UCLA
UCLA Electronic Theses and Dissertations

Title
Recognition and Classification of the Wolf Motor Function Test Items using Multimode Sensor Fusion

Permalink
https://escholarship.org/uc/item/9n21974t

Author
Wang, Yan

Publication Date
2012

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Recognition and Classification of the Wolf Motor Function Test Items using Multimode Sensor Fusion

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Electrical Engineering

by

Yan Wang

2012

ABSTRACT OF THE THESIS

Recognition and Classification of the Wolf Motor Function Test Items

using Multimode Sensor Fusion

by

Yan Wang

Master of Science in Electrical Engineering

University of California, Los Angeles, 2012

Professor William J. Kaiser, Chair

Human motion monitoring and activity classification, specifically in the free-living environment, are becoming increasingly important as preventative, diagnostic and rehabilitative measures in health and wellness applications. Besides vision-based movement tracking, wearable sensors are leading a more and more important role in this area due to their miniature size, easy deployment, portability and capacity.

In contrast to gait analysis, wearable sensor-based evaluation of upper body activities is not well studied. The work in this thesis tends to explore a novel system for upper limb activity monitoring and classification. The system focuses specifically on the application of motion classification to a complex task of automating rehabilitation evaluation, such as the Wolf Motor Function Test. The presented system consists of a novel wearable motion sensor platform that

integrates accelerometers, gyroscopes and flex-sensors, and classification algorithms that convert motion data into an alphabet representation and form a string of primitives. String expressions are then derived for each test item and a regular expression based searching method is developed. We present results from the successful application of the proposed system to upper limb activity characterization in the context of the Wolf Motor Function Test.

The thesis of Yan Wang is approved.

Gregory J. Pottie

Lieven Vandenberghe

William J. Kaiser, Committee Chair

University of California, Los Angeles

2012

This thesis is dedicated to my parents, Weiming Wang and Chunya Wei, and my boyfriend, Qian Wang, for their understanding and spiritual support.

I would like to thank Professor William J. Kaiser, for his inspiration, guidance, encouragement and financial support.

I would like to thank Professor Gregory J. Pottie, and Professor Lieven Vandenberghe for their instructions.

I would like to thank Maxim A. Batalin and Xiaoyu Xu, for their help.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

On a worldwide scale, stroke is the second leading cause of death and it accounts for 9% of the 50.5 million deaths each year. Up to 2010, stroke is the No.4 cause in death and the leading cause of disabilities among adults in the United States [1]. Nationwide, more than 7 million individuals are affected by stroke every year, which is approximate to one person every 40 seconds [2]. According to the statistics provided by the University of Medicine & Dentistry of New Jersey, currently more than 4 million stroke survivors are still suffering from its after-effects [3].

The total cost of treatment and care for a stroke patient in the United States is estimated at $43 billion per year, among which the medical care and therapy account for $28 billion. Zoomed into an individual scale, the average cost of care for a patient up to 90 days after stroke is $15 thousand and 10% of patients spend even more. A breakdown analysis of the cost indicates that 16% of the amount is used for rehabilitation [3]. Considering the huge number related to the total cost, automated rehabilitation with application of novel technology has a high potential impact not only for expedient patient recovery, but also significant reduction in healthcare costs.

The Wolf Motor Function Test (WMFT), a time-based method to assess upper extremity motor function in adults with hemiplegia through a series of functional tasks is widely adopted in rehabilitation program [4]. The most common way to conduct the test is under the supervision of a therapist, who will manually time each test item. The estimated time of the whole process takes approximately 30 minutes [5]. Therefore, automation of the test process will largely reduce the demands of human resources in hospitals and rehab centers. At the same time, patients can also benefit from it in terms of reduced costs and more flexible testing schedules.

1.2 Aim, Objectives and Contributions

The ultimate goal of our research is to characterize human upper limb motions in free-living environments, which includes extracting meaningful activity segments from the infinity of daily motions, providing insight into joint-specific as well as total limb movements and evaluating upper extremity motor functions. The above problem is extremely complex due to the following complications: (1) Terminologies describing upper extremity activities are not concretely defined. For example, people hold various understanding of the activity named as ‘grasping’; (2) Increased degree of freedom of upper limb joints loosens the constraint of activity performance where within-subject motions can present numerous variations of a single activity under the same definition; (3) Performance evaluation is almost impossible as the difference between performance deficiency or inability and personal habits can hardly be recognized by a classification system without ground truth provided by subjects such that the system performance largely relies on subjects’ honesty.

To make the problem manageable, the presented system is targeted at recognizing and classifying a set of upper extremity activities extracted from the WMFT. The system will process

the data through two steps: (1) The data sequence is segmented into several smaller sessions, each of which represents a test item, an activity; (2) Each session is classified into a specific activity class where the overall activity set is predefined based on the fixed set of test items. To note that sensor data is usually streamed based on a certain sampling rate, time cost to accomplish each test item is recorded. Therefore, classification output will be sufficient for upper extremity performance evaluation, recalling the traditional timing method of conducting the WMFT.

This thesis includes the following contributions: (1) classification of a set of upper limb activities extracted from the WMFT; (2) architecture design for a wearable sensor platform with both inertial and flex sensors; (3) novel and intuitive method of activity decomposition into primitives; (4) application of regular expressions in the area of upper extremity activity recognition and classification.

1.3 Related Work

Advances in embedded technology have led to the proliferation of the wearable sensor platforms. The application of wearable motion sensing platforms to human activity classification promotes the wellness and healthcare objectives in assisted living, disease diagnosis and rehabilitation care. [6] proposed a novel machine learning and statistical technique to extract parameters from EMG sensors placed on the tibialis anterior and gastrocnemius muscles for human postural control system assessment. In [7], a novel hierarchical subject state classification and walking parameter computation system showed its prospect in the application of stroke patient rehab care.

