces a technique to reduce complexity of matching an observation by applying a tree search among a set of already constructed activities. Finally, section 4.3 discusses how to analyze the importance of each recording used to construct the archetype, and how to remove unnecessary recordings from the set.

4.1 Algorithm Element: The Activity's Archetype

The first element of the MUAC addresses the creation and use of the activity archetype. The archetype is created from an **example set**, a set of pose sequences which are given to represent variations of performing the activity. The example set is analogous to the training set used by approaches in Section 2. The pose sequences can be generated from any source,

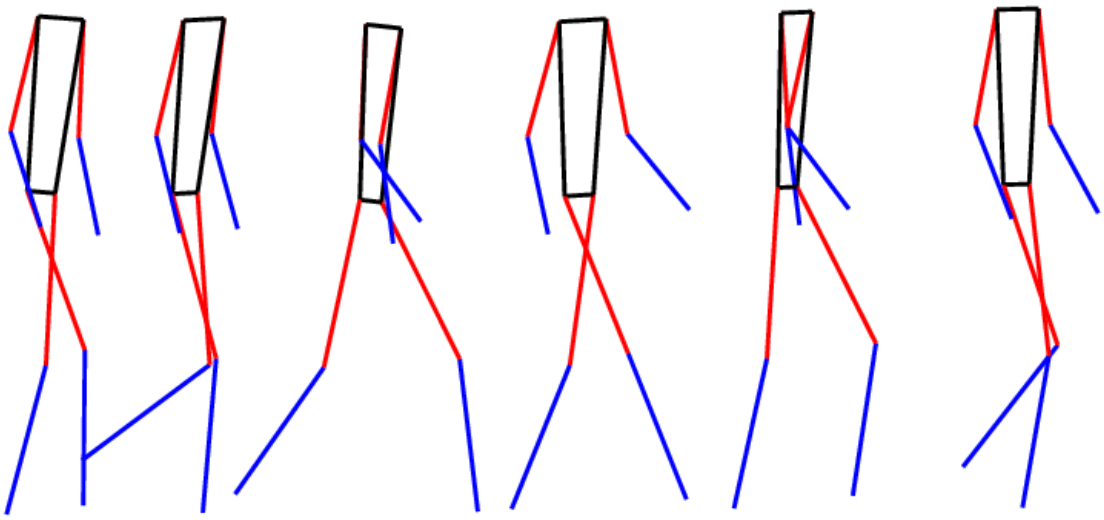

Figure 4.1: A sequence of segmented body model poses in which the subject is walking. Generated by video motion capture data publicly available via CMU Graphics Laboratory: `http://mocap.cs.cmu.edu`, 2012.

such as a video motion capture system that is capable of producing the spatial coordinates of many locations on a subject's body for every frame of video captured (see Figure 4.1). This section is only concerned with how to use a given example set to construct the archetype. Refer to Section 4.3 for a metric of the impact each example makes on the resulting archetype and Section 5.1.3 for how an example set can be analyzed. The algorithm discussed here can be used with any mathematical model; as such, this chapter will refer to a general N dimensional model instead of the specific 9 dimensional model introduced in Chapter 3.

Throughout this section examples will be given of each component using two activities, LeftStep and ShoulderCircle, which are used in Chapter 5 for experimental analysis. LeftStep is an activity where the subject was asked to take a single step forward with the left foot. For the ShoulderCircle activity the subject was asked to stand with arms extended and rotate both arms around the shoulder together. From the subject's point of view, the left

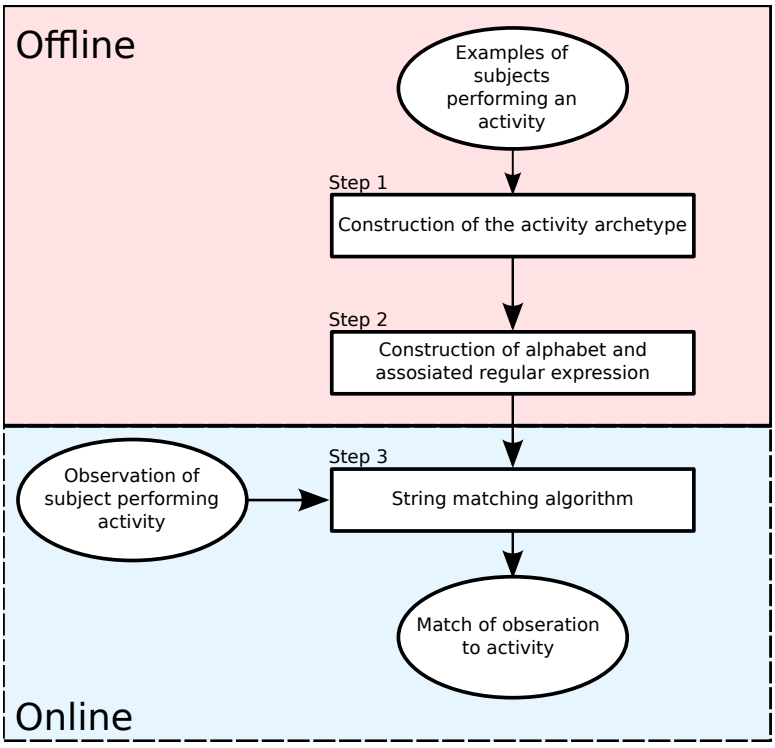

Figure 4.2: An outline of steps to create an use an activity archetype to classify an observation in a real time environment.

arm rotates clockwise, and the right arm counterclockwise. Chapter 5 gives more details on what these activities are and how they were recorded. These two activities were chosen to demonstrate components because they are significantly different and can illustrate the flexibility that the MUAC offers.

An outline of creating and using an activity's representation from an example set is shown in Figure 4.2. In the figure, the **offline** steps, any computations that can be performed prior to system deployment, are marked in red and surrounded by a solid box. The **online** steps, any computations that must be performed real time, are marked in blue and surrounded by a dashed box. Discussion of each step outlined in Figure 4.2 follows.

4.1.1 Computing Activity Archetype (Step 1)

The first step discussed in Figure 4.2 is the creation of the activity archetype. This step is computationally intensive and is to be performed offline, prior to system deployment. The archetypical activity is computed using four components outlined in Figure 4.3, with details on each component following.

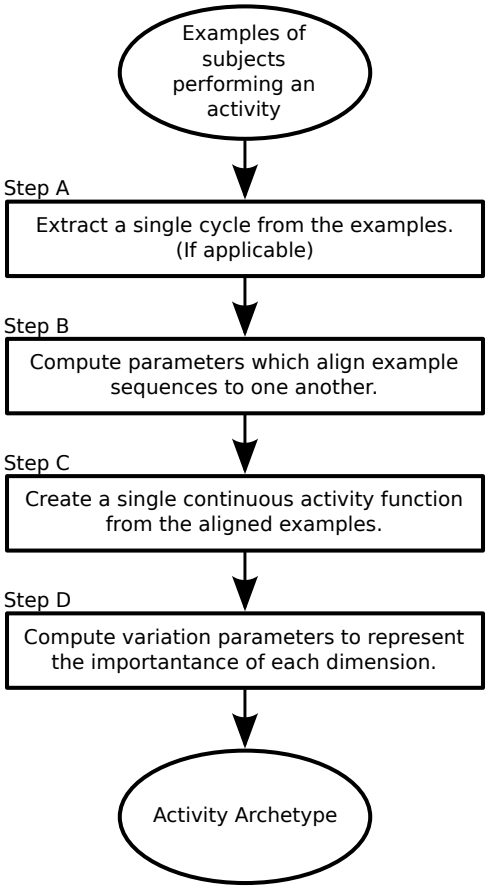

Figure 4.3: An outline of components required to create an activity archetype from an example set.

Each of the components requires an example set containing sequences of the activity being performed. The example set, $\mathbf{e}$, is a set of M examples where the j^{th} example is a continuous

function in quaternion space defined as

$$\mathbf{e}_j : t \to \mathbb{H}^N, \tag{4.1}$$

and contains a distance metric defined as

$$\delta\left(\mathbf{p}_1, \mathbf{p}_2\right) = \sum \mathbf{d}\left(\mathbf{p}_1, \mathbf{p}_2\right). \tag{4.2}$$

Cycle Extraction (Step A)

Some activities that the MUAC may be asked to identify can be **cyclic**, meaning that the same movement is repeated multiple times. Walking is an example of a cyclic activity. Given an example set of a cyclic activity, which has at least two cycles, a single cycle is first extracted from the example.[1] The single cycle is then used as the example set in the rest of the steps to construct the archetypical activity. If the activity is not cyclic then this step is not performed.

Given that e_j has a length of $|e_j|$ and belongs to a cyclic activity, a single cycle is extracted by computing the point, x, which

$$x = \min_{t \in [0,\ 0.5 \cdot |e_j|]} \int_0^t Var\left(\left[\mathbf{e}_j\left(x + 0t\right),\ \mathbf{e}_j\left(x + 1t\right),\ \ldots,\ \mathbf{e}_j\left(x + \left\lfloor \frac{L}{t} \right\rfloor t\right)\right]\right) dx. \tag{4.3}$$

For this thesis, the minimization is computed utilizing MATLAB's `fmincon` function while the integral is computed using the `quad` function.

Because Equation 4.3 is based on the variance between two or more periods, it is required

[1] Detecting *if* an activity is cyclic or not is not required, cyclic activities are labeled manually for the MUAC.

that the recording contain at least two cycles. With the computed x, the example is replaced by a single cycle of the example

$$\mathbf{e}_j = \mathbf{e}_j(t) : 0 \leq t \leq x. \tag{4.4}$$

The cycle extraction algorithm is demonstrated in Figure 4.4 on a single recording containing three full cycles of the ShoulderCircle activity. The three full cycles and one partial cycle contained on the recording are plotted such that each cycle starts at the same point. In the ShoulderCircle activity, the arms are moving in a circle while the rest of the body stays relatively still. In Figure 4.4, the second and fourth dimension (left and right upper arm) display significant cyclic behavior which the algorithm used to determine the length of single cycle.

Estimating Alignment Parameters (Step B)

In order to use the example set, the relative alignment on the example set must be computed. For a cyclic activity, such as ShoulderCircles, the alignment is necessary to account for different phases of the single period extracted from the cyclic recording. For a non-cyclic activity, such as LeftStep, alignment is necessary to account for the reaction delay between start of recording and start of performance as well as subjects performing the activity at different speeds. The method computes the alignment by finding two vectors; the scaling vector $\boldsymbol{\alpha}^*$, and phase translation vector $\boldsymbol{\phi}^*$, which minimize

$$\|\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\phi})\|. \tag{4.5}$$

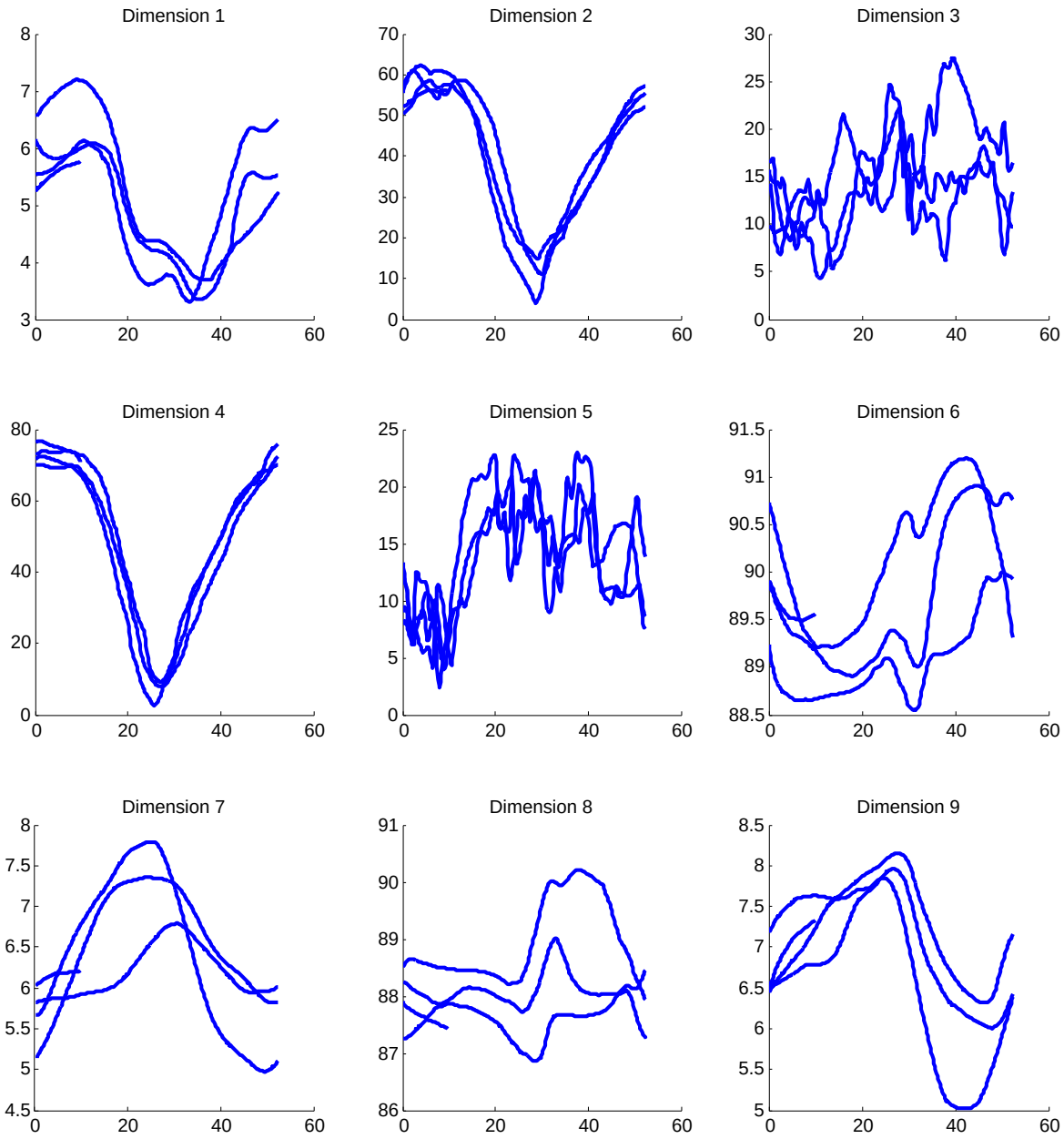

Figure 4.4: The angle of each dimension for a single period found by the Cycle Extraction algorithm applied to a recording of Shoulder Circles activity.

$\mathbf{G}$ is a matrix which represents the difference between each example for given alignment vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$. Entry j, k of the matrix $\mathbf{G}$ is expressed as

$$\int_0^1 \delta\left(\mathbf{e}_j\left(\alpha_j\left(t+\phi_j\right)\right), \mathbf{e}_k\left(\alpha_k\left(t+\phi_k\right)\right)\right)^2 \, \mathrm{d}t \tag{4.6}$$

The two vectors' define the start and time scaling of each example that best aligns the examples. The starting point on $\mathbf{e}_j\left(t\right)$ is expressed as $\alpha_j\left(0+\phi_j\right)$ while the ending point is expressed as $\alpha_j\left(1+\phi_j\right)$. In this thesis, the minimization is computed by MATLAB's `fmincon` function utilizing the *interior-point* algorithm, while the integral in Equation 4.6 is computed computed by MATLAB'S `quad` function. Further, at the solution point, the entries of $\mathbf{G}(\boldsymbol{\alpha}^*, \boldsymbol{\phi}^*)$ can be examined to verify that all of the sequences are aligned, where entries much larger than the others indicate an issue.

Figure 4.5 and Figure 4.6 show this alignment being applied to ten examples of Shoulder-Circles and LeftStep activities.