Among all the data collection methods, employment of inertial sensors has been one of the most

popular approaches to record human motion details. [8] used a waist-mounted tri-axial accelerometer to detect falls and classify transitional movements. [9] proposed a primitive-based activity classification method using the output of a tri-axial accelerometer and a bi-axial gyroscope. Recently, flex sensor has attracted more attentions on its application of measuring physiological bending angles. [10] proposed a motion-based game control system where players' motion is detected from the flex sensor band worn on the elbow and the knee, and the flex sensor belt on the waist. [11] presented a novel flex sensor glove for measuring hand kinematics of a rhesus macaque performing a grasping task with 25 different objects.

Researchers have studied the upper limb motion estimation extensively. [12] proposed a Kalman filter based algorithm to estimate forearm movements. [13] estimated subjects' upper-limb orientation when the person is performing reaching tasks. However, classification method of clustering different upper extremity activities is still in demand. Otherwise, performance evaluation can be hardly applied.

The above investigation of the related work has proven the ability of inertial sensors in the application of human motion characterization. The addition of flex sensor supplies the information structure with one more dimension. The thesis is exploring a novel way to solve the recognition and classification problem encountered in the upper limb activity characterization community by proposing a new sensor fusion method.

Chapter 2

System Design

This chapter starts with an introduction of the system capacity and specifies several terminologies. Then it presents the hardware architecture followed by a description of the recognition and classification algorithm.

2.1 System Overview

The challenges of upper limb activity recognition and classification have been roughly described in Section 1.2. The key of characterizing upper extremity movements is to handle the diversity of hidden meanings behind activity descriptions and the infinity of activities that can be performed in the free-living environment. Constraints on the activity set and the range of performance variations are very necessary not only to simplify the system design but also highlight its application value as not all the motion segments are worthy of deep analysis. The presented system is focused on classifying a set of meaningful activities that can provide insight into the upper extremity motor functions of stroke patients. A good reference for the activity set selection is the WMFT. The application scenario of our system is set as a testing environment, which implies that subjects will perform each test item/activity once based on the testing rules explained in the WMFT.

The widely used version of the WMFT consists of 17 test items and some of them have very similar motion paths involving the same joint movements. For example, ‘forearm to table’ (subjects attempt to place forearm on a table by abducting at the shoulder) and ‘forearm to box’ (subjects attempt to place forearm on a box, 25.4cm tall by abducting at the shoulder) will make only very slight difference in the view of both inertial sensors and flex sensors. Also ‘flip 3 cards’ (subjects attempt to flip each card over using the pincer grasp), ‘turning the key in the lock’ (subjects attempt to turn a key 180 degrees to the left and right using pincer grasp while maintaining contact) and ‘fold towel’ (subjects grasp towels, fold its lengthwise and then use the tested hand to fold the towel in half again) can all be grouped into the same category of ‘wrist rotation’.

By merging the similar testing items and abstracting the common key motion components, 6 test items/activities are selected: (1) forearm to the table: subjects attempt to place forearm on a table by abducting at the shoulder; (2) extend elbow: subjects attempt to reach across a table by extending the elbow; (3) hand to table: subjects attempt to place involved hand on a table; (4) reach and retrieve: subjects attempt to reach across a table by using elbow extension and flexion; (5) lift: subjects attempt to lift a bottle and bring it close to his/her lips with a cylindrical grasp (6) flip: subjects attempt to flip a card over using the pincer grasp. Videos of demonstrating each testing item can be found in [14, 15]. When subjects are taking the compact ‘WMFT’ which consists of the six activities listed above, the system can delimit each activity segment from the whole motion sequence and classify them into the predefined 6 categories while retaining all the timing information. The order of the activities performed will not affect the system performance.

Through the thesis, there are several terminologies we will use and their definitions are specified

at the context of our application. (1) **Activity**: it refers to a test item included in the compact 'WMFT' and there are 6 activity classes that the system is capable to classify; (2) **Primitive**: it is the basic motion component to form an activity. We decompose activities into a sequence of primitives according to certain signal signatures; (3) **Activity instance**: an instance describes a certain activity performed by a subject.

Also we will use the terminologies like 'overall activity' and 'specific activity' defined based on the above explanations. 'Overall activity' is used to differentiate activities defined in (1) from other upper extremity movements which are not targeted by our system. 'Specific activity' refers to an activity class. In our system design, we have 6 specific activities.

2.2 Hardware Architecture

The main part of the system hardware is a sensing platform. It is integrated with functions of sensor manipulation, data collection and data storage. The data collection part consists of three sensor breakouts, a tri-axial accelerometer, a bi-axial gyroscope and a flex sensor, which are connected to the platform via ADC ports. All of the hardware components are off-the-shelf electronic devices.

The platform is the SparkFun Logomatic v2 Serial SD Datalogger [16]. It has an ARM processor, 8 ADC ports, an on-board USB mass storage stack, and a micro SD card slot. The hardware configuration is quite convenient by simply adjusting the variable assignments in the configuration file. A sample file is shown in Table 2.1 with the line-by-line explanations aside. During the data collection phase, the platform periodically polls readings from the enabled ADC ports and stores the data in an inserted micro SD card. When connected to a computer via a USB

cable, the platform serves as a normal USB device so that users can copy the desired data file to the computer even without a SD card reader.

The tri-axial accelerometer breakout is the SparkFun MMA7361L chip [17], whose output can be directly sampled without any amplifier circuits. The chip supports two dynamic measurement ranges of $\pm 1.5g$ and $\pm 6g$ respectively. Though the smaller range corresponds to a higher resolution, the $\pm 6g$ mode is chose in our application considering the range of general human motion acceleration as well as the constant $1g$ gravity. Figure 2.1 shows the schematic of the peripheral of the accelerometer breakout. Note that the $\pm 6g$ mode is selected by pulling the $g - select$ pin to VCC through a $10K$ resistance.