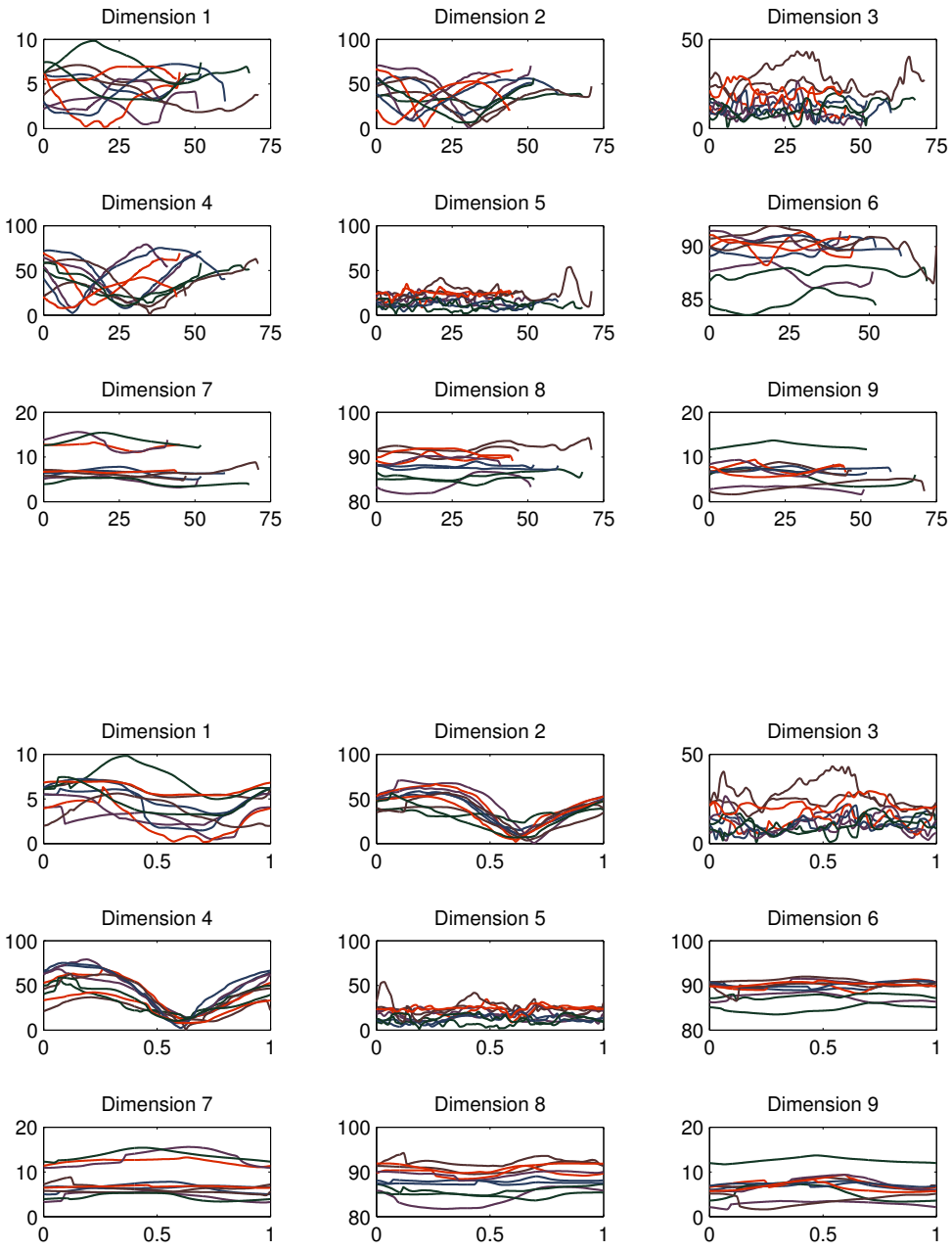

Figure 4.5: The top set of nine graphs show the angle of each dimension in respect to time for ten unaligned and unscaled examples of the ShoulderCircle. The bottom set of nine graphs shows the angle of each dimension after $\boldsymbol{\alpha}^*$ and $\boldsymbol{\phi}^*$ have been applied in respect to time normalized by the $\boldsymbol{\phi}^*$ parameter.

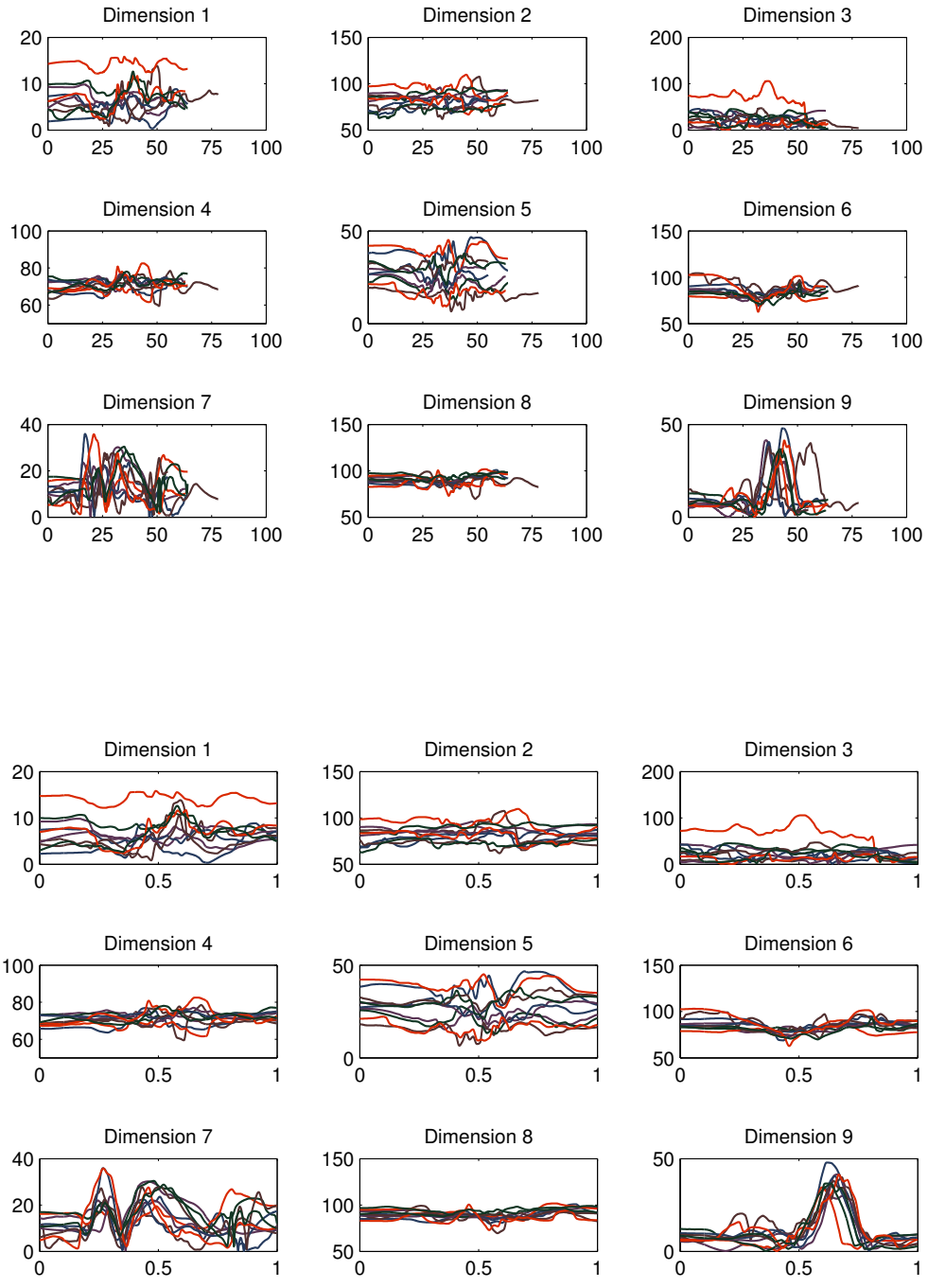

Figure 4.6: The top set of nine graphs show the angle of each dimension in respect to time for ten unaligned and unscaled examples of the LeftStep. The bottom set of nine graphs shows the angle of each dimension after $\boldsymbol{\alpha}^*$ and $\boldsymbol{\phi}^*$ have been applied in respect to time normalized by the $\boldsymbol{\phi}^*$ parameter.

Computing the Continuous Activity Function (Step C)

With an aligned example set, the activity archetype is now ready to be created. The activity archetype, $\mathbf{a}$, is formally defined as a continuous function of time t in quaternion space, where

$$\mathbf{a} : t \rightarrow \mathbb{H}^N \tag{4.7}$$

The continuous activity function is computed using the minimal bounding circle on the parametric surface of all M aligned quaternion rotation sequences. A function, `minBoundQuat`, accepts a vector of quaternions $\boldsymbol{\beta}$ and computes a single quaternion q_{mid} which satisfies

$$q_{mid} = \arg\min_{q} \max_{j} \ \theta\left(q, \ \beta_j\right). \tag{4.8}$$

When that equation is satisfied, q_{mid} will be set to the quaternion that is a minimal distance away from the furthest quaternion in $\boldsymbol{\beta}$.

Figure 4.7 shows the minimal `minBoundQuat` technique being applied to the left shin of 10 subjects standing. The top graph of Figure 4.7 shows 10 quaternion's rotation of a unit vector, while the bottom graph shows each quaternion's parametric representation.

Using the `minBoundQuat` function, the i^{th} dimension of the activity archetype is expressed as

$$a(t)_i = \texttt{minBoundQuat}\left(\left[e_1\left(\alpha_1\left(t+\phi_1\right)\right)_i, \ e_2\left(\alpha_2\left(t+\phi_2\right)\right)_i, \ \ldots, \ e_M\left(\alpha_M\left(t+\phi_M\right)\right)_i\right]\right) \tag{4.9}$$

where $e_j\left(t\right)_i \in \mathbb{H}$ expresses the i^{th} dimension of the j^{th} example at time t.

Using a minimal bounding circle to compute the archetype is preferred over the simpler approach of taking the statistical mean of the examples. The issue that arises with a statistical

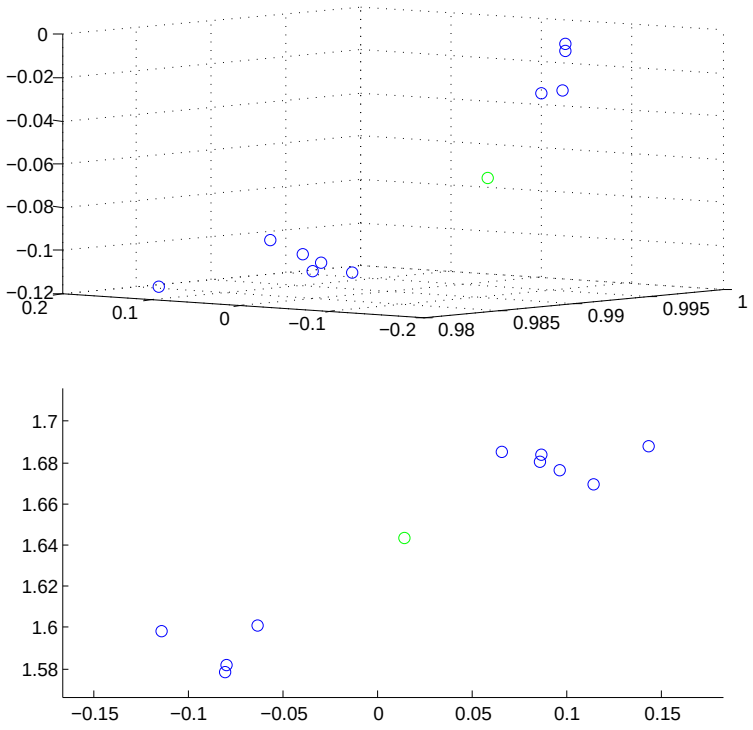

Figure 4.7: Applying the minimum bounding circle algorithm on a set of rotations. The top graph shows the XYZ result of rotating the unit vector by the quaternions. The bottom graph shows the respective parametric parameters. The green point on both graphs indicates the point in the minimal bounding circle.

mean is that the result could be skewed by redundant examples in the set. When building the archetype, it is desired to represent *all* variations of the activity; the mean being skewed toward any particular value is detrimental to this goal. Using the minimal bounding circle is an approach that places the archetype such that it minimizes the maximum distance to *any* variation seen in the example set. The minimal bounding circle approach is not affected by any redundant examples that may be present in the example set.

Computing Weights (Step D)

To account for anthropometric variations among the example set, a set of **weights** are used in conjunction with a distance metric on the archetype. The weights directly indicate how important a particular joint's rotation is in identifying the activity. For example, in observed walking sequences, leg segments have a consistent, cyclic behavior across all subjects, but the movement of the arm segments exhibits little consistency across subjects. Therefore, the weights computed for the rotation of the arm segments would be much lower than weights assigned to the rotation of the leg segments.

The weights are defined as

$$\mathbf{w} \in \mathbb{R}^N,\ \mathbf{w} \geq 0, \tag{4.10}$$

A distance metric on the archetype uses the weights to compute the distance between two poses $\mathbf{p}_1$ and $\mathbf{p}_2$. The distance metric is defined as

$$\eta\left(\mathbf{p}_1, \mathbf{p}_2\right)_{\mathbf{w}} = \max\left(\mathbf{w} \odot \mathbf{d}\left(\mathbf{p}_1, \mathbf{p}_2\right)\right) \tag{4.11}$$

where $\odot$ indicates the Hadamard Product.

A maximum in the distance metric is used because it is desired to know any time the metric

is outside the variation defined by the weights. Consider two poses where the distance between them is barely inside the weights for each dimension. Now consider two poses where the distance between them is outside the weights for one dimension and equal on the rest. Because the second scenario contained a dimension outside the weights, it is desired that the score be *larger* than the first. Using the maximum is the only metric which can generally indicate when the distance between the two poses is outside the variation allowed for the activity in any dimension.

The activity archetype $\mathbf{a}(t)$ computed above, the i^{th} element of the weight vector is expressed as

$$\frac{1}{w_i} = \int_0^1 \max_{j \in [1,\ M]} d_i \left(\mathbf{a}(t),\ \mathbf{e}_j \left(\alpha_j \left(t + \phi_j \right) \right) \right) dt, \tag{4.12}$$

the average of the maximum distance the archetype is from any aligned example at time t.

Figure 4.8 and Figure 4.9 show a visualization of the weights for ShoulderCircle and LeftStep activity respectively. The ShoulderCircles activity contains more variation in the moving arm dimensions, which are performing the circles at a variety of angles, compared to the relatively static dimensions of the standing legs. The LeftStep activity contains more variation in the arm dimensions, which are not necessary to perform the action, compared to the leg dimensions, which must move in a certain manner to perform the action. Additionally because the LeftStep activity was performed with the right side of the subject facing the Kinect, as will be described in Section 5.3, there was more error in detecting the subject's far side (left). Therefore LeftStep weights have more variation on the left side for both the arms and legs.

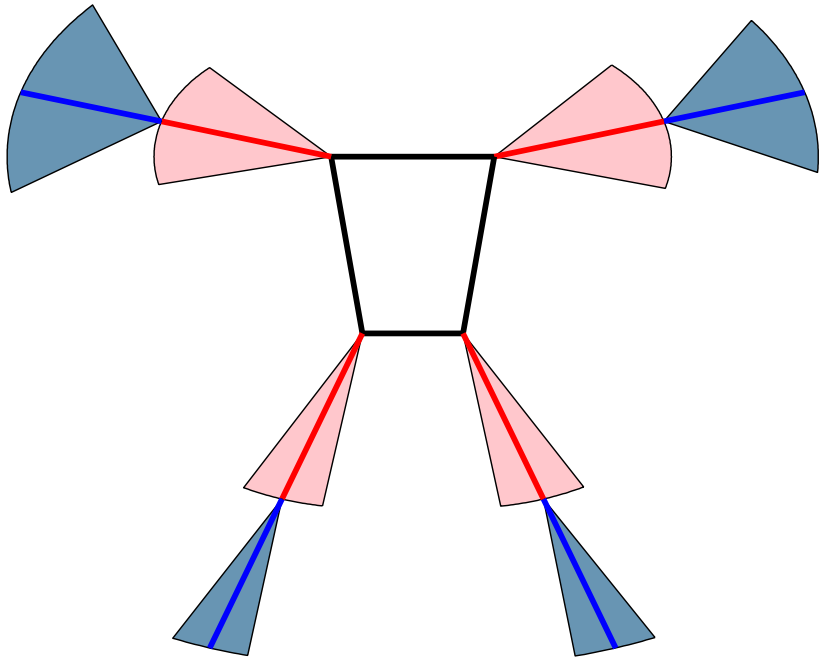

Figure 4.8: A representation of the amount of allowed variation as defined by the weights on each dimensions for the ShoulderCircles activity. The larger the cone on a segment, the more variations allowed on that segment's dimension.

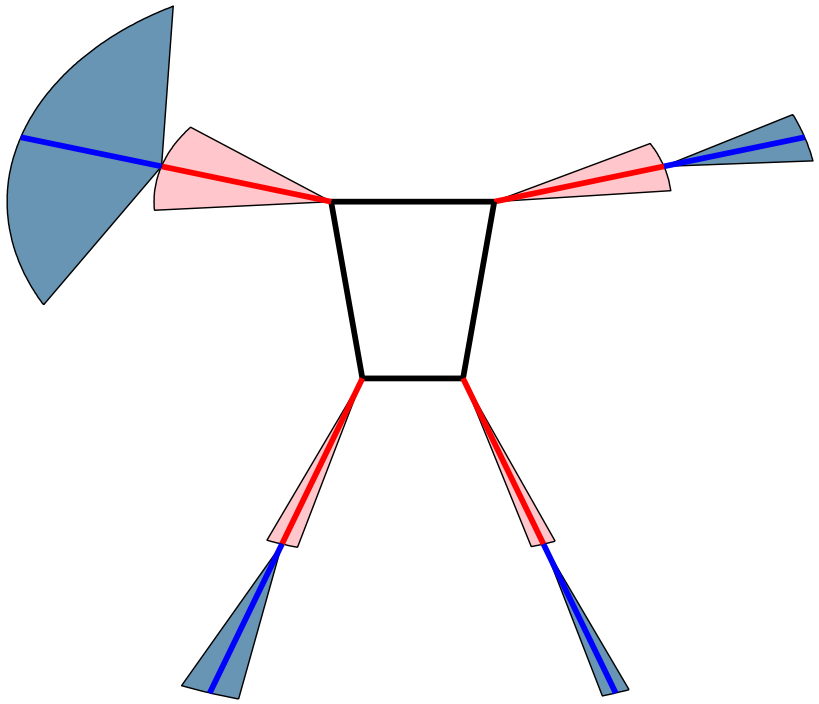

Figure 4.9: A representation of the amount of allowed variation as defined by the weights on each dimensions for the LeftStep activity. The larger the cone on a segment, the more variations allowed on that segment's dimension.