Table 2.1: Example Configuration File

Variable	Assignment	Explanation
MODE	2	ADC logging
ASCII	Y	Log in ASCII format
Baud	4	Not applied in ADC mode
Frequency	100	Sampling rate is 100Hz
Trigger Character	$	Not applied in ADC mode
Text Frame	100	Not applied in ADC mode
AD1.3	8	Disable Output 8
AD0.3	1	Enable Output 1
AD0.2	2	Enable Output 2
AD0.1	3	Enable Output 3
AD1.2	7	Disable Output 7

AD0.4	4	Enable Output 4
AD1.7	5	Enable Output 5
AD1.6	6	Enable Output 6
Safety On	Y	Enable Output 1

The bi-axial gyroscope breakout is the SparkFun LPY5150AL chip [18], which also requires few peripheral circuit designs. The gyroscope chip has two sets of output of $1\times OUT$ and $4\times OUT$ respectively. The $4\times OUT$ pin is connected to the sensing platform for an amplified output with the pay of decreased sensing range bounded by $1500 deg/s$. Figure 2.2 shows the schematic of the peripheral circuit of the gyroscope chip.

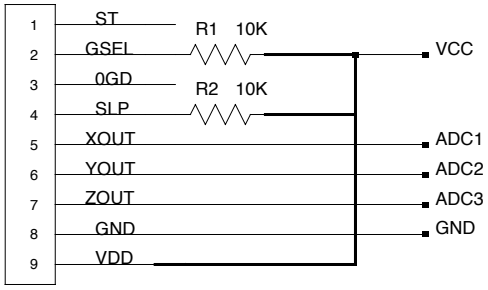


Figure 2.1: Schematic of accelerometer breakout peripheral

Figure 2.2: Schematic of gyroscope breakout peripheral

The flex sensor is basically a variable resistor whose resistance is determined by its deformation

[19]. The sensor is only sensitive in one side and when this side is curved, the resistance will increase correspondingly. Resistance can be converted to a voltage output when applied an operational amplifier circuit. In order to increase the sensing range, we provide the amplifier with dual power supplies. The peripheral circuit of the flex sensor is shown in Figure 2.3. The operational amplifier chip is TLC25M4CN [20] and the ADM8829 chip [21] is used to invert the voltage for negative power supply. The input and output relationship of the flex sensor circuit is shown in equation,

$$V_{out} = VCC \times \frac{R_1}{R_2},$$

where $R2$ is changing responding to the sensor deformation.

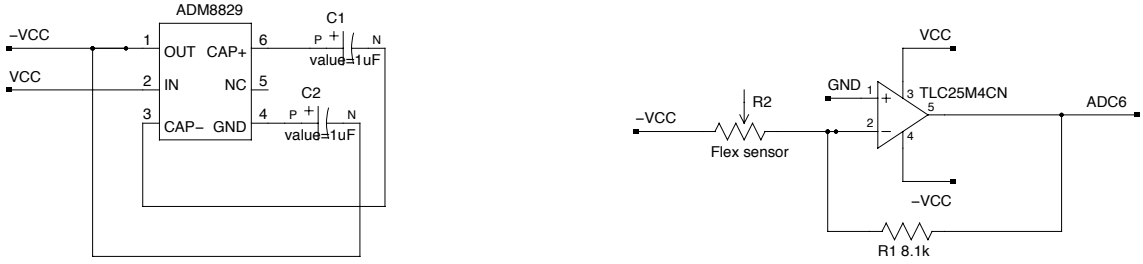


Figure 2.3: Peripheral circuits of flex sensor

All the sensors are connected to the main platform via the ADC ports. The sensing platform is configured to sample at the frequency of $100Hz$. During the data collection phase, the sensor reading is temporarily stored in the inserted micro SD card. Afterwards, all the data would be transferred to a computer for offline process. Figure 2.4(a) shows the complete hardware and Figure 2.4(b) shows the mounting option for the upper limb activity monitoring. Inertial sensors (accelerometer and gyroscope) are used for measuring movement amplitude and orientation, while flex sensor is used for accurately and directly measuring elbow angle changes.

a. System hardware b. Hardware mounting

Figure 2.4: System hardware and mounting for an upper arm

2.3 Algorithm Description

The system algorithm employs a primitive-based template searching method to accomplish the task of activity segmentation, recognition and classification. The thesis draws inspiration and implements the primitive concept in the context of motions and activities described in [9] where the authors claimed that a physical movement could be divided into a sequence of several smaller motions defined as primitives. A transcript of this movement (a sequence of motion primitives assigned to a specific activity) would record order and timing of the basic motions. For example, a transcript for foot during walking could consist of (1) lifting the foot; (2) moving the foot forward; (3) placing the foot on the ground and (4) bearing weight on the foot, with certain periods of time associated with each primitive. In their approach, primitives are constructed by a moving window centered at each point of the signal stream and features are extracted within each window. The primitive classification is accomplished by a Gaussian

Mixture Model (GMM) whose outputs are converted into alphabetical symbols so that an activity is abstracted into a string expression. They use the edit distance to compare the activity template (a transcript which best represents a specific activity) with the symbolized data stream and recognize the targeted activity by thresholding the distance.

The presented system in the thesis inherits the concept of primitive based activity characterization. However, the definition of primitive is different considering its specific application in upper limb activity characterization. Activities are segmented into a sequence of primitives according to the flex sensor measurements. The decomposed signal is a compact version of the original data stream with each small data segment merged into a single primitive. Features are extracted from each primitive and processed by a similar GMM mentioned in [9]. GMM labels each primitive with a cluster symbol and coverts an activity into a symbolic string. In the training phase, given a defined WMFT item, the presented system generates specific templates utilizing the regular expression. For a new set of motion data, the regular expression based searching method is employed to compare each activity template with the symbolized data stream. After a turn of template rotation, all the targeted activities will be recognized. Thus, there is no obvious boundary among the steps of activity segmentation, recognition and classification. The string matching method perfectly combines the above tasks.

2.4 Algorithm Implementation

Figure 2.5 shows the block diagram of the signal processing procedure. The raw sensor data will go through the outlier filter, the primitive segmentation module, the primitive classifier, the sequence analyzer and the activity inference module.

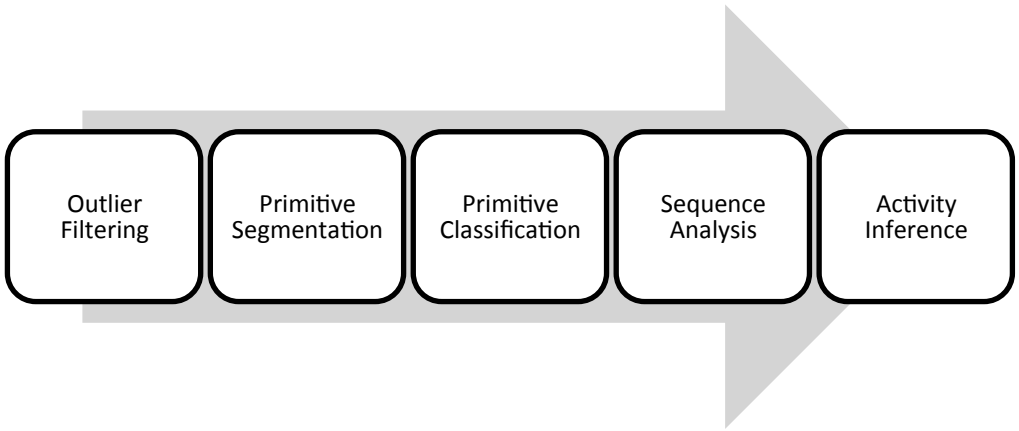


Figure 2.5: Block diagram of signal processing

2.4.1 Outlier filtering

The raw data will be first fed into an outlier filter. Mean filter, median filter and low pass filter are all widely adopted techniques to remove 'ripples' and 'spikes' in human motion signals. In the presented algorithm, a median filter is implemented because of its advantage of edge preservation under certain conditions [22]. This property is extremely important in our application where the flex sensor signal will be used for primitive segmentation.

The performance of a median filter is determined by two key factors: (1) the size of the filter window and (2) the method to handle the boundary issue. In our model, the value of each data point is replaced with a median value of 20 neighboring points. This window size is determined by experiments. The boundary data processing is avoided by shortening the stream length where the first median output is calculated after the system has collected 20 data points. Experiments have proven that the smoothing method would retain sufficient information for human motion characterization. After the filtering step, the profile of the flex sensor signal is extracted.

2.4.2 Primitive Segmentation

Primitive segmentation can be categorized into one of the windowing techniques listed in [23]. Compared to the definition of all the techniques, it is closest to the event-based windowing method. In the primitive segmentation module, the data is segmented into primitives at the points where the flex sensor signal profile achieves a local maximum or minimum.

Instead of employing the traditional peak detection algorithm by looking for the maximum or minimum value in a sliding window with a fixed size, the module takes a novel approach by evaluating the change in the sign of the derivative. We first calculate the derivative between each two neighboring data points and then determine the sign of the individual derivative through a voting mechanism. When a sign flip is detected between two neighboring derivatives, either jumping from negative to positive or vice verse, a sequence of the following 30 samples is evaluated. If the majority of the derivatives are positive, the sign is enforced to be positive and vice versa. The size of the voting window is determined by experiments where 30 is the optimal to recognize all the boundary points of segments as well as avoiding too trivial segmentations.

2.4.3 Primitive Classification

A primitive remains as an unknown data segment until it is labeled through the primitive classification module. The labeling task is accomplished by a GMM classifier.

A Gaussian Mixture Model is a parametric probability density function represented as a weighted sum of a group of Gaussian component densities [24]. For example, a GMM consisting of M weighted component Gaussian densities can be expressed in the equation,

$$p(x|\lambda) = \sum_{i=1}^{M} w_i \, g(x|\mu_i, \Sigma_i),$$

where x is a D -dimension continuous-valued data vector, $w_i, i = 1, \ldots, M$, are the mixture weights and $g(x|\mu_i, \Sigma_i), i = 1, \ldots, M$, are the component Gaussian densities. Each component density is a Gaussian function of the form,

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(x - \mu_i)'{\Sigma_i}^{-1}(x - \mu_i)\},$$

with mean vector μ_i and covariance matrix Σ_i. The mixture weights satisfy the constraint that $\sum_{i=1}^{M} w_i$=1 so that the overall cumulative density function is valued in the interval of $[0, 1]$. In our application, the D -dimension data vector, x is the feature vector we extract from individual primitives. Table 2.2 lists the features we use and the corresponding size they hold in the feature vector. There are 6 channels of data, 3 inputs of the accelerometer, 2 inputs of the gyroscope and 1 input of the flex sensor and each channel has 9 features. The final feature vector turns out to be a 54-tuple. To manually handle the large dimensioned vector is almost impossible.

Table 2.2: Features for primitive classification

Category	**Description**	**Size**
Start to end amplitude	value of subtracting the ending point from the starting point	1×6
Peak to peak amplitude	value of subtracting the minimum value from the maximum value	1×6
Maximum derivative	value of maximum derivative	1×6

Data vector at the maximum point	6 element data vector logged when one channel get its maximum	6×6

A complete GMM is parameterized by the mean vectors, covariance matrices and mixture weights of all component densities. By far the most popular and well-established method to estimate the above parameters is the maximum likelihood (ML) estimation [25]. The way ML estimation works is to find the model parameters which maximize the likelihood of the GMM give the training data. A compact way to express all the parameters is using a vector λ where

$$\lambda = \{w_i, \mu_i, \Sigma_i\}\, i = 1, \dots, M.$$

For a set of training vectors $X = \{x_1, \dots, x_T\}$, the GMM likelihood can be expressed as

$$p(x|\lambda) = \prod_{t=1}^{T} p\,(x_t|\lambda).$$

It is based on the assumption that the training data are independent on each other. To avoid the non-linearity of the maximization problem, a special case of the expectation-maximization (EM) algorithm [26] is employed iteratively. First, an initial guess of λ is given. Then, in each iteration, a new $\bar{\lambda}$, such that $p\left(x\middle|\bar{\lambda}\right) \geq p(x|\lambda)$ is calculated and replaces the previous λ. The following formulas are used in our approach to guarantee a monotonic increase in the likelihood value,