4.1.2 Finding Alphabet and Regular Expression (Step 2)

In order to match an observation to an archetype, the same optimization of alignment parameters discussed in the previous section could be used. However, this approach is computationally demanding and not well suited for real-time use on a wearable system. Instead, the following describes the second step towards creating and using an activity archetype, the generation of an alphabet and associated regular expression using the continuous curve, $\mathbf{a}$, and weights, $\mathbf{w}$, computed above. The regular expression allows the use of a less computationally intensive matching algorithm based on string matching [14]. The generation of the regular expression is performed offline; the regular expression itself is used in the online string-matching operation.

The alphabet, Σ, used by the string matching algorithm, is comprised of a set of characters

$$\Sigma = \left[\mathbf{c}_1 \ \mathbf{c}_2 \ \ldots \ \mathbf{c}_{|\Sigma|}\right] \tag{4.13}$$

where each character is a hypersphere in $\mathbb{H}^N$ space. The size of the characters is determined by a single parameters ρ which can be varied to determine the size of the alphabet. Given a time t in an activity, $\mathbf{a}$, it is required that the associated alphabet satisfy

$$t \ \exists \left(\ \mathbf{c} \in \Sigma\right) \ \mid \ \eta(\mathbf{a}(t), \ \mathbf{c})_{\mathbf{W}_k} \leq \rho. \tag{4.14}$$

Figure 4.10 illustrates the basic concept by displaying 7 characters of an alphabet for a 2 dimensional function in $\mathbb{R}$ space. In this example dimension 2 has larger weight than dimension 1, allowing dimension 1 to vary more before forming a new character. When viewed with unweighted distance, as plotted in the figure, the region surrounding each character is an ellipsoid.

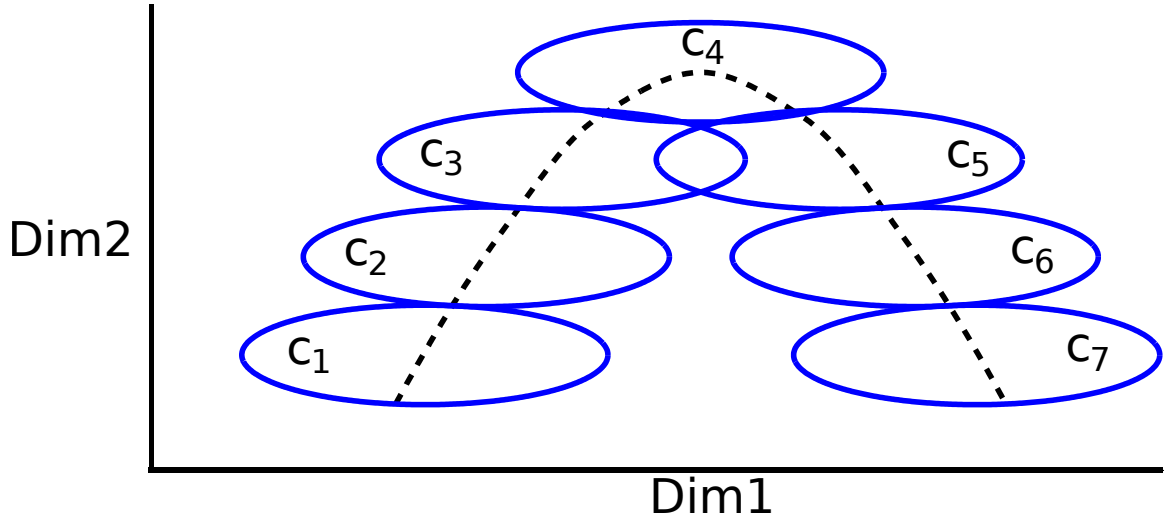

Figure 4.10: A simple example of an alphabet for a two dimensional function in $\mathbb{R}$ space.

For a given archetype, $\mathbf{a}$ and radius ρ, Algorithm 1 computes the set of characters $\mathbf{c}$ and lengths of each character $\mathbf{l}$. The run-time complexity of the algorithm depends on how much the shape of the activity archetype after its weights have been applied. For the character $c_n \in \mathbf{c}$, the minimum, ξ_n, and maximum, λ_n, length is computed as

$$\begin{aligned} \xi_n &= l_n - \left(\max_{j \in [1,\ M]} |e_j| - \min_{j \in [1,\ M]} |e_j| \right) \\ \lambda_n &= l_n + \left(\max_{j \in [1,\ M]} |e_j| - \min_{j \in [1,\ M]} |e_j| \right), \end{aligned} \tag{4.15}$$

where $|e_j|$ is the length of the j^{th} example. The regular expression is then constructed as,

$$R = u_1 u_2 \dots u_{|\Sigma|} \tag{4.16}$$

where u_n is expressed as

$$u_n = \mathbf{c}_n^{\xi_n} \left(\varepsilon \mid \mathbf{c}_n^1 \mid \mathbf{c}_n^2 \mid \dots \mid \mathbf{c}_n^{(\lambda_n - \xi_n)} \right). \tag{4.17}$$

Algorithm 1 Compute the alphabet from an activity archetype

function COMPUTEALPHABET($\mathbf{a}, \rho$)
  $start \leftarrow 0$
  $end \leftarrow 1$
  $\mathbf{c} \leftarrow \{\}$
  $\mathbf{l} \leftarrow \{\}$
  while $start < end$ **do**
    $x \leftarrow \min_{t \in [start,\ end]} \eta\left(\mathbf{a}\left(start\right), \mathbf{a}\left(t\right)\right)_{\mathbf{w}} \geq \frac{\rho}{2}$
    if $x \leq end$ **then**
      $y \leftarrow end$
    else
      $y \leftarrow \min_{t \in [x,\ end]} \eta\left(\mathbf{a}\left(x\right), \mathbf{a}\left(t\right)\right)_{\mathbf{w}} \geq \frac{\rho}{2}$
    end if
    $\mathbf{c} \leftarrow \mathbf{c} \cup \mathbf{a}\left(x\right)$
    $\mathbf{l} \leftarrow \mathbf{l} \cup \left(y - x\right)$
    $start \leftarrow y$
  end while
  return $[\mathbf{c},\ \mathbf{l}]$
end function

The creation of a new character can be invoked by any dimension of the model. A new character will often be caused by important dimensions whose value is non-constant. For example, the left and right upper arm during Shoulder Circles (second and fourth dimension in Figure 4.12) force the creation of a new character, as does the left and right shin in LeftStep activity (seventh and ninth dimension in Figure 4.11).

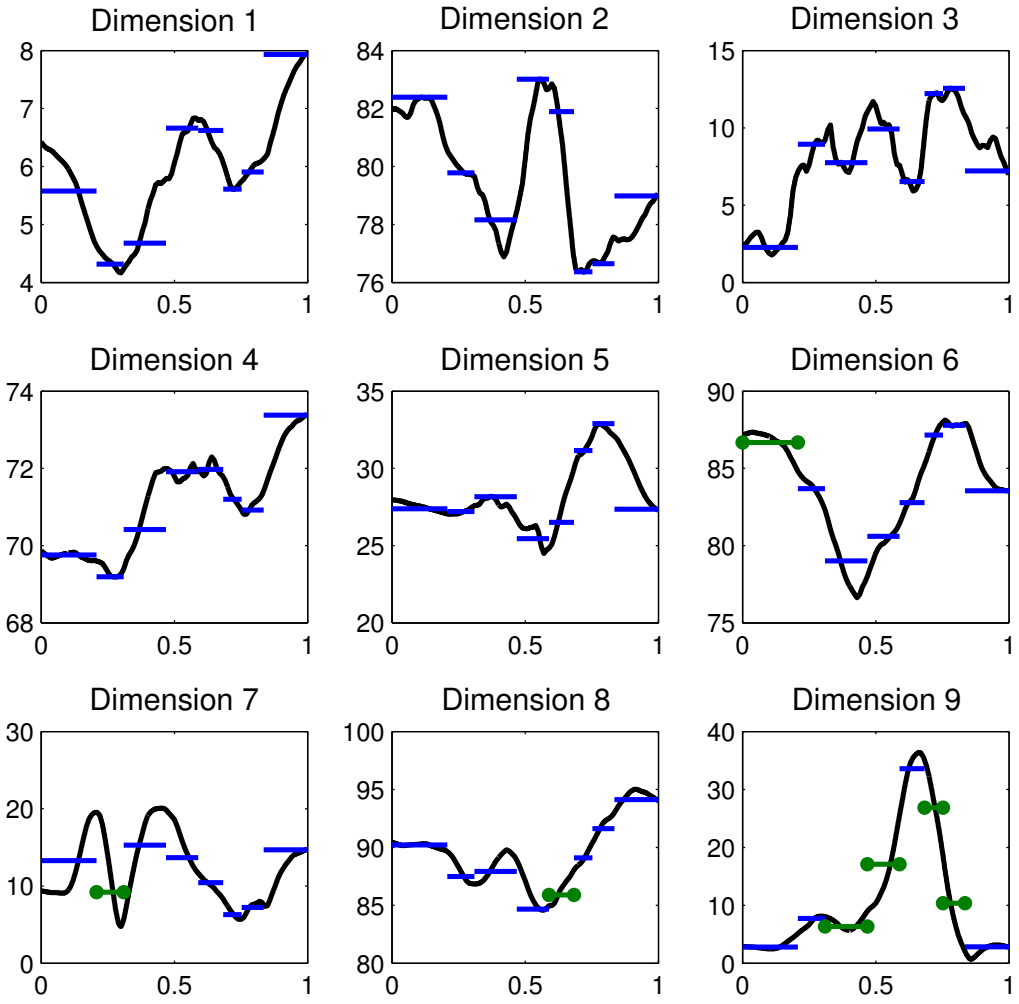

Figure 4.11: Superposed graph and created alphabet for the LeftStep Activity on all dimensions. Each horizontal line represents a character. A green line marked with two dots indicates the dimensions which caused the creation of the character.

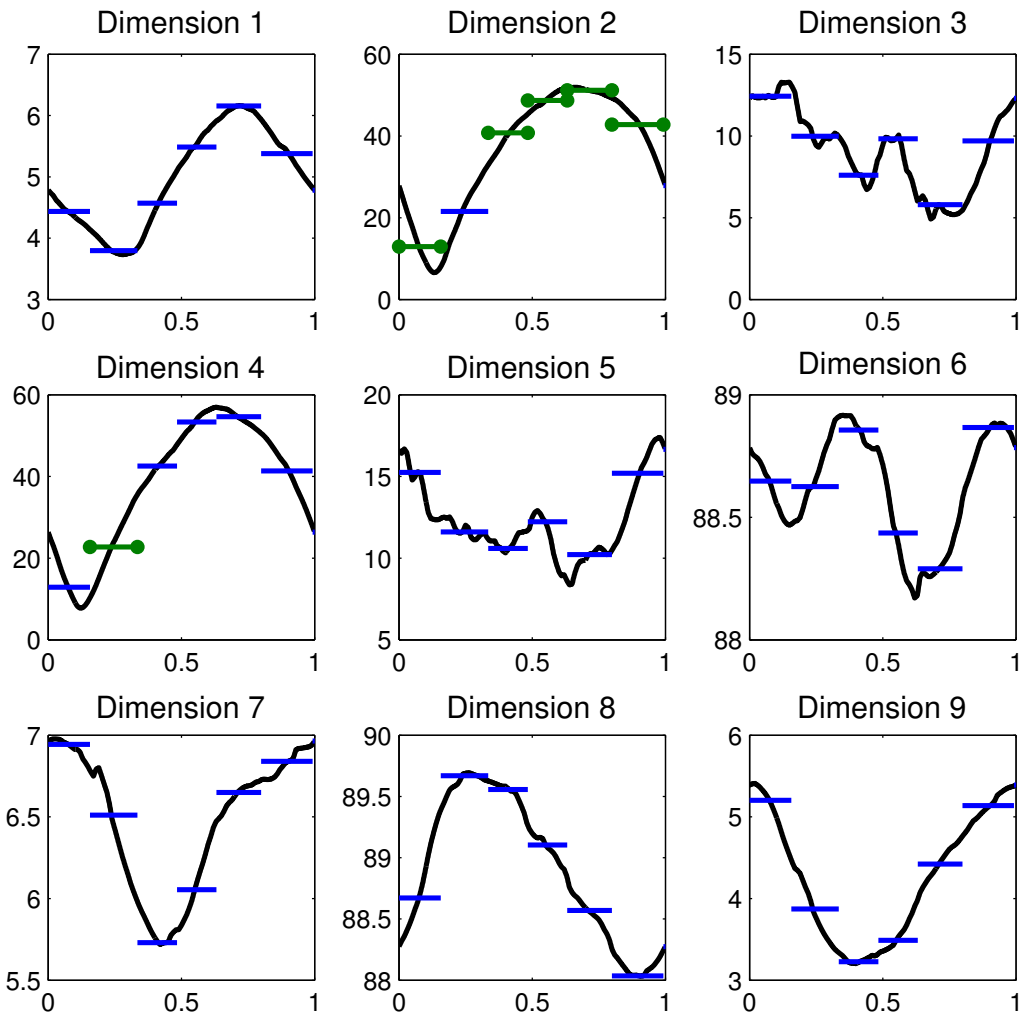

Figure 4.12: Superposed graph and created alphabet for the ShoulderCircles Activity on all dimensions. Each horizontal line represents a character. A green line marked with two dots indicates the dimensions which caused the creation of the character.

4.1.3 Matching Observed to Regular Expression (Step 3)

The third and final step toward creating and using an activity archetype in MUAC uses the regular expression associated with the activity, R_a, to find the match of an observation to the activity in real-time. Given a discrete observed sequence in the form of

$$\mathbf{o} : t \rightarrow \mathbb{H}^N, \tag{4.18}$$

a vector of how well the observation matches to an activity is computed with the function

$$M : [\mathbf{o}, R_a] \rightarrow \mathbb{R}^N. \tag{4.19}$$

Taking the maximum element of the vector returned gives the minimal edit distance of $\mathbf{o}$ to a pattern allowed by the regular expression R_a. This function is a standard string matching algorithm that allows substitutions of an observed character to the activity characters and deletion of an observed character. In general the computational complexity of editing an observation to an pattern is $\mathcal{O}(|\mathbf{o}||R_a|)$. In practice, this complexity is limited by the min/max characters allowed by the regular expression. The cost of editing the character ι in the observation to character τ in the regular expression is expressed as

$$C_{\iota,\tau} = \mathbf{d}(\mathbf{o}[\iota],\ \mathbf{c}_\tau) \odot \mathbf{w}. \tag{4.20}$$

Finally, a criteria T is defined as the maximum value the observation is allowed to match. If an observation matches above the threshold for some activity, then the observation cannot be classified as that activity. The computation of T is application specific and discussed in Section 5.3.

4.2 Algorithm Element: Grouping Similarity

The methodology described in Section 4.1 is used to construct the **inside set**, a set of activities which the MUAC is expected to be able to classify. A simple approach toward classifying an observation computes the match of the observation to each inside activity. This classification approach is the **flat structure**; it is organized such that each activity is equal compared to each other. Classifying using the flat structure is comparable to a linear search in complexity. To reduce this complexity, this section introduces the **tree structure**. The tree is organized as a hierarchy based on the inside activities' similarity to one another. For example, brushing hair and brushing teeth both involve standing up; therefore, the dimensions associated with the legs are expected to overlap. Classifying on the tree structure is comparable to a tree search, entire branches of activities can be disregarded with one match score. The use of the tree counteracts the increased complexity as more activities are added into the inside set, which in turn allows the MUAC to be scalable.

The similarity between two activities is quantified as the number of dimensions that overlap with one another. The overlap is determined between two activities, a_1 and a_2, by a function

$$\texttt{overlap} : (\mathbf{a}_1,\ \mathbf{a}_2) \rightarrow \mathbb{R}^N. \tag{4.21}$$

An algorithm, shown in Algorithm 2, uses the `overlap` function and a set of K activities, $\mathbf{A} = \{\mathbf{a}_1,\ \mathbf{a}_2,\ \ldots,\ \mathbf{a}_K\}$, to construct the tree structure tree. This algorithm has an overall complexity of $\mathcal{O}\left(|A|^2\right)$ because each pair of activities overlap must be investigated. The tree is organized by the number of dimensions that are shared between the children activities. The deeper any two activities share an ancestor indicates a greater number of dimensions that overlap between the activities. The activities contained in A used to create the tree are leaf nodes on the tree structure. An branch node on the tree is itself an activity created by

combining the examples of all the leaf activities under the node. An observation is classified by doing a tree search on the structure. A child node of the root whose match is both below the node's threshold and minimum among all the root's node is selected. If that is in turn itself an branch node then this process repeats until a leaf node is selected, or no leaf node can be selected.