Mixture Weights:

$$\overline{w}_i = \frac{1}{T}\sum_{t=1}^{T} \Pr\,(i|x_t, \lambda).$$

Means:

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T} \Pr\,(i|x_t, \lambda)\, x_t}{\sum_{t=1}^{T} \Pr\,(i|x_t, \lambda)}.$$

Variances:

$$\bar{\sigma}_i{}^2 = \frac{\sum_{t=1}^{T} \Pr\,(i|x_t, \lambda)\, x_t{}^2}{\sum_{t=1}^{T} \Pr\,(i|x_t, \lambda)} - \bar{\mu}_i{}^2.$$

However, the EM algorithm can only be applied when the cluster number, or equivalently the number of component densities is given. By decomposing activities into primitives, it is extremely hard to decide how many primitives would be sufficient to characterize all the motion segments. So we rely on the Bayesian information criterion (BIC) to get a reasonable guess [27]. BIC is a likelihood criterion penalized by the model complexity or the number of parameters in the model. In our application, the number of clusters directly affects the number of parameters needed for model construction. Following the notions in the above two paragraphs, when $X = \{x_1, \dots, x_T\}$ is the data set and $\lambda = \{\lambda_i : i = 1, \dots, K\}$ are the candidates of parametric models and β_i is the number of parameters in the model λ_i. The BIC is defined as,

$$BIC_i = log\mathrm{P}(X|\lambda_i) - \alpha\frac{1}{2}\beta_i logN,$$

where α is a penalty weight and a larger BIC value implies a better model. Note that the $\lambda = \{\lambda_i : i = 1, \dots, K\}$ here indicates the optimal parameter set for different models, which is different from the $\lambda = \{w_i, \mu_i, \Sigma_i\}\, i = 1, \dots, M$, in the previous paragraph where it represents different parameter set for a fixed model in each iteration.

The primitive classifier is embedded with a GMM where all the parameters have been estimated

through the training phase and configured properly to receive unclassified inputs. For an unknown primitive, the a posteriori probability for component i is give by

$$\Pr(i|x_t, \lambda) = \frac{w_i g(x_t|\mu_i, \Sigma_i)}{\sum_{k=1}^{M} w_k g(x_t|\mu_k, \Sigma_k)}$$

where x_t is the feature vector extracted from the primitive. The value of i, which results in the largest a posteriori probability is assigned to the primitive as its classification result. Because we use alphabets to represent each cluster, the classifier assigns individual primitive a unique alphabet as a class symbol.

2.4.4 Sequence Analysis

We decompose an activity into a sequence of primitives according to the flex sensor signal profile. Various combinations and permutations of primitives build up the activities. The way to assemble the primitives makes an activity distinguishable. This property enables the generation of a unique, regular expression based template for each specific activity class, which is referred to as a specific template. Based on the specific templates, a general template is abstracted for the overall activity classes, which can detect activity instances but doesn't give any classification.

The sequence analyzer stores the regular expression based templates for both specific activities and the overall activity classes. A regular expression is a special pattern that specifies a set of strings in an extremely compact way [28]. Instead of numerating all the strings one by one, it uses special operations to construct string expressions. Table 2.3 lists the operations we use in our system design with their grammars aside. The sequence analyzer examines an unknown primitive sequence by interpreting it through a regular expression processor. The processor compares the predefined templates with the current input and finds out all the matching cases.

2.4.4 Activity Inference

The activity inference module continues the processing of the sequence analyzer and assigns a more concrete result of the system output. After an unknown primitive sequence is processed through the sequence analyzer, the activity inference module returns a starting point, an ending point and a matched data segment when a match is detected and associates it back to the current template. To put it in a more user-friendly way, this module presents the activity recognition and classification result: (1) Whether there are predefined activities included in the data stream; (2) If there are activities of interest detected, what activities they are or what class they should be assigned to; (3) For each classified activity, what the starting time is and how long it takes.

Table 2.3: Regular expression operation

Category	Symbol	Grammar
Boolean ‘or’	vertical bar: ‘|’	Separates alternatives. For example, gray|grey can match ‘gray’ or ‘grey’.
Grouping	parentheses: ‘( )’	Define the scope and precedence of the operators. For example, gray|grey and gr(a|e)y are equivalent patterns which both describe the set of ‘gray’ and ‘grey’.
Quantification	question mark: ‘?’	Indicate there is zero or one of the preceding element. For example, colou?r matches both ‘color ’ and ‘colour’.
	asterisk: ‘*’	Indicate there is zero or more of the preceding element. For example, colou*r matches ‘color ’, ‘colour’, ‘colouur’, ‘colouuur’, and so on.
	plus sign: ‘+’	Indicate there is one or more of the preceding element. For example, colou+r matches ‘colour’, ‘colouur’, ‘colouuur’, and so on, but not ‘color’.

Chapter 3

Activity Modeling

In the presented system, an activity is defined as a sequence of primitives. To model an activity, we need to construct the fundamental primitives and arrange them in a proper way. We use alphabetical symbols to represent each primitive class so that an activity is characterized by a sequence of alphabets, a string.

In this section, we focus on the rationale behind the activity modeling approach presented in the thesis. The following terminologies will be introduced: (1) Template: a template is a sequence of alphabets/a string to characterize a specific upper extremity activity; (2) Specific template: a specific template is an abstracted pattern to describe one of the 6 test items/activities specified in our system; (3) General template: a general template incorporates the characteristics of all the 6 specific templates and is used to recognize activity instances from a series of upper limb motions.

3.1 Primitive Construction

The idea of decomposing activities into primitives is an extension of the concept of windowing techniques. A traditional way to delimit data into smaller sessions is to use a sliding window through the data sequence and each window generates a ‘primitive’. This method has been adopted in a lot of cases of lower body activity classification [9]. In our early work, we proposed a similar time-based windowing techniques but it turns out to be not effective in the application of upper limb activity characterization. The reason can be traced to the special nature of upper

limb activities. Unlike the walking signals, they are not periodical and don't hold detectable signatures.