Using such a tree structure has two benefits:

- **Complexity Reduction** - The amount of activities that must be checked for each observation can be reduced since many similar activities can be eliminated at one time.
- **Proximity of Unknown** - When given an unknown activity that does not match into any activity in **A**, the tree can be used to determine how similar the unknown activity is to a known activity.

Algorithm 2 Recursively compute hierarchy for **A**, the set of activities

Require: $topNodes - These of nodes which have no parent.$
Require: $D - Dimension requiriment, initially set to N.$
Require: $T - Thresholdcriteria.$
function CREATETREE($topNodes,\ D,\ T$)
$Nodes \leftarrow \{\}$
for all $node1 \in topNodes$ **do**
for all $node2 \in topNodes$ **do**
if $node1 == node2$ **then**
continue;
end if
$\mathbf{a}_1 \leftarrow node1.leafNodes$
$\mathbf{a}_2 \leftarrow node2.leafNodes$
$Signature \leftarrow \texttt{zero\_array}(N)$ ▷ Empty Signature
for $i \leftarrow 1$ to N **do**
if $\max_{\alpha \in a_1,\ \beta \in a_2} \texttt{overlap}\,(\alpha,\ \beta) \leq T$ **then**
$Signature[i] \leftarrow 1$ ▷ Set signature for this dimension
end if
end for
if $\texttt{sum}\,(Signature) \geq D$ **then**
if $Node \in Nodes$ and $Set.Signature == Signature$ **then**
if $node1 \in Node$ **then**
$Node = Node \bigcup node2$
else if $node2 \in Node$ **then**
$Node = Node \bigcup node1$
else
$Node = Node \bigcup node1$
$Node = Node \bigcup node2$
end if
else ▷ Create new set
$NewNode \leftarrow \{node1,\ node2\}$
$NewNode.Signature \leftarrow Signature$
$Nodes = Nodes \bigcup NewNode$
end if
end if
end for
end for
$TopNodes \leftarrow topLevelNodes()$ ▷ Get all top level nodes
if $TopNodes.Count > 1$ and $D > 1$ **then** ▷ Defines when to stop recursion
createTree($topLevelNodes(),\ D-1,\ T$)
end if
end function

4.3 Algorithm Element: Removing Redundancy

The construction of a user-independent archetype detailed in Section 4.1 relies on using an example set that expresses the variations of the population performing an activity. In this section, a metric is introduced that is used to measure how much variations a specific example contributes to the example set. The metric is used during the construction of an example set and has no bearing on the archetype created by an example set. This section also introduces a method to remove examples that do not sufficiently contribute new variations. Removing non-important examples during the construction of an example set is not absolutely necessary since the `minBoundQuat` algorithm is not affected by repeating elements; however, the removal of such examples is beneficial by simplifying the creation of an activity archetype.

The importance of a specific example among a set of M examples, $\mathbf{E} = \mathbf{e}_1,\ \mathbf{e}_2,\ \ldots,\ \mathbf{e}_M$, is determined by the ratio of the archetype's weights constructed with and without the example. Given the two weight vectors are $\mathbf{w}_{with}$, the weights of the full example set and $\mathbf{w}_{without}$, the weights of the example set with example i removed, the importance of example i is expressed as

$$\texttt{impact}\,(\mathbf{E},\ i) = 1 - \min \mathbf{w}_{without} \odot \hat{\mathbf{w}}_{with}, \tag{4.22}$$

where $\odot$ indicates the Hadamard product operation and $\hat{}$ indicates the Hadamard inverse operation. The result of this function is a single number in $\mathbb{R}$ space that represents the relative change in the weights with and without example i.

Using the `impact` function, Algorithm 3 gives an algorithm to remove non-important examples in $\mathbf{E}$, using a threshold, Δ, to decide if an example's impact is enough to keep. This algorithm has a computational complexity of $\mathcal{O}\left(|\mathbf{E}|\right)$, the impact of each example in the example set is computed by this algorithm. If multiple examples have an impact less than

Delta, the example with the smallest impact is selected and removed. This process is then repeated on the modified example set until no example is found to have an impact less than *Delta*.

Algorithm 3 Eliminate redundant examples from a set

function REMOVEREDUNDANT($\mathbf{E}, \Delta$)
  for $j = 1 \rightarrow M$ **do**
    $d_j \leftarrow$`impact`$(\mathbf{E}, j)$
  end for
  if $\min \mathbf{d} > \Delta$ **then**
    return E
  else
    $\mathbf{E}^{'} \leftarrow \mathbf{E} \cap \mathbf{E}_{\arg\min \mathbf{d}}$
    return `removeRedundant`$\left(\mathbf{E}^{'}\right)$
  end if
end function

Chapter 5

Experiment

This chapter details the implementation of the MUAC and introduces experiments done on various characteristics of the MUAC method. This chapter is organized as follows: Section 5.1 describes how each component discussed in Chapter 4 was implemented and the constraints placed on the implementation. Section 5.2 describes the framework developed to allow data collection of human poses. Finally Section 5.3 describes the activities tested, the data collection process, and the experiments done in order to test the quality of the MUAC method.

5.1 Implementation

This section describes how each component of the MUAC method was implemented such that the method could be used to perform experiments. Section 5.1.1 gives details on the Cycle Extraction component described in Section 4.1.1. Section 5.1.2 gives details on the Sequence Alignment component described in Section 4.1.1. Finally, Section 5.1.3 gives details on the creation of the archetype and alphabet.

5.1.1 Finding Single Cycle

The Cycle Extraction component is responsible for the implementation of Equation 4.3 from Section 4.1.1, which is used to extract a single cycle from a cyclic recording. Given a cyclic function of four or more periods, there may exist multiple solutions at harmonies of the cyclic function. For example, a $sin(t)$ will minimize Equation 4.3 when $t = i \cdot 2\pi \; i > 0$.

Because the underlying curve may contain many cycles, it is necessary to make sure that only a single cycle has been extracted. The global minimum, x^*, and the value of Equation 4.3 at that point, y^*, are parameters computed after the first minimization. A smaller harmony is found by searching for the parameters, x and y, which satisfies

$$\begin{aligned} y &\leq 1.1y^* \\ x^* &> x + \frac{x}{2}. \end{aligned} \tag{5.1}$$

The parameters are selected such that x is outside the global min and evaluates to a value that is at least 90% of the global min y^*. This step is repeated until the above criteria cannot be satisfied.

For some activities it may be *desired* to have multiple cycles. It is known from the activity the example set represents how many cycles of a cyclic activity are needed. To extract N cycles, the above steps are performed to extract a single activity, then two parameters x_N and y_N are found which satisfy

$$\begin{aligned} y_N &\leq 1.1y^* \\ x_N &> (N-1) \cdot x + \frac{x}{2} \\ x_N &< (N+1) \cdot x - \frac{x}{2}. \end{aligned} \tag{5.2}$$

x_N is selected such that its value is within 10% of the global minimum, and it is inside the appropriate harmony.

5.1.2 Sequence Alignment

The sequence alignment component is responsible for the implementation to perform the minimization of Equation 4.6, which is used to align a set of curves in respect to one another. When minimizing Equation 4.5, it is necessary to have some constraints on the alignment parameters, without constraints the parameters could create unintended alignments. For example, without constraints a set of non-cyclic curves could be aligned such that the scaling parameter is very small where all the curves happen to have the same value.

In this thesis the constraint is achieved by selecting a single curve as a reference. For the reference curve, the phase translation parameter, ϕ, is set to zero and the scaling parameter, α, is set to the length of the curve. The parameters for the remaining curves are solved by minimizing Equation 4.5. This process is done iteratively, with each curve being considered as the reference. The reference curve which produces the best alignment, as determined by the lowest evaluation of Equation 4.5, is used as the ultimate alignment of the curves. By iterating over all possible reference curves, it is ensured that no curve is held above the others.

5.1.3 Activity Archetype Construction

An activity archetype is constructed iteratively from a bin of potential examples. First, a random ten examples are pulled from the bin and are aligned with respect to each other. Algorithm 3 is applied to remove any unnecessary examples according to a Δ value of zero. With a value a zero, an example would only be removed if it contributes *no* new information.

Next an additional random example is chosen, and the new curve and the previously aligned step are aligned with each other. Algorithm 3 is applied again on the new example set, removing any unnecessary examples. This process is repeated until all examples have been considered; this final archetype is what is used as the archetype. With the constructed archetype the alphabet is computed using a radius $\rho = 0.5$. This value was specifically chosen such that the diameter of each character would be 1.

5.2 Testing Framework

In order to perform experiments on the MUAC, a framework was developed that utilizes a data-source to construct pose sequences of activities. This thesis uses a motion capture system to convert a set of XYZ points on the human body to generate the pose, as defined in Chapter 4. The framework was developed in C++ and can accept two data-sources: C3D files using the interface provided by C3DServer[6], and a Microsoft Kinect using the interface provided by Microsoft's Kinect For Windows SDK[7].

Both C3D and Kinect for Windows SDK provide a vector, $\mathbf{v}$, of 14 XYZ positions points on the human body. Table 5.1 details what point on the body each entry in $\mathbf{v}$ corresponds to. With $\mathbf{v}$, the pose, as defined in Chapter 3, is constructed by using the algorithm given in Algorithm 4.

Algorithm 4 Algorithm to compute the pose from a set of points on the body. Refer to Appendix B for the definition of `VectorRotate` and `VectorQuaternion` functions.

function GETPOSE($\mathbf{v}$)
  $q_T \leftarrow$ a quaternion which satisfies the following assertions.
  $R\_Hip \leftarrow$ `VectorRotate` $(q_T,\ v_{10})$
  $L\_Hip \leftarrow$ `VectorRotate` $(q_T,\ v_7)$
  $C\_Hip \leftarrow$ `VectorRotate` $(q_T,\ v_{13})$
  $C\_Sho \leftarrow$ `VectorRotate` $(q_T,\ v_{14})$
  `assert` $(L\_Hip_1 > R\_Hip_1,\ L\_Hip_2 = R\_Hip_2,\ L\_Hip_3 = R\_Hip_3)$
  `assert` $(C\_Sho_1 = C\_Hip_1,\ C\_Sho_2 = C\_Hip_2,\ C\_Sho_3 > C\_Hip_3)$

  $\mathbf{v}^* \leftarrow$ Each element in $\mathbf{v}$ rotated by q_T.
  $[q_{LUA},\ q_{LFA}] \leftarrow$ `LimbQuaternions` $(\mathbf{v}_1^*,\ \mathbf{v}_2^*,\ \mathbf{v}_3^*)$
  $[q_{RUA},\ q_{RFA}] \leftarrow$ `LimbQuaternions` $(\mathbf{v}_4^*,\ \mathbf{v}_5^*,\ \mathbf{v}_6^*)$
  $[q_{LT},\ q_{LS}] \leftarrow$ `LimbQuaternions` $(\mathbf{v}_7^*,\ \mathbf{v}_8^*,\ \mathbf{v}_9^*)$
  $[q_{RT},\ q_{RS}] \leftarrow$ `LimbQuaternions` $(\mathbf{v}_1^*0,\ \mathbf{v}_1^*1,\ \mathbf{v}_1^*2)$
  $\mathbf{p} \leftarrow [q_T\ q_{LUA}\ q_{LFA}\ q_{RUA}\ q_{RFA}\ q_{LT}\ q_{LS}\ q_{RT}\ q_{RS}]$
  return p
end function

function LIMBQUATERNIONS($Start,\ Mid,\ End$)
  $q_U \leftarrow$ `VectorQuaternion` $(Start, Mid)$
  $Mid^* \leftarrow$ `VectorRotate` $(q_U,\ Mid)$
  $End^* \leftarrow$ `VectorRotate` $(q_U,\ End)$
  $q_F \leftarrow$ `VectorQuaternion` (Mid^*, End^*)
  return $[q_U,\ q_F]$
end function

v_i	Point on Body
1	Left Shoulder
2	Left Elbow
3	Left Wrist
4	Right Shoulder
5	Right Elbow
6	Right Wrist
7	Left Hip
8	Left Knee
9	Left Ankle
10	Right Hip
11	Right Knee
12	Right Ankle
13	Center Hip
14	Center Shoulder

Table 5.1: Points required to construct the pose of the user

5.3 Experiment Set Up

The framework discussed above is utilized in this section to collect data of subjects performing various activities. The activities investigated in this thesis are split into two sets. The **inside set** contains activities for which an archetype is constructed using the methodology described in Chapter 4. Any observation of an inside activity should be matched to an archetype known by the classifier. The **outside set** contains activities whose archetype is not constructed. Any observation of an outside activity should classified by the MUAC as unknown, the observation does not match to any archetype.

All recordings of a subject's trials are placed into one of two partitions. The **example partition** contains recordings which are used to construct the archetype for the inside activities. The **observed partition** contains recordings which are used as observations in the experiments of this thesis. Refer to Table 5.2 for a breakdown of the interaction between the inside/outside set and example/observed sequences. Because each partition contains different subjects, any conclusions drawn in the results of this thesis are user-independent.

Ideally, publicly available motion capture data would have been used as the data source;

	Observed Partition	Example Partition
Inside Set	Observations to the classifier that should match to an archetype.	Examples to construct archetypes with.
Outside Set	Observations to the classifier that should match as "unknown" or to a branch node.	N/A

Table 5.2: The interaction between the activity sets and recording partitions.

however, such data does not consistently contain recordings of multiple subjects performing the same activity multiple times. Therefore a user study was performed using the Microsoft Kinect as a motion capture system. As a motion capture system, the Kinect has a view capability of 57.5° horizontally and 43.5° vertically with a depth of 1.2 to 3.5 meters [26]. Due to this limited field of view, the activities chosen had to be contained in this space. Running or walking could not be done since the subject would move out of view before the entire activity could be recorded by the Kinect.

A total of 25 subjects were recorded performing each activity five times. Of the 25 subjects, 18 of them were male while 7 were female. The subjects tested had an average age of 29 years old, an average weight of 156 pounds, and an average height of 69 inches. Any recording which the subject failed to follow instructions or the moderator failed to capture the subject's activity were removed from the data set. For each inside activity, a random 10 subjects are selected and all the trials the selected subjects were placed into the example partition. The remaining 15 subject's trials are placed into the observed partition.

For each of the 25 subjects the following eight activities were recorded. The activities were chosen because they are activities which a physical therapist may ask a patient to perform [27][28] or are activities which a doctor may be interested in knowing if and when a subject performed [2]. Another consideration in choosing the activities was activities that are similar to others to test MUAC's ability to distinguish similar activities. Cyclic activities are marked by "*cyclic*", if no text is given the activity is non-cyclic. Unless otherwise noted, a single period was extracted from the cyclic activities.

- **LeftStep** - The subject performs a single step forward with their left foot.
- **LeftStepUp** - The subject performs a single step forward with their left foot, onto a ten inch high step stool placed at a distance natural to the subject.
- **ShoulderCircles**[*cyclic*] - Standing with arms extended, the subject rotates both arms around shoulder together. From the subject's point of view, the left arm rotates clockwise, and the right arm counterclockwise. The subject is asked to maintain a 45° angle to the rotation axis, and to perform one rotation in 2 seconds.
- **RightShoulderRotation** - Standing with the right elbow pinned at the hip and the left hand across belly, the subject rotates the shoulder as far as comfortable keeping the elbow on hip. Then the subject returns the hand to the starting position. The subject is asked to perform this in about 3 seconds, without significantly pausing when changing directions bringing the arm back.
- **DrivingTwoHands**[*cyclic*] - The subject sits in a chair and is asked to behave like driving, feet at pedals and both hands on steering wheel.
- **RightBrushHair**[*cyclic*] - While standing, the subject makes a motion as if they are brushing their hair with their dominant hand. For consistency across a range of hair styles, subjects were told to brush as if they had short, Caesar cut length hair. If the subject is left handed, this activity is mirrored.
- **RightBrushTeeth**[*cyclic*] - While standing the subject makes a motion as if they are brushing their teeth with their dominant hand. If the subject is left handed, this activity is mirrored. For this activity two cycles were extracted such that the length is consistent with the other activities.
- **RightEat**[*cyclic*] - Sitting down at a table, the subject acts like they are eating soup

from a bowl with their dominant hand. If the subject is left handed, this activity is mirrored.

In addition, ten subjects performed nine activities that are only inserted into the example partition; none of the recordings of these activities act as observations. The purpose of these activities is to add a more diverse set of inside activities in order to more thoroughly investigate any confusion that can occur when classifying. More inside activities also cause the tree created to be larger, which can allow a better analysis of the tree structure's performance gains. The activities included in this set are:

- **StandHorizShoulderCircles**[*cyclic*] - While standing with the arms extended, the subject swings their arms forward and back in unison keeping them parallel to the floor. The subject is asked to swing a maximum of 45°, and to perform one rotation in 2 seconds.
- **StandPutDishesAway**[*cyclic*] - While standing, the subject acts as if grabbing a dish from a table and placing the dish in an overhead cabinet.
- **StandButtonShirt** - While standing, the subject acts as if buttoning three buttons; one at belly button, one in the sternum, and one on the neck.
- **SitComputerUsage** - While sitting, the subject acts as if typing on a computer, then reaches with right hand and use a mouse for about two seconds, then returns to typing position.
- **SitVertShoulderCircles**[*cyclic*] - While sitting with the arms extended, the subject swings their arms forward and back in unison keeping them perpendicular to the floor. The subject is asked to swing a maximum of 45°, and to perform one rotation in 2 seconds.

- **LeftShoulderRotation** - This activity is a vertical mirror of **RightShoulderRotation**.
- **LeftBrushHair**[*cyclic*] - While standing, the subject makes a motion as if they are brushing their hair with their dominant hand. If the subject is right handed, this activity is mirrored.
- **LeftBrushTeeth**[*cyclic*] - While standing the subject makes a motion as if they are brushing their teeth with their dominant hand. If the subject is right handed, this activity is mirrored. For this activity two cycles were extracted such that the length is consistent with the other activities.

Finally, ten subjects also performed the following seven outside activities, which are only inserted into the observed partition.

- **StandVertShoulderCircles**[*cyclic*] - While standing with the arms extended, the subject swings their arms forward and back in unison keeping them perpendicular to the floor. The subject is asked to swing a maximum of 45°, and to perform one rotation in 2 seconds.
- **StandArmsBySide** - The subject is asked to stand with their arms by their side
- **StandComputerUsage** - While standing, the subject acts as if typing on a computer, then reaches with right hand and use a mouse for about two seconds, then returns to typing position.
- **SitHorizontalShoulderCircles**[*cyclic*] - While sitting with the arms extended, the subject swings their arms forward and back in unison keeping them parallel to the floor. The subject is asked to swing a maximum of 45°, and to perform one rotation in 2 seconds.

- **SitRightAnswerPhone** - The subject sits as if both arms are resting on a desk. The subject then reaches out to pick up a phone and brings the phone to their ear.
- **SitArmsBySide** - The subject sits with their arms by their side
- **SitButtonShirt** - While sitting, the subject acts as if buttoning three buttons; one at belly button, one in the sternum, and one on the neck.

The classification criteria, T, is computed using the chi-square distribution at a confidence interval of 95%. Among a set of activities $\mathbf{s}$, an observation O is classified as activity $a \in \mathbf{s}$ if the following criteria are true

$$\begin{aligned} &M(O,\ a) < T \text{ and,} \\ &M(O,\ a) \leq M(O,\ b)\ \forall\ b \in \mathbf{s}, \end{aligned} \tag{5.3}$$

where $M(\cdot, \cdot)$ is the function to compute the score of a match, as defined in Equation 4.19.

When using the flat structure to classify observations of inside activities, a successful classification occurs when the observation is classified to the respective archetype. If this does not occur for an observation, it is an instance of failure by the classifier. There are two types of failures can occur which can result in an inside observation not being matched to an activity.

- **Confusion** - An observation is classified as the wrong activity.
- **Unknown Classification** - An observation is classified as belonging to no activity.

Section 6.3 presents the rates at which the inside observations were classified to the set of inside activities using the flat structure. When classifying the outside observations on the flat structure, a successful classification occurs when the observations is classified as unknown, ensuring that the classifier does not over-fit to the inside activities. Section 6.4 presents the classification rates of the outside observations on the flat structure.

When using a tree the same criteria above is applied multiple times recursively, effectively changing the set of activities being matched to as the classification moves inside branches along the tree. If a node in the current set satisfies Equation 5.3 and is a branch node, the set of child nodes are considered as the new activity set. This process is repeated until a leaf activity is selected, or no activity in the set can be selected. Like when classifying on the flat structure, an observation of an inside leaf activity should be classified as that leaf activity, otherwise it is a point of confusion by the classifier. Section 6.5 and Section 6.6 presents the same inside and outside observation sets used above being classified on the tree structure. For both sets the tree is shown to improve performance and allow for a partial classification to be done when an observation is classified as unknown.

Chapter 6

Results

This chapter discusses results gathered from the experiments detailed in Chapter 5. Section 6.1 illustrates why the `minBoundQuat` algorithm is preferred to a more classical approach of using the mean and variance. Section 6.2 demonstrates how the archetypes grow during the iterative building of the example set. Section 6.3 demonstrates the ability of the MUAC to classify observations of the inside activities. Of all the inside observations, a 84.8% rate of classifying the observation to the correct activity was achieved. Section 6.4 demonstrates the the MUAC classifying the outside observations to the inside activities. Of all the outside observations, a 69.1% rate of classifying the observation as unknown was achieved. Section 6.5 provides the tree structure built for the inside set of activities and its effect on the classification scheme. Using the tree structure reduces the computations required by 35.1% and has no effect on the accuracy of classification. Finally Section 6.6 demonstrates how information can be gained from an outside observation classifying to a specific branch node in the tree structure.

6.1 Archetype Computation

To demonstrate the benefit of using the minimal bounding circle to construct the activity archetype, the approach is compared to a mean and variance used in previous systems [4]. First the `minBoundQuat` algorithm is used to construct the set of activities defined in Section 5.3. With these archetypes and the iterative approach defined in Section 5.1.3, the min, mean, and max match score of the example set is recorded.

Next the `minBoundQuat` algorithm is replaced with an approach that uses the statistical mean to compute the continuous curve and statistical variance to compute the weights. The archetypes are re-created with this modified method and the min, mean, and max match score of the example set is again recorded.

When using the `minBoundQuat` approach the score of the examples to the archetype is found to be always less than one, as demonstrated in the third column of Table 6.1. The `minBoundQuat` algorithm creates an archetype where the distance of the maximum variation in the example set to the archetype is exactly one.[1] Since it is not required, or expected, that a single example always be a maximum variation for its entire length, no single example matched to the archetype with a score of exactly one.

When using a mean/variance approach, the match score of the examples to the archetype is difficult to predict, as demonstrated in the last column of Table 6.1. The mean/variance approach results in an archetype where the distance to the maximum variation of the example set is not constant. This is due to both the mean and variance being influenced by commonly recurring aspects of an activity. This unpredictability does not exist when using the `minBoundQuat` approach, illustrating the benefit of that approach as the basis for the weights.

[1] This score is achieved by integrating the maximum variation to the distance metric η.

Activity	minBoundQuat Approach			mean/variance Approach		
	Min	Mean	Max	Min	Mean	Max
LeftStep	0.30	0.52	0.90	0.85	2.40	13.16
LeftStepUp	0.36	0.55	0.86	0.84	3.09	17.76
ShoulderCircles	0.28	0.59	0.92	0.86	2.55	6.05
Rt.ShoulderRot	0.16	0.60	0.99	1.23	3.29	12.52
BrushTeeth	0.41	0.71	0.94	1.29	2.62	6.08
BrushHair	0.31	0.70	0.93	0.58	2.33	6.64
Driving	0.24	0.63	0.96	0.72	3.23	10.67
Eating	0.29	0.51	0.94	0.73	2.67	8.14

Table 6.1: Minimum, mean, and maximum match score of all observations when using the `minBoundQuat` approach introduced in this thesis and a classical mean/variance approach.

6.2 Example Set Size Convergence

To demonstrate the iterative archetype building process discussed in Section 5.1.3, each iteration recorded the impact of each new example and the mean match score to the observed partition. In general, the mean score decreased as more examples were added, a behavior demonstrated by the stair plots in Figure 6.1. As more examples are added, the archetype contains more variations of the activity. More variations in turn cause larger weights and reduces the match score between the observations and activity. For both Driving and RightBrushHair activity a significant reduction in the mean error occurred early in the iterative creation when the new example had a large impact on the example set. This indicates that the new variations the example added were important to appropriately represent variations in the activity. This property is not universally true, i.e., an example that exhibits a large impact does not have to cause a resulting reduction in the mean error. As introduced in Section 4.1.1, the distance metric on the archetype is based on the maximum across each dimension, the variation that causes a large impact on the example set can occur on a dimension already deemed unimportant in the distance metric. The mean error also does not have to be monotonically decreasing as more variations are added into the archetype. This is due to the imprecise nature of the regular expression, and because a new variation can

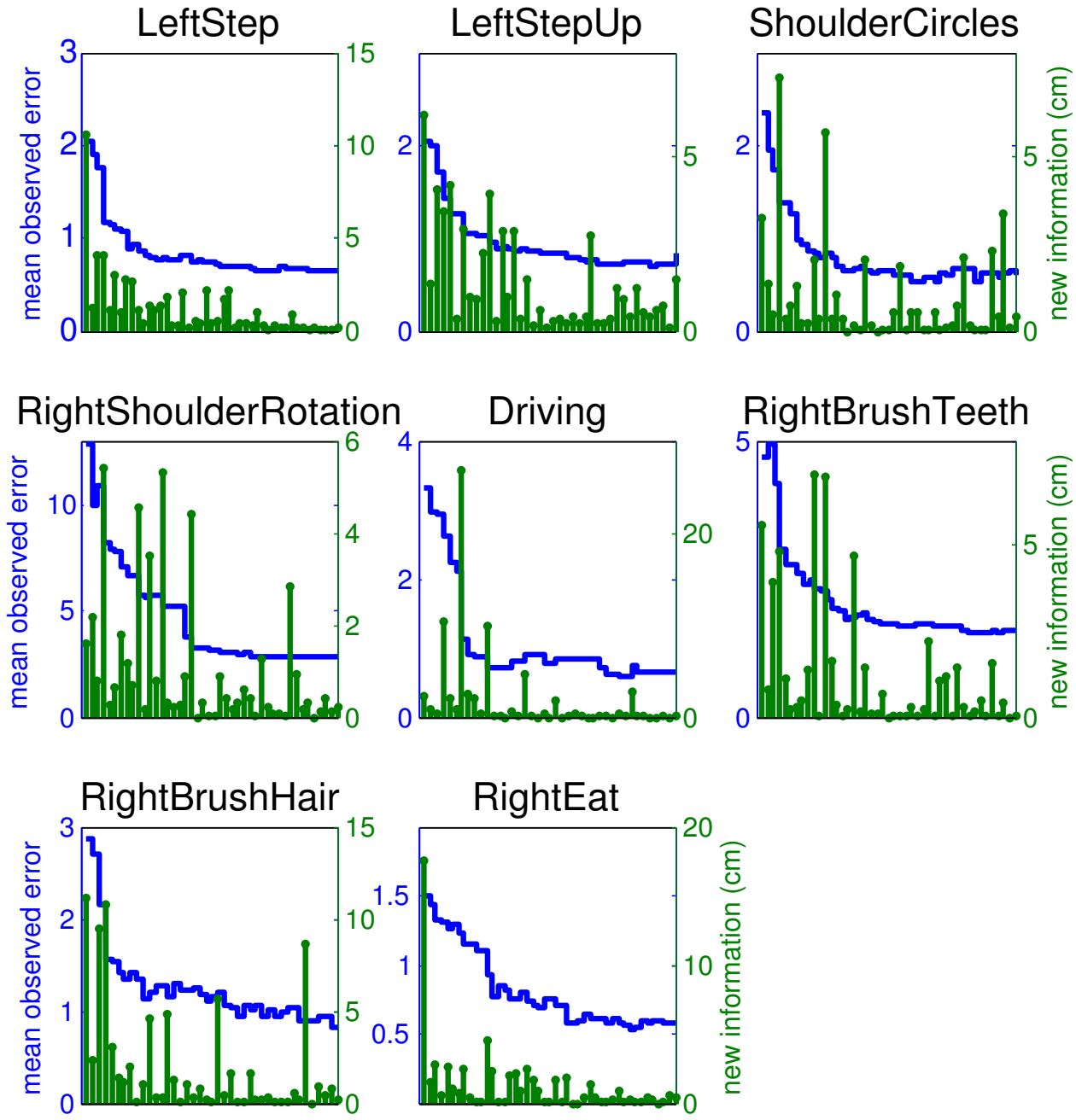

Figure 6.1: The mean match of observations shown by the blue stair plot and the amount of new variations added as the example size increases shown by the green stem plot for the activities used in this thesis. In each plot, a value to the right is a new example added to the example set. The impact is converted into a measurable dimension by applying the normalized difference onto an average human male height of 175.76 cm (69.2 in).

cause the archetype to move away from some specific examples (but still be contained inside the weights).

In general, the examples that add the most variations occur early on in the construction process. When the example set is small, a new example can make a larger impact because the variations it contains has not been introduced yet. As discussed above, these early large-impact additions often also cause a reduction in the mean match score. Occasionally, toward the end of the example set construction, a new example will be added that makes a large impact and has little change in the mean match score. Such behavior was seen toward the end of RightShoulderRotation and RightBrushHair activities. While these variations are new to the example set, the variations occur on a dimension that does not carry much significance in the archetype; and thus have little effect on the mean match score. The spike that occurred towards the end of RightShoulderRotation activity was caused by a variation on the right thigh. No match score of a RightShoulderRotation observation was determined by the right thigh for the steps before or after the example being added. Therefore, the new variation added by the example had little effect to the mean observation score for the step.

6.3 Inside Activity Recognition on Flat Structure

The performance of the MUAC at classifying on the flat, non-tree structure is determined by computing the match of the observations of the inside activities to the inside activities' archetype. See Section 5.3 for a detailed discussion on how these sets are made and what activities are recorded for each set. The rate of classifying observations to each inside activity is shown in Table 6.2. The number of overlapping dimensions between the inside activities are shown in Table 6.3. This table is created during the tree generation by Algorithm 2

Observations \ Archetypes	LeftStep	LeftStepUp	ShoulderCircles	Rt.ShoulderRot	Driving	RightBrushTeeth	RightBrushHair	RightEat	LeftShoulderRot	LeftBrushTeeth	LeftBrushHair	StandHorizSh.Circles	StandPutDishesAway	StandButtonShirt	SitComputerUsage	SitVertSh.Circles	Unknown
LeftStep	92.0	6.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.3
LeftStepUp	0	94.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.4
ShoulderCircles	0	0	96.0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.0
RightSh.Rotation	0	0	1.4	95.8	0	0	0	0	0	0	0	0	0	0	0	0	2.8
Driving	0	0	0	0	80	0	0	17.3	0	0	0	0	0	0	0	0	2.7
RightBrushTeeth	0	0	0	0	0	48.6	44.3	0	0	0	0	0	0	0	0	0	7.1
RightBrushHair	0	0	0	0	0	4.6	80.3	0	0	0	0	0	0	0	0	0	15.1
RightEat	0	0	0	0	7.4	0	0	89.7	0	0	0	0	0	0	0	0	2.9

Table 6.2: Confusion matrix when classifying on the flat structure. Each cell is the percentage that the row's activity as an observation gets confused as the column's activity archetype.

	LeftStep	LeftStepUp	ShoulderCircles	Rt.ShoulderRot	Driving	RightBrushTeeth	RightBrushHair	RightEat	LeftShoulderRot	LeftBrushTeeth	LeftBrushHair	StandHorizSh.Circles	StandPutDishesAway	StandButtonShirt	SitComputerUsage	SitVertSh.Circles
LeftStep	9	9	6	7	3	7	7	2	7	7	7	6	6	7	3	2
LeftStepUp	9	9	7	7	6	6	6	4	8	6	7	6	7	7	3	2
ShoulderCircles	5	4	9	6	3	6	6	1	5	6	6	9	5	5	1	5
RightShoulderRotation	8	8	5	9	4	7	7	1	6	6	6	5	5	7	4	1
Driving	4	5	5	5	9	5	4	9	5	5	4	4	7	6	9	7
RightBrushTeeth	7	6	6	7	6	9	9	3	6	5	5	6	5	6	3	2
RightBrushHair	7	7	7	8	5	9	9	3	6	5	5	7	5	6	4	2
RightEat	3	6	3	4	9	4	5	9	3	2	2	3	2	5	9	7
LeftShoulderRotation	7	7	5	6	4	6	6	2	9	7	7	5	6	7	4	1
LeftBrushTeeth	4	6	6	6	5	5	5	3	7	9	9	6	6	6	4	2
LeftBrushHair	6	8	7	6	5	5	5	3	8	9	9	7	5	6	3	3
StandHorizontalShoulderCircles	6	5	9	4	3	5	6	1	5	5	5	9	5	5	1	4
StandPutDishesAway	2	2	5	4	3	5	5	1	5	5	5	5	9	5	1	1
StandButtonShirt	6	7	5	8	5	6	6	4	7	7	6	5	6	9	5	1
SitComputerUsage	3	4	1	4	9	3	3	9	3	2	2	3	3	5	9	6
SitVerticalShoulderCircles	2	3	3	2	5	3	2	5	1	1	1	5	1	1	5	9

Table 6.3: Overlapping dimensions between each activity. Each cell indicates the number of dimensions that the row's activity overlaps when matched to the column's activity. The maximum possible overlap is 9 dimensions.

in Section 4.2 used in the next section; however, the table contains information about the inside activities useful in this section.