Therefore, we use the profile of flex sensor signal to delimit primitives. Whenever the trend of the signal evolution changes, a new primitive is registered. Note the intuition behind this representation in the physical world. For example, the activity 'lift' can be accomplished by the following four steps: (1) put arm on the table; (2) arm ascent; (3) arm descent and (4) arm retrieval. Primitive transition typically involves the elbow angle change. Flex sensor, which is mounted around the elbow, measures the angle changes and thus is used for primitive segmentation. However, angle changes sometimes occur within a primitive defined in the physical world. Thus, in our application, the definition of a primitive is yielded to the segmentation capability of a flex sensor. Even though, we try to avoid very trivial primitives.

3.2 Primitive Classification

We define primitive as a data segment where the flex sensor signal is monotonic. For any two neighboring primitives, the difference of each other can be easily identified as one of them is monotonically increasing while the other is monotonically decreasing. However, the disparity between any two arbitrary primitives is very abstract. Table 3.1 lists activities, targeted in this paper, with detailed description of primitive decomposition by the flex sensor. In the view of the flex sensor, 'lift' gets two more primitives compared to the intuitive primitive decomposition in the physical world. When subjects sitting on a chair with his/her hands freely putting on the laps, tend to lift a can on the table, they will first retract the forearm a little bit to elevate the hand to the same level of the tabletop, which results in a decreasing of elbow joint angle. Then he/she will extend the whole arm to reach the can, during which, the elbow joint angle changes in an

opposite direction. The lifting process demands an arm retraction so that another primitive is registered. To put the can back to the table is another extension motion. Finally, to retrieve the arm back to the body and put the hand back on the lap are another two primitives delimited by the different elbow joint angle changes. The above description implies that 6 is the minimum set of primitives required to characterize the targeted 6 activities.

However, identifying individual primitive classes solely referring to the intuitive description is still insufficient. The amplitude and orientation of motions, which can hardly be accurately expressed using linguistic descriptions, will also contribute to primitive properties and they are not negligible in primitive classification. For example, all the activities in Table 3.1 contain the primitive of 'retrieve'. In 'forearm to table', the 'retrieve' means retracting the forearm apart from the table so that the hand can be put back on the lap. But in 'extend elbow', subjects should first retrieve the forearm across the table and then apart from the tabletop and these two motions cannot be delimited by observing the elbow joint angle changes.

Thus, the key priority is to find a criterion to characterize primitives from different classes and classify an unknown primitive by quantifying its degree of matching to each class. As mentioned in the section of algorithm implementation, we extract features from primitives and build the GMM classifier to distinguish them. The benefit of GMM is that it is an unsupervised learning model, which means given a set of training data without any manual labels, the EM algorithm can automatically estimate the model parameters. Therefore, property analysis of individual primitive class is saved.

The BIC value is used to select the optimal cluster number, which further saves the endeavors of primitive property estimation and evaluation. However, BIC is not a strictly convex function of

the cluster number. Therefore, another criterion to evaluate the optimal cluster number is set in our application. The system should generate a unique and consistent template for each targeted activity class. It is based on the observation that when the cluster number is increased, the primitive disparity between each two activity classes is strengthened. However, the same activity performed by one subject also presents increased variations in different instances. In this case we leverage the uniqueness and consistence requirement with the consideration of a BIC value and set the cluster number to be 18.

Table 3.1: Primitive segmentation of targeted activities

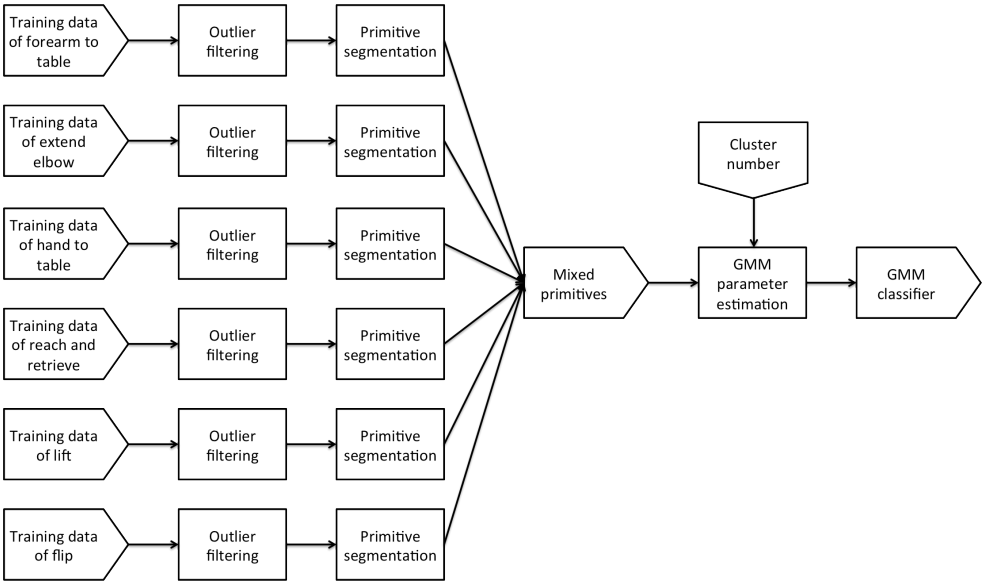
Activity Class	**Primitive Decomposition**	**Sample Template**
Forearm to table (side)	Lift -> lay -> retrieve -> relax	$(G\|B)(O\|A) + P * E * M + Q$
Extend elbow	Lift -> extend -> retrieve -> relax	$(G\|B)L + (D\|I) + Q$
Hand to table (front)	Lift -> lay -> retrieve -> relax	$C(L\|F) + (D\|I) + Q$
Reach and retrieve	Lift -> reach -> retrieve -> relax -> retrieve -> relax	$(C\|R)(L\|F) + K + P + (M\|E) + Q$
Lift	Lift -> reach -> feed -> return -> retrieve -> relax	$(C\|R)(L\|I) + H + H + D + Q$
Flip	Lift -> reach -> lift -> rotate -> retrieve -> relax	$C(I\|O) + N + I + J * D + Q$