An overall success rate of 84.8% was achieved using the MUAC to classify the inside observations to the inside activities. Unknown classification occurred in 5.06% of the inside observations, attributing to 33.33% of the failures. The low unknown classification rate demonstrates the user-independence of MUAC since subjects used to generate the activities were separate from the subjects used as observations. Confusion occurred in 10.12% of the

observations, attributing to 66.67% of the total failures. Of the confusion that occurred in the experiment, 99.83% percent of it was caused by activities with a full nine dimensions of overlap (see Table 6.3). This high success rate demonstrates that the MUAC is capable of accurately distinguishing between activities that are well separated by 1 or more dimensions.

Of the activities with a full nine dimensions of overlap, three distinct sets of similar activities will now be investigated. The LeftStep and LeftStepUp activities are a pair of activities that share a full nine dimensions of similarity. The MUAC was capable of classifying an observation of these two activities for 92% of LeftStep observations and 94.6% of LeftStepUp observations. This correct classification occurred because the method was able to identify and use the difference of movement in the left shin, which must lift up during the LeftStepUp activity.

Driving, RightEat, and SitComputerUsage are three activities which also all overlap with nine dimensions. The difference between these activities is due to the outstretched arms of driving, and the right arm being raised to the mouth during eating. The MUAC was able to correctly classify observations of Driving 80% of the time and of RightEat 89.7% of the time. Driving was confused as RightEat at a rate of 17.3%, a rate significantly higher than other confusion seen. This is because some subjects held the steering wheel close to their body, the resulting pose was very similar to poses in the eating motion, causing a miss-classification.

RightBushHair and RightBrushTeeth are the final pair of activities which overlap with each other in nine dimensions. For observations of RightBrushTeeth, confusion to RightBrushHair occured at a rate of 44.3%, this confusion failure alone accounted for 35.6% of all failures in the experiment. No instruction was given to the subject on how to brush their teeth, among the example set there were a variety of angles in which subjects held their elbow ranging from straight out their shoulder, to subjects who pinned their elbow against their side. This

variation caused the weights of the RightBrushHair activity to be large and overwhelm the similar RightBrushTeeth activity.

6.4 Outside Observations on Flat Structure

The performance of classifying the outside observations defined in Section 5.3 on the flat structure is shown in Table 6.4. Overall, a 69.1% success rate was achieved at classifying an observation of an outside activity as unknown. 45.3% of the error was due to StandVerticalSoulderCicles observations being classified into the ShoulderCircles activity. In the ShoulderCircle example set, there were examples of subjects rotating their arms in a variety of ellipsoids. Because the activity archetype expresses all variations in the example set, all elliptical performances, including narrow performances of ShoulderCircles, are represented by the archetype. Therefore, StandVerticalShoulderCircles observations, which in essence are an extremely narrow ShoulderCircle activity, classify closely to the ShoulderCircle activity. This behavior also explains why observations of the outside activity SitHorizontalShoulderCircles matched to the inside activity SitVerticalShoulderCircles, which attributed to 20.4% of the outside observations that were classified to an activity. The examples of SitVerticalShoulderCircles contained a variety of maximum angles in of the activity. When classifying observations of SitHorizontalShoulderCircles, the error caused by the orthogonal movement was not enough to overcome the large angle variations allowed by SitVerticalShoulderCircles.

Observations	LeftStep	LeftStepUp	ShoulderCircles	Rt.ShoulderRot	Driving	RightBrushTeeth	RightBrushHair	RightEat	Left.ShoulderRot	LeftBrushTeeth	LeftBrushHair	StandHorizSh.Circles	StandPutDishesAway	StandButtonShirt	SitComputerUsage	SitVert.Sh.Circles	Unknown
StandVerticalSh.Circles	0	0	98	0	0	0	0	0	0	0	0	0	2	0	0	0	0
StandArmsBySide	0	0	0	2.0	0	0	0	0	16.0	0	0	0	0	0	0	0	82.0
StandComputerUsage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
SitHorizontalSh.Circles	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	94.0	6.0
SitRightAnswerPhone	0	0	0	0	24.0	0	0	22.0	0	0	0	0	0	0	0	0	54.0
SitArmsBySide	0	0	0	0	10.0	0	0	0	0	0	0	0	0	0	10.0	0	40.0
SitButtonShirt	0	0	0	0	0	0	0	6.0	0	0	0	0	0	0	0	0	94.0

Archetypes

Table 6.4: Confusion matrix when classifying outside activities on the flat structure. Each cell is the percentage that the row's activity as an observation gets confused as the column's activity archetype.

6.5 Inside Activity Recognition on Tree

With the same inside set used above, Algorithm 2 in Section 4.2 is applied in order to find the tree structure of the inside activities. The overlap, defined in Section 4.2, between each activity during the algorithm and shown in Table 6.3. The resulting tree, shown in Figure 6.2, found a total of 12 branch nodes. As the dimension constraint in Algorithm 2 decreased, activities with less overlap were grouped together under a branch node. For example, the last grouping, which occurred at one dimension, finally allowed the grouping of the sitting and standing activities, which share only one dimension of the torso pointing up.

The advantage to building the tree is to reduce the number of matches that must be computed for each observation. Table 6.5 gives the number of regular expression matches that must be computed to arrive at each leaf activity, while Table 6.6 gives the number of matches that must be computed to arrive at each branch node. When matching the inside set on the observed partition an average of 10.37 matches had to be computed, a 35.1% improvement compared to the flat structure required match computation of 16 activities per observation.

When using the tree structure results in the same 91.6% classification accuracy when using the flat structure. The confusion when using the tree structure (Figure 6.3) is not affected

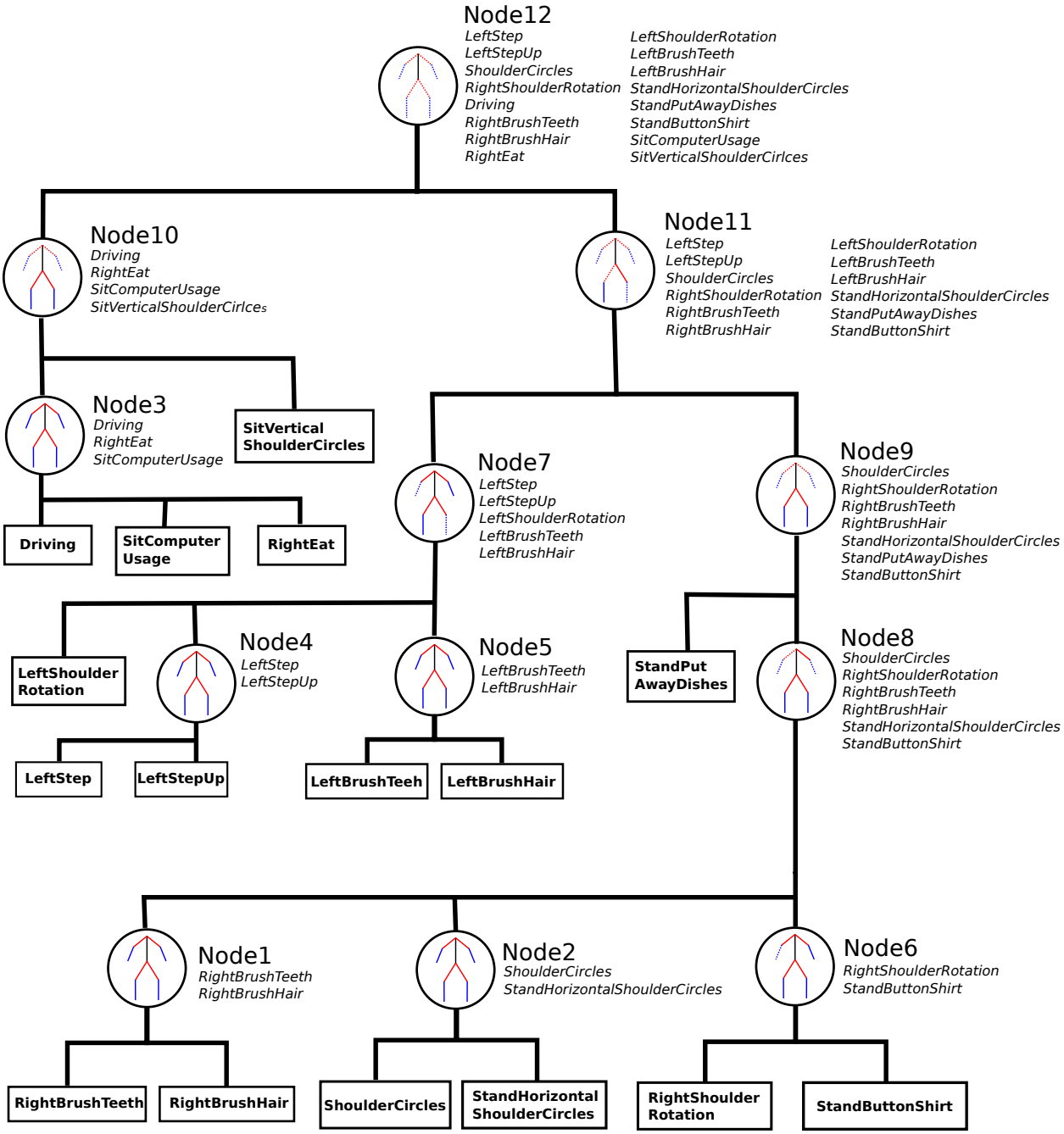

Figure 6.2: The tree structure generated by the inside activities. Branch nodes are marked with a circle while leaf nodes are marked by a square. A solid segment on the stick figure inside each branch node indicates that dimension was used to create the node.

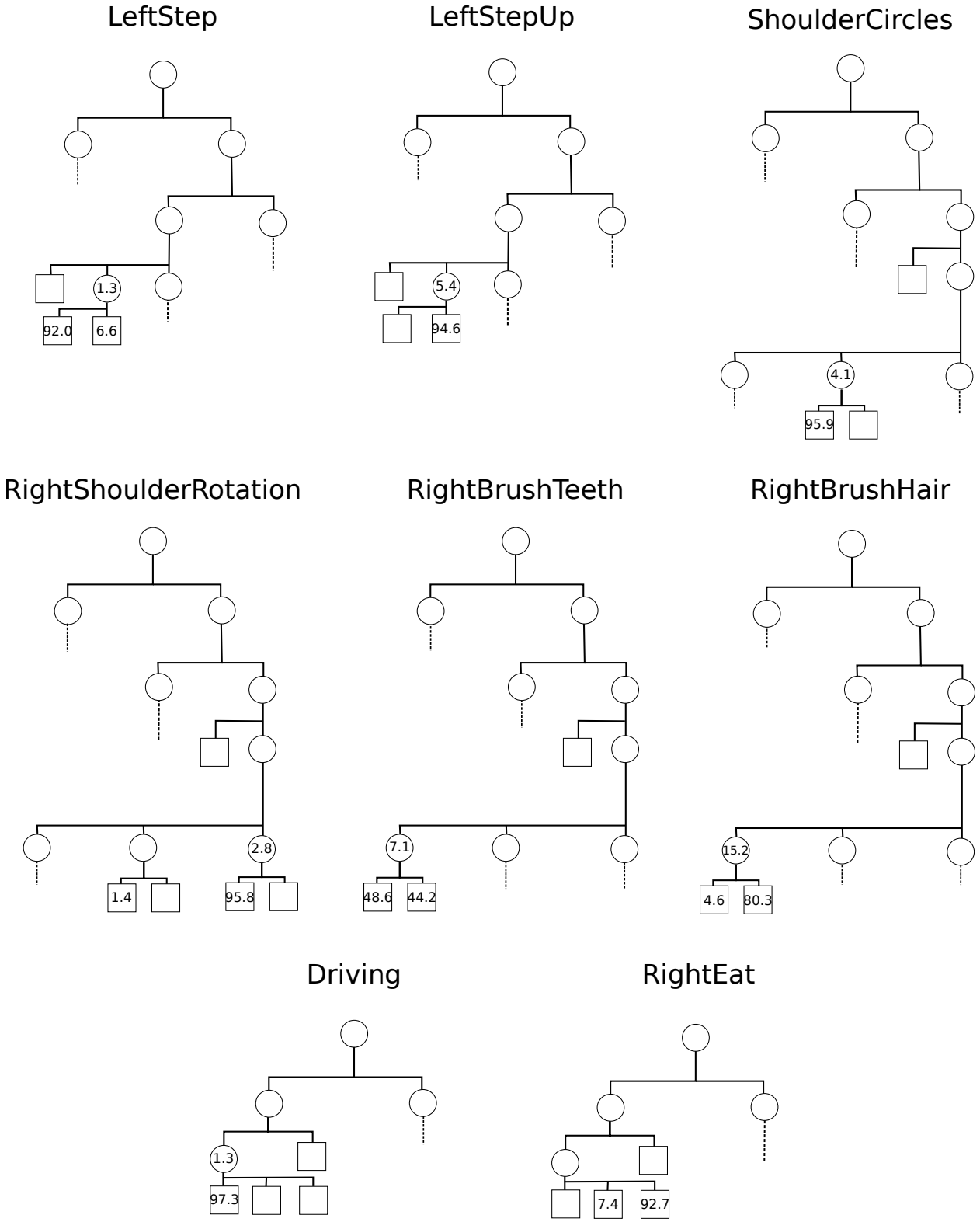

Figure 6.3: Detail on where each inside observation classified to on the tree structure. The number inside each node indicates the percentage of observations which were classified as that node. If no number is given, then no observations were classified. Portions of the tree were omitted for readability if no observations were classified to a parent.

Activity	Computations	Activity	Computations
LeftStep	10	LeftStepUp	10
ShoulderCircles	12	RightShoulderRotation	9
LeftSoulderRotation	8	RightBrushTeeth	12
RightBrushHair	12	Driving	8
Eating	8	StandHorizShoulderCircles	12
StandPutDishesAway	6	StandButtonShirt	9
SitComputerUsage	8	SitVertShoulderCircles	5
LeftBrushHair	10	LeftBrushTeeth	10

Table 6.5: Number of matches that must be computed to reach each leaf activity on the tree defined in Figure 6.2.

Branch Node	Computations	Branch Node	Computations
Node 12	1	Node 11	3
Node 10	3	Node 9	5
Node 8	7	Node 7	5
Node 6	10	Node 5	8
Node 4	8	Node 3	5
Node 2	10	Node 1	10

Table 6.6: Number of matches that must be computed to reach each branch node on the tree in Figure 6.2.

compared to the flat structure. This is to be expected because the tree does not modify the activity archetypes, and therefore does not affect the confusion.

When using the tree structure, inside observations that are not classified as any leaf activity can still classify to a node inside the tree. This partial classification can be used be used to determine some characteristics of the observation. For example, the observations of RightBrushHair activity that were classified as unknown on the flat structure were classified instead to Node8 (ShoulderCircles, RightShoulderRotation, RightBrushTeeth, RightBrushHair, StandHorizontalShoulderCircles, StandButtonShirt) on the tree structure. For both structures, the observations contained some variation not seen in the example set of RightBrushHair. These variations inhibited these observations classifying to the correct leaf activity. By looking at the dimensions that caused Node8 to form, it can be determined that the variation is from either the left upper arm, the right forearm, or the left forearm.

6.6 Outside Activity Recognition on Tree

To demonstrate the classifier's ability to partially classify an observation that is not known by the classifier, the same outside observations used in Section 6.4 are classified to the tree structure in Figure 6.2. The classification, visualized in Figure 6.4, achieved the same 69.1% success rate at classifying an unknown observation to a branch node as the flat structure classifying the observation as unknown (Section 6.4). With the tree structure, an average of 8.6 matches were computed per observation. This number of matches is lower compared to the inside observations because more outside observations are able to stop classifying at branch nodes and do not continue further down into the tree.

Similar to Section 6.5, the branch node classification of an an observation can be used to gain more information on the observation compared to the unknown classification given by the flat structure. For example, the SitButtonShirt observations were classified to Node3 (Driving, RightEat, SitComputerUsage) 94% of the time. This branch node classification indicates that the SitButtonShirt observations are of a subject sitting down in a chair with their arms in front of them, a characteristic shared between the activities used to create Node3. Another example is the StandArmsBySide observations, which were classified to either Node4 (LeftStep, LeftStepUp), Node7 (LeftStep, LeftStepUp, LeftShoulderRotation, LeftBrushTeeth, LeftBrushHair), or Node6 (RightShoulderRotation, StandButtonShirt). It could be determined that the observations were of the subject's body being upright and the right upper arm down by the subjects side, since that is the commonly occurring aspect of the classifications. It may seem peculiar that some of the observations of StandArmsBySide classified to Node4, a branch node created by two stepping activities. The stepping activities in Node4 both start and stop with the subject standing with their arms by their side. Because of this, StandArmsBySide observations match very close to Node4, especially at the beginning and end of the activity, causing some observations to be classified to Node4.

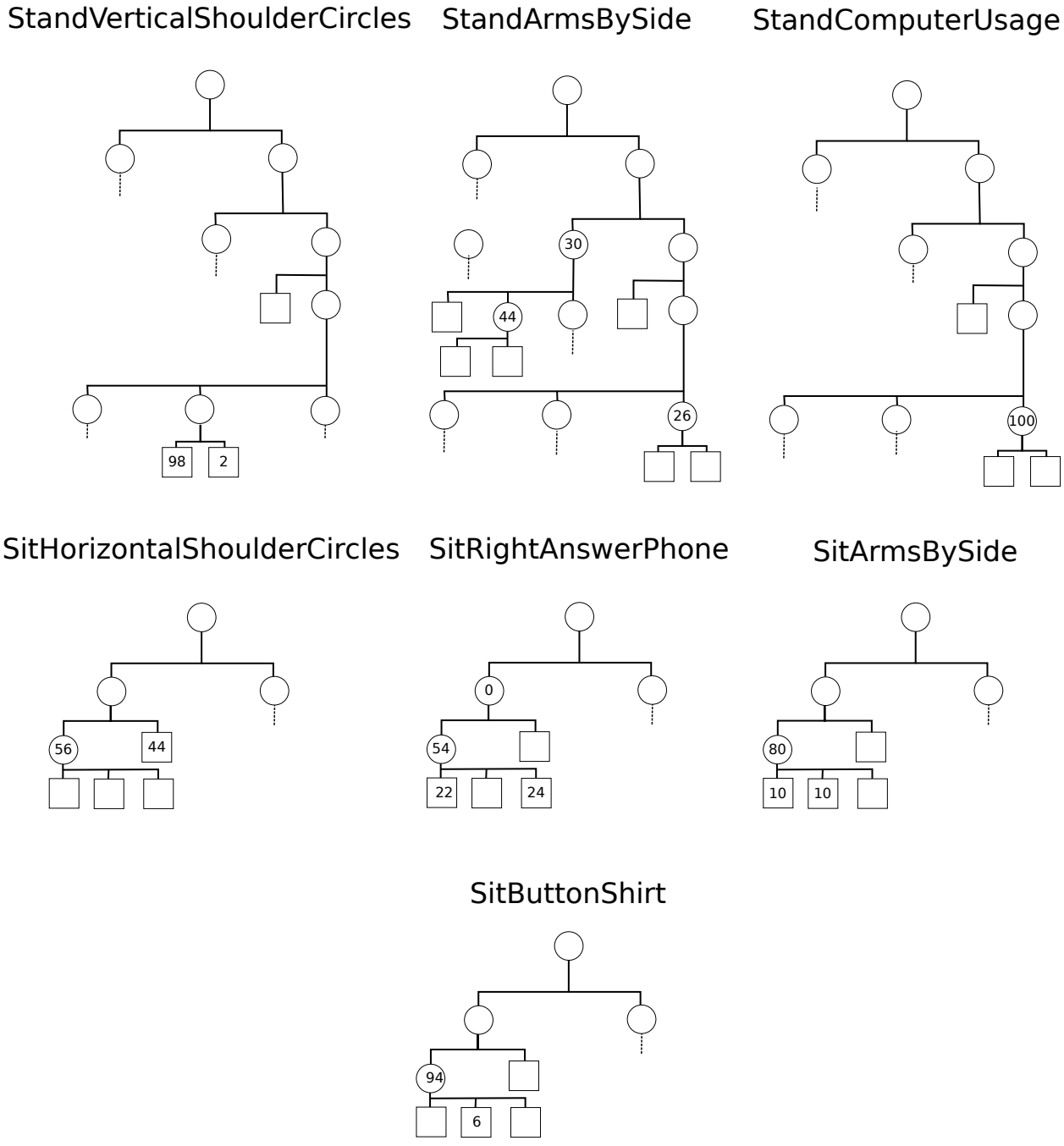

Figure 6.4: Detail on where each observation of the outside activities classified to in the tree structure. The number inside each node indicates the percentage of observations which were classified as that node. If no number is given, then no observations were classified. Portions of the tree were omitted for readability if no observations classified to a parent.

Chapter 7

Conclusion

This thesis introduced the MUAC, an activity classification algorithm targeted for deployment on a wearable garment. The method is based on an abstract model of the human body, allowing the system to be independent of any specific sensor set that can express the users pose with a model. From a set of examples of an activity being performed, a single representation is constructed that expresses the variations of the activity. The example set includes different subjects performing the activity differently, allowing the system to be independent of any specific user. A regular expression is constructed from the single representation and a string matching algorithm is used to compute the match of an observation to the activity. String matching algorithms have been shown to be very efficient at sequence alignment. Finally, using the activities to be classified, a tree is generated, which reduces the task of matching an activity from a linear search to a tree search. The tree allows the MUAC method to be scalable as additional activities are added into the classifier.

A user study was performed using the Microsoft Kinect to record the subject's motion. Twenty five participants recorded a core eight everyday activities, of the twenty five subjects, ten subjects recorded an additional nine activities. These are the activities known by the

classifier and were constructed from ten randomly chosen subjects. A 84.8% success rate of matching an observation to an activity inside the classifier was achieved. Ten subjects out of the twenty five participants also recorded seven activities whose activity representation was not constructed. A 69.1% success rate of *not* matching one of these observations to an activity known by the classifier was achieved.

The MUAC excels at identifying variations in order to ignore unimportant aspects of an activity. Most of the failure seen in this thesis demonstrates a weakness of using the MUAC method. If the activity contains significant variations on how to perform that activity, the resulting representation will be large, possibly too large in which it may engulf other activities. When using MUAC it is important to identify any significant variations in performing an activity and consider splitting the activity into multiple parts. For example, in the experiments in this thesis, it may of been advantageous to create two distinct RightBrushTeeth activities, one with the arm extended out, and one with the arm pinned in. It would have also been advantageous to give stricter parameters on subjects performing the various ShoulderCircle activities. The wide range of angles performed by subjects meant the activity representation contained a lot of variations. The similar movement of other forms of shoulder circles were close enough to the large variations to cause the observation to match closely to the other forms.

7.1 Future Work

The future work includes to applying the algorithm introduced into this thesis into a wearable garment. This garment must be able to construct the body model from sensed values. With such a garment, sensor independence could be demonstrated by using two different sensor domains to construct the activities and record observations. A use case of sensor

independence is to use an accurate sensor domain, such as motion capture, to record examples and create each activity's representation. A less accurate sensor domain, which a garment is expected to be, can then be used as observations matching to the activities constructed from a more accurate source.

Bibliography

[1] R. Marchiando and M. Elston, "Automated ambulatory blood pressure monitoring: clinical utility in the family practice setting.," *American family physician*, vol. 67, no. 11, p. 2343, 2003.

[2] Danbury Hospital, "Holter monitor diary instructions," 2010. `http://www.danburyhospital.org/~/media/Files/Patient%20Education/patiented-english/pdf_Diagnostic/HolterMonitorDiaryInstructions.ashx`.

[3] West Health Institute, "About reflexion," 2013. `http://www.westhealth.org/institute/our-innovations/reflexion`.

[4] C. Tran and M. Trivedi, "3-d posture and gesture recognition for interactivity in smart spaces," *Industrial Informatics, IEEE Transactions on*, vol. 8, pp. 178 –187, feb. 2012.

[5] T. Stiefmeier, D. Roggen, G. Troster, G. Ogris, and P. Lukowicz, "Wearable activity tracking in car manufacturing," *Pervasive Computing, IEEE*, vol. 7, pp. 42 –50, April-June 2008.

[6] C3DServer, "C3Dserver Software," 2012. `http://www.c3dserver.com/`.

[7] Microsoft, "Kinect For Windows," 2012. `http://www.microsoft.com/en-us/kinectforwindows`.

[8] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the space of a human action," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 144–149, IEEE, 2005.

[9] A. Vahdatpour, N. Amini, and M. Sarrafzadeh, "On-body device localization for health and medical monitoring applications," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pp. 37–44, IEEE, 2011.

[10] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on*, pp. 82–89, IEEE, 2001.

[11] M. H. Ko, G. West, S. Venkatesh, and M. Kumar, "Using dynamic time warping for online temporal fusion in multisensor systems," *Information Fusion*, vol. 9, no. 3, pp. 370–388, 2008.

[12] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[13] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 17–24, IEEE, 2010.

[14] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, p. 2001, 1999.

[15] Y.-S. Chung, C. Lu, and C. Tang, "Efficient algorithms for regular expression constrained sequence alignment," in *Combinatorial Pattern Matching*, pp. 389–400, Springer, 2006.

[16] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, "Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach," in *Wearable Computers (ISWC), 2012 16th International Symposium on*, pp. 17–24, 2012.

[17] J. Farringdon, A. Moore, N. Tilbury, J. Church, and P. Biemond, "Wearable sensor badge and sensor jacket for context awareness," in *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pp. 107–113, 1999.

[18] K. Murao, T. Terada, A. Yano, and R. Matsukura, "Evaluating gesture recognition by multiple-sensor-containing mobile devices," in *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pp. 55–58, 2011.

[19] G. Gioberto and L. Dunne, "Garment positioning and drift in garment-integrated wearable sensing," in *Wearable Computers (ISWC), 2012 16th International Symposium on*, pp. 64–71, 2012.

[20] H. Bayati, J. del R Millan, and R. Chavarriaga, "Unsupervised adaptation to on-body sensor displacement in acceleration-based activity recognition," in *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pp. 71–78, 2011.

[21] N. Kern, B. Schiele, and A. Schmidt, "Multi-sensor activity context detection for wearable computing," in *Ambient Intelligence*, pp. 220–232, Springer, 2003.

[22] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 1, pp. 119–128, 2006.

[23] J. P. Varkey, D. Pompili, and T. A. Walls, "Human motion recognition using a wireless sensor-based wearable system," *Personal and Ubiquitous Computing*, vol. 16, no. 7, pp. 897–910, 2012.

[24] O. Banos, A. Calatroni, M. Damas, H. Pomares, I. Rojas, H. Sagha, J. del R Millán, G. Troster, R. Chavarriaga, and D. Roggen, "Kinect= imu? learning mimo signal mappings to automatically translate activity recognition systems across sensor modalities," in *Wearable Computers (ISWC), 2012 16th International Symposium on*, pp. 92–99, IEEE, 2012.

[25] J. Funda, R. H. Taylor, and R. P. Paul, "On homogeneous transforms, quaternions, and computational efficiency," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 3, pp. 382–388, 1990.

[26] "Kinect for windows human interface guidelines," tech. rep., Microsoft, 2012. `http://go.microsoft.com/fwlink/?LinkID=247735`.

[27] M. Musé, B. LeFew, and M. Shafiei, *Exercise for the chronic pain patient.*

[28] T. Ellenbecker, *Shoulder Rehabilitation: Non-operative Treatment.* Thieme Publishers Series.

[29] H. Dreyfuss, *Designing for People.* W. W. Norton, 2003.

Appendix A

Segment Lengths

The length of a segment on the body model is based upon normalized lengths of average human limb lengths. Table A.1 shows fully grown male and female average lengths of the human body as well as the lengths normalized to height [29]. Due to the little difference of the normalized lengths between male and female, a single set of parameters can be used to express both male and female normalized limb lengths. The shoulder and hip joints are offset from the body center by half the shoulder/hip breadth and half the torso height. This in effect means that each limb itself is offset from the center of the body by the offset of the shoulder/hip joint it connects to. These lengths have no bearing on the operation of MUAC, however; they are needed in order to convert a pose into a set of 3D positions, which is useful in order to display the pose.

	Male Length (in)	Female Length (in)	Male Norm	Female Norm	Together
Height	69.2	63.2	1	1	1
Torso	17.5	17.5	0.2529	0.2769	0.2649
Thigh	18	16.6	0.2601	0.2627	0.2614
Shin	16	15.1	0.2312	0.2389	0.2351
Upper Arm	12	12.2	0.1734	0.1930	0.1832
Forearm	10.4	9.7	0.1503	0.153	0.1519
Shoulder Width	11.5	10.8	0.1662	0.1709	0.1685
Hip Breadth	6.8	7	0.0983	0.1108	0.1045

Table A.1: Average and normalized lengths of human body

Appendix B

Quaternion Rotations

To rotate a Euclidean vector, v, by a quaternion, q, Algorithm 5 is applied, making use of quaternion conjugation and quaternion multiplication. To find the quaternion which rotates a Euclidean vector v_1 to another vector v_2, Algorithm 6 is applied, making use of quaternion cross product and quaternion dot product.

Algorithm 6 constructs a quaternion to represent the segments Y and Z rotation from the coordinate system it is connected to. No X axis rotation is represented, each dimension has two degrees of freedom.

This approach gives the rotation of the fore segment two degrees of freedom, a joint which physically only has one degree of freedom. This is necessary to prevent an ambiguous rotation that can arise when when using a model which assigns 3 degrees of freedom to the upper segment and 1 degree of freedom to the fore segment. Under that approach, if the fore segment extends directly out of the upper segment (a rotation of [1 0 0 0]), then there are infinite quaternions for the upper segment which will describe the same body orientation.

Constricting each segment to two degrees of freedom, the Y and Z axis, prevents this from happening. Insuring that each unique pose corresponds to a unique body orientation.

Algorithm 5 Rotate a vector by a quaternion

function RotateVector(v, q)
$qv_w \leftarrow 0$
$qv_x \leftarrow v_x$
$qv_y \leftarrow v_y$
$qv_z \leftarrow v_z$
$\overline{q} \leftarrow (q^*)(qv)(q)$ ▷ * indicates Quaternion Conjugation
$\overline{v}_x \leftarrow \overline{q}_x$
$\overline{v}_y \leftarrow \overline{q}_y$
$\overline{v}_z \leftarrow \overline{q}_z$
return $\overline{v}$
end function

Algorithm 6 Find the quaternion between two points

function VectorQuaternion(v_1, v_2)
$v \leftarrow v_2 - v_1$
$Halfv \leftarrow v$
$Halfv_x \leftarrow Halfv_x + \|v\|$
$Halfv \leftarrow \dfrac{Halfv}{\|Halfv\|}$
$\lambda \leftarrow v \times Halfv$ ▷ $\times$ indicates Vector Cross Product
$q_w \leftarrow v \cdot Halfv$ ▷ $\cdot$ indicates Vector Dot Product
$q_x \leftarrow \lambda_x$
$q_y \leftarrow \lambda_y$
$q_z \leftarrow \lambda_z$
return q
end function

Appendix C

Segment Positions

To determine a segment's position relative to the body's center, Algorithm 7 is applied on the pose. To get the position of an upper segment, the offset of the upper segment's shoulder/hip joint are added to the vector resulting from the upper segment's rotation. To get the position of a fore segment, a more complicated steps are required due to the rotation, and therefore position, of a fore segment being dependent on the rotation of the upper segment. The rotation of the fore segment, relative to the body, is the product of the fore segment's quaternion to the upper segment's quaternion. This new rotation results in the vector of the fore segment from the end of the upper segment in the body's coordinate system. To get *position* of the fore segment relative to the body, this vector is added to the position of the upper segment. Finally, to determine a segment's orientation relative to the environment, the vector resulting from Algorithm 7 is rotated by the torso's quaternion q_T.

Algorithm 7 Finding position of a given segment

```
function POSITION(Segment)
    QRot ← Segment.Q
    P ← Segment.Parent
    while P ≠ NULL do
        QRot ← QRotSegment.Q
        P ← P.Parent
    end while
    Vrot.x ← Segment.Length
    VRot.y ← 0
    VRot.z ← 0
    VRot ←Rotatevector(VRot, QRot)
    if Segment.Parent == NULL then
        VRot ← VRot + Segment.Limb.OffsetVector
    else
        VRot ← VRot+Position(Segment.Parent)
    end if
    return VRot
end function
```

Appendix D

Implementation Testing of MUAC

This section tests the implementation of the MUAC as discussed in Chapter 5. To test the hypotheses simple test consisting of elementary functions were performed. The use of elementary functions are used so that there exists an exact solution with an error of zero, which allows for the precision of the implementation to be measured. In addition, for many of the components, there is a specific example of the component operating on some of the activities defined in Chapter 5.