3.3 Template Generation

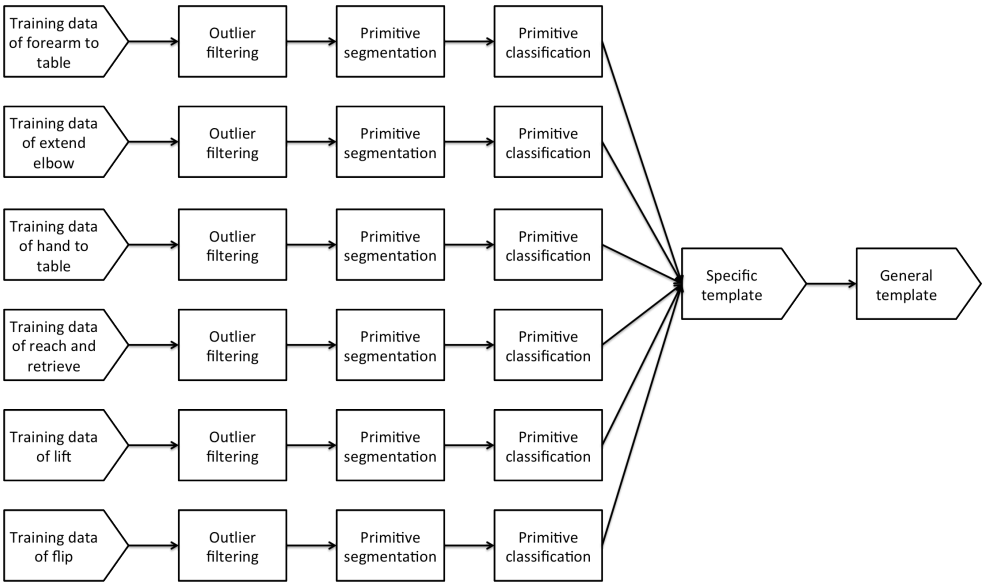
Templates are used to characterize activities so that activity recognition and classification can be easily carried out by thresholding the disparity between a motion sequence and a templated activity. Specific templates to characterize a specific activity among the 6 targeted activities are generated during the training phase. During the phase, subjects are requested to repeat individual test items/activities at least five times. A large training set is required not only for the GMM parameter estimation through the EM algorithm, but also necessary to incorporate instance variations considering the large freedom of human upper limb movements.

Figure 3.1 illustrates the information flow of the template generation procedure. The training data are firstly smoothed by the same outlier filter as shown in Figure 2.5. Then the motion sequence is delimited into a group of primitives through the primitive segmentation block by only feeding the flex sensor output.

In the first stage, all the segmented primitives are mixed together to create the GMM. Note that this group of primitives also includes the decomposed motion segments not related to any of the targeted activities. They may be transition motions between two consecutive activities or fake motions added intentionally to simulate upper limb movements in the free-living environment. So the necessity of the automatic selection of a cluster number and the unsupervised training process is further emphasized.



a. Training in the first stage



b. Training in second stage

Figure 3.1: Information flow in the training phase

During the second stage, primitives from the same activity instance are assembled back together as a data block. By assigning different alphabets to individual primitive classes, an activity instance is converted into a string after primitive-wise classification and labeling.

Based on the ground truth recorded during the training phase, activity labeling of individual data blocks or equivalently strings is carried out by manual intervention. Compared to the intractable primitive labeling task, activity labeling is much easier and more practical following the activity definition and description formulated in the section of system overview. Assembling strings with the same label together, we want to generate a template for each activity class.

Template generation should take into account both the underfitting and overfitting issues. If we randomly choose one string from the string set of the same activity label, the overfitting problem is inevitable. On the contrary, if we extract the common part among the strings under the same activity class and only put constraints at these parts, sometimes underfiitting problem would be brought up as we cannot guarantee these parts are sufficient to differentiate a specific activity from the overall activity set. Thus, we want to incorporate all the instance variations of a specific activity into the specific template. To store multiple strings as a single activity template is an option. But the activity recognition and classification process would be very cumbersome. For example, if we have 5 strings for each activity class, we should run a string matching algorithm 30 times to get a complete result. Therefore, we take the regular expression approach usually embedded in text editors for fast and efficient pattern recognition. Strings representing instances of the same activity class are grouped together to abstract a regular expression based template. Table 3.1 lists samples of the specific template for each activity class. The template of ‘forearm to table (side)’ is

$$(G|B)(O|A) + P * E * M + Q.$$

The interpretation of the template is as the following: (1) The activity can at least be decomposed into 4 primitives with the ending primitive is fixed as Q; (2) The starting primitive can be either G or B. The same situation applies to the second primitive switching between the options between O and A; (3) The activity can also be performed by including the motion of P and E. But their occurrence is quite independent and their repetition is also uncontrollable; (4) The primitive, M is indispensable to complete the whole activity but the time of repetition is not constrained.

Currently, we still don't have very strict formulations of how to abstract templates from a set of activity related strings and at the same time avoiding both the underfitting and overfitting problem. The basic rules are described as the following: (1) If a common part is detected among the strings of the same activity class and this part is unique for this specific activity class, we usually emphasize its contribution to activity classification in the regular expression template and loose the constraints of the rest of the expression; (2) If no common parts are detected or the common part makes no difference among the overall activity set, we will numerate the instance patterns and combine them using the regular expression operations; (3) If all the strings under the same activity class hold the same length, we are more inclined to using the operator, | to incorporate the alphabet variations in the individual string position; (4) If any two strings have different lengths, we put $+$ in the position where the alphabet ahead appears in every instance and $*$ where the alphabet ahead appears occasionally.