D.1 Cycle Detection

The implementation of the cycle detection component defined in Section 4.1.1 is demonstrated by finding the period of sinusoidal functions. The sinusoid is of the form

$$f(x) = sin(\lambda x + \phi). \tag{D.1}$$

A set of 100 curves bounded by $0 \leq x \leq \rho$ were constructed by randomly selecting $1 \leq \lambda \leq 10$, $0 \leq phi \leq 2\pi$, and $2 \leq \rho \leq 10$.[1] The cycle detection component of MUAC is used to compute the period for each sinusoidal. Among all 100 trials, the maximum difference between the period found by the cycle detection algorithm and the theoretical period was 6.6e-7. This is consistent with the optimizer used to solve the equation whose constraint tolerance is set to 1e-6.

D.2 Cyclic Sequence Alignment

The implementation of the sequence alignment component defined in Section 4.1.1 is demonstrated on cyclic curves by aligning a set of elementary functions in $\mathbb{R}$ space. Alignment is performed over a set of five curves in the form of

$$sin\left(\lambda x + \phi\right) \tag{D.2}$$

where λ and ϕ are random integer bounded by $1 \leq \lambda \cdot 100 \leq 1000$ and $0 \leq \phi \cdot 100 \leq 10000$. The curves in both aligned and unaligned states are shown in Figure D.1. The distance that each aligned curve is from the ideal aligned curve is found to be less than the constraint on the optimizer of 1e-2 (Table D.1). This shows that the implementation done for this thesis is capable of using the MUAC to align a set of cyclic examples.

[1] ρ in this case defines the number of periods. As stated in Chapter 4, at least 2 cycles are needed to find the period.

Function	α	ϕ	Distance from Ideal
$\sin(4.70t + 2.30)$	1.34	0.10	9.57e-04
$\sin(8.44t + 1.94)$	0.74	0.16	1.43e-03
$\sin(2.25t + 1.70)$	2.79	0.20	1.02e-03
$\sin(2.27t + 4.35)$	2.77	-0.22	1.47e-03
$\sin(3.11t + 9.23)$	2.02	0.00	9.30e-04

Table D.1: The alignment of five cyclic curves and the resulting distance from the average of all aligned curves

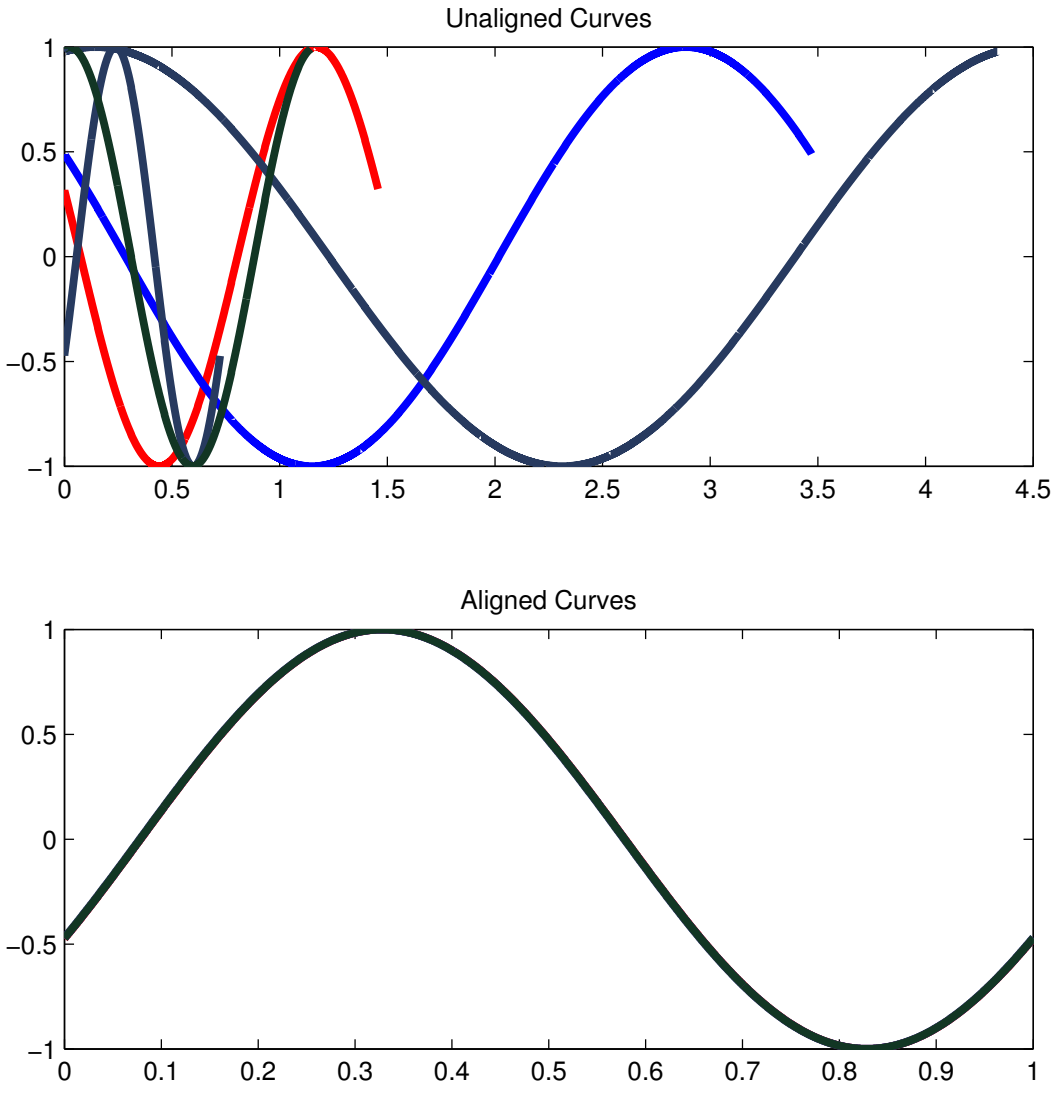

Figure D.1: Each cyclic curve before and after the alignment parameters are aligned.

D.3 Non-Cyclic Sequence Alignment

The implementation of the sequence alignment component defined in Section 4.1.1 is demonstrated on non-cyclic curves by aligning a set of elementary functions in $\mathbb{R}$ space. Alignment is performed over a set of five curves in the form of the piecewise function

$$\Lambda(\lambda x+\phi)=\begin{cases}(\lambda x+\phi)e^{(2-(\lambda x+\phi))}, & \text{if } x\geq\dfrac{1-\phi}{\lambda}\\ -((\lambda x+\phi)-2)e^{(2+((\lambda x+\phi)-2))}, & \text{if } x<\dfrac{1-\phi}{\lambda}\end{cases} \tag{D.3}$$

where λ and ϕ are random integer bounded by $100\leq\lambda\cdot 100\leq 400$ and $-700\leq\phi\cdot 100\leq -2000$. Note that the bounds placed on the parameters are arbitrary.. The curves in both aligned and unaligned states are shown in Figure D.2. The distance that each aligned curve is from the ideal aligned curve is found to be less than the constraint on the optimizer of $1e-2$ (Table D.2). This shows that the implementation done for this thesis is capable of using the MUAC to align a set of non-cyclic examples.

Function	α	ϕ	Distance from Ideal
$\Lambda(3.40t+-7.38)$	3.75	0.00	3.54e-03
$\Lambda(3.78t+-16.50)$	3.52	0.66	2.91e-02
$\Lambda(2.46t+-14.53)$	5.22	0.55	2.58e-03
$\Lambda(1.71t+-12.97)$	7.35	0.45	1.44e-02
$\Lambda(3.88t+-14.11)$	3.24	0.54	1.43e-02

Table D.2: The alignment of five non-curves and the resulting distance from the average of all aligned curves

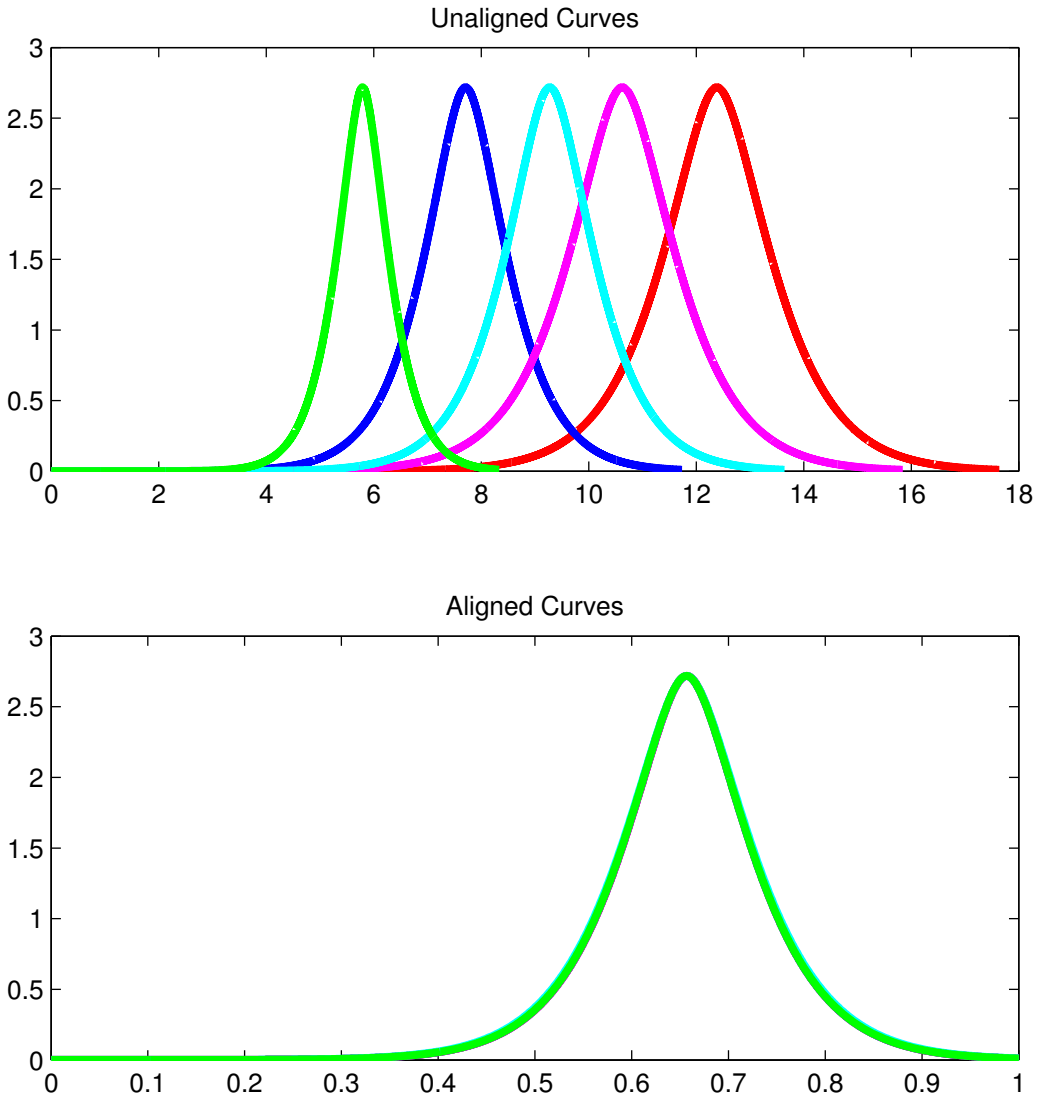

Figure D.2: Each non-cyclic curve before and after the alignment parameters are aligned.

D.4 Minimal Bounding Circle

The implementation of the minimal bounding circle algorithm defined in Section 4.1.1 to find an "ideal" quaternion is demonstrated by running the algorithm on a set of random rotations. A set of quaternions $\mathbf{q}$ are constructed which randomly deviate from "center" quaternion $q_c = [1\ 0\ 0\ 0]$ at a maximum angle of 45°. That is, q_j is allowed to be any rotation so as long as the rotation 45° or less away from q_c. This angle choice is arbitrary but must be less than 90° because any greater angle will give the minimal bounding circle algorithm to approximately pick either $[1\ 0\ 0\ 0]$ or $[-1\ 0\ 0\ 0]$.

With a set of random quaternions, q_a is computed by

$$q_a = minBoundCircle(\mathbf{q}) \tag{D.4}$$

and should be an approximation of q_c. Table D.3 shows the mean and standard deviation of the angle of q_a to q_c for 10000 $\mathbf{q}$ sets at a variety of sizes. To get a perfect q_a only two quaternions orthogonal to each other and at the same angle are required. This is exceptionally rare, therefore as more points are added into $\mathbf{q}$ more points close to the 45° edge are contained within the set, therefore the quaternion that defines the minimal bounding circle is closer to q_c. This shows that the `minBoundCircle` algorithm can correctly find the quaternion which minimizes the maximum distance among some quaternion set.

$\mathbf{q}$ Set Size	Mean Angle From Ideal	Std-dev From Ideal
2	22.8	10.2
20	7.5	3.8
200	2.41	1.25
2000	0.7	0.4

Table D.3: The resulting mean and standard deviation of the error at different set sizes

D.5 Alphabet Computation

The ability for the implementation to correctly construct an alphabet sequence is shown by using Euclidean distance and a piecewise function in $\mathbb{R}$ space to find the alphabet. Elementary functions are again used because the exact alphabet can be computed and compared to the output of the implementation. The piecewise function given by

$$\Phi(x) = \begin{cases} 0, & \text{if } x \leq 10 \\ x - 10, & \text{if } 10 < x \leq 20 \\ -x + 30, & \text{if } 20 < x \leq 30 \\ 0, & \text{if } 30 \leq x \end{cases} \tag{D.5}$$

The alphabet from $0 \leq x \leq 40$ is computed using a radius of one. It is therefore known that the theoretical characters should occur when $\Phi(x)$ is equal to 11, 13, 15, 17, 19, 23, 25, 27, and 29. The maximum error of the calculated alphabet(shown in Figure D.3) to these expected values was found to be $6.02e-6$, which is consistent with the optimizer used to solve the minimization in Equation 1 constrained to $1e-6$.

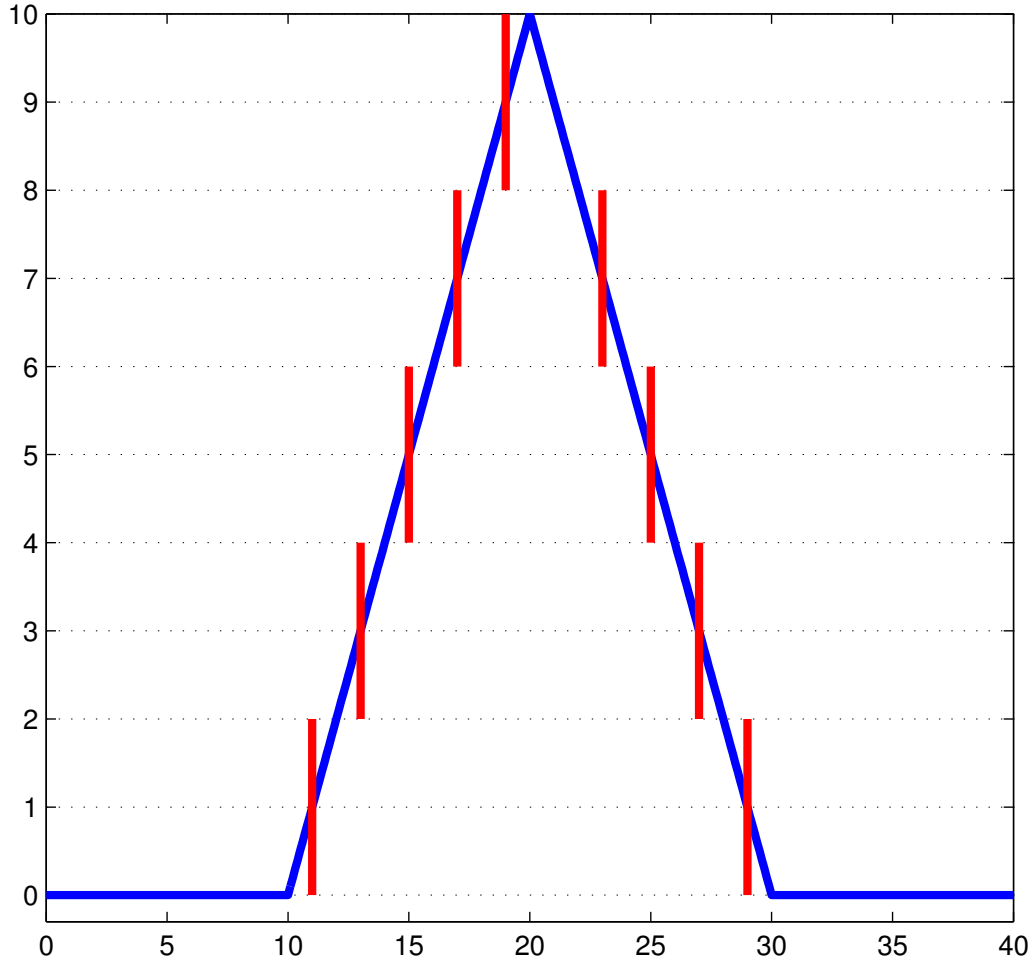

Figure D.3: Characters found for the given function using a radius of one. The vertical bars represent each character and the range on $\Phi(x)$ which the character is responsible for.