Specific templates are used for individual activity classification. To make the system more robust, the concept of general template is proposed. Different from the specific template, a general

template can recognize targeted activities from a series of motions but not give any classification result. Associating specific templates with specific activities, the general template is corresponded to the overall activity set. So each subject has only one general template but 6 specific templates.

The general template is abstracted based on the patterns from individual specific templates. Basically, we truncate the starting primitive and the ending primitive from all the templates. Referring to Table 3.1, the general template for this subject is

$$(B|G|C|R)(A|D|E|F|H|I|J|K|L|M|N|O|P)(A|D|E|F|H|I|J|K|L|M|N|O|P) + Q.$$

It is generated based on four interesting discoveries on the specific activity templates: (1) Activities are always started from a fixed set of primitives. When patients taking the WMFT, we assume they usually start with their hands freely rest on the laps and end with the hands back to the initial place. So the starting and ending primitive represent the motions of lifting the hands from the laps to the tabletop and putting hands back to the initial position respectively. (2) Activities can be decomposed at least into four primitives. The simplest activity in our activity set is 'hand to table'. To perform this activity, patients should first retract the arm a little bit to lift the hand to the same level with the tabletop. When putting hands on the table, it is actually an elbow joint extension process. To get ready for the next test item, patients will put their hands back on the laps, which is pretty the same as reversing the above two motion segments. 'Lift' and 'flip' are much more complex activities and the increased primitive number is expected. (3) The set of starting primitives and ending primitive are very unique patterns which only appear in this two specific string positions. Through all the activity instances collected during the training phase, these two primitives have never been detected in a different position.

3.4 Template Adjustment

The system is aimed at classifying a set of test items/activities extracted from the WMFT. The test is used to evaluate the rehabilitation of stroke patients. So performance improvement is expected. A set of immutable activity templates would not be applicable considering the performance variations. In order to handle this kind of situation which can be usually encountered in rehab care, the templates are kept editable such that the template adjustment is maintained to be flexible.

It is a normal case that in the first couple of weeks after the stroke attack, patients would feel extremely difficult to freely extend his/her arms to reach something. He/she will have to reply on the whole upper body movements to control the approaching path of the upper limbs. During the process of recovery, patients will gradually be more capable of arm flexion and extension. So cases like that specific templates generated during the first couple of weeks cannot recognize activity instances performed by the same subject in a later time are inevitable because primitive motions may have changed tremendously. Therefore, we want to adjust specific templates from time to time. However, the above example is just an extreme case in order to describe the necessity of template adjustments.

In the thesis, we want to propose a way to update specific templates. Details about the condition to trigger the template updates and how to combine manual interventions for ground truth would not be covered. The method is based on the observation through our experiments that the general template is robust to identify activity instances though without any classification result. When we use the general template to search for pattern matches, the algorithm will return a string segment for each matching. Note that the general template only puts constraints on the starting and ending

primitives. The returned string segments would be activity instances with various patterns in the middle session. For the same data stream, we then rotate the stored specific templates and use them to search for pattern matches. We assume that each test item was performed once and the specific templates are designed to be extremely immune to false positives at the pay of slight vulnerability to false negatives. So the sum of the detected activity instances by individual specific template would be equal to or less than the number of instances identified by the general template. The later case indicates that some of the specific templates trained at an earlier time are no longer compatible to the current activity variations. By comparing the pattern of the matched data segments returned by the general template and the specific templates under the help of human labeling of ground truth, adjustment of specific templates is applicable. A special case with one missing segment return after a complete rotation of every specific template, enables the system to precede template adjustment without any human interventions.

Template adjustments are simply incorporating new instance variations to the previous generated templates where the new variations are identified by the general template. The ground truth guides the system in terms of which activity class, the variation should be added to. Two principles need to follow through adjustment operation: (1) The updated templates should persist to the patterns inherited from its ancestor. The addition of new patterns should not degrade the old patterns. This rule is used to avoid the overfitting problem brought by the special situation where the unrecognized activity instance may be a temporary performance variation; (2) The updated templates should remain unique, which implies that the new incorporated pattern should be new to all the other specific templates and any changes should be distinguishable from template to template. If either of above is violated, the system will reject the updating request or proceed the updates selectively.

Chapter 4

System Evaluation

The goal of the presented system is to recognize from a serials of motions, activity instances belonging to the 6 test items/activities listed in Table 3.1 and classify them into correct category. The evaluation metrics include whether the system is capable of properly delimiting the data stream into primitive segments according to the flex sensor signal profile, whether the system is capable of completely recognizing all the activity instances in the data stream and whether the system is capable of correctly classifying each activity instance into its corresponding activity category. For the first metric, we give an example output of the primitive segmentation module by feeding a typical data stream into the system. For the rest two metrics, we will show the result by providing precisions and recalls.

4.1 Experiment Setup

The system is a personalized activity recognition and classification system such that the personal information, including the GMM parameters and activity templates, is archived beforehand during the training phase. Subjects are instructed to perform the 6 test items/activities listed in Table 3.1 without any outside interference. Each activity is repeated for five times in order to collect the possible instance variations. Different activities are separated by a shaking signal and the complete experiment is videotaped and post analyzed for accurate ground truth information.

Using the procedure introduced in the section of activity modeling, individualized primitive classifier, and templates of specific activity and overall activity set are established.

System evaluation is based on the testing data collected when subjects are instructed to perform the 6 activities in a random order. Figure 4.1 shows an example data stream collected during the testing phase. The data is supplied to the signal processing pipeline shown in Figure 2.1. We put a probe at the output of the primitive segmentation module. Figure 4.2 shows the segmentation result. The amplitude of the solid line in the figure is a binary function of the flex sensor derivative. When the sign of the derivative is positive, the value is set to 350. Otherwise it remains as 0. The value of 350 is chosen for better illustration. The sign of the derivative is determined by the voting mechanism. In the figure, the upward and downward steps illustrate the primitive boundary. Note that there are no trivial primitives inside the motion sequence. This is a very typical case through all the data sets, which indicates the primitive segmentation output is quite reliable and all the delimited data segments can be trusted as a primitive in our definition.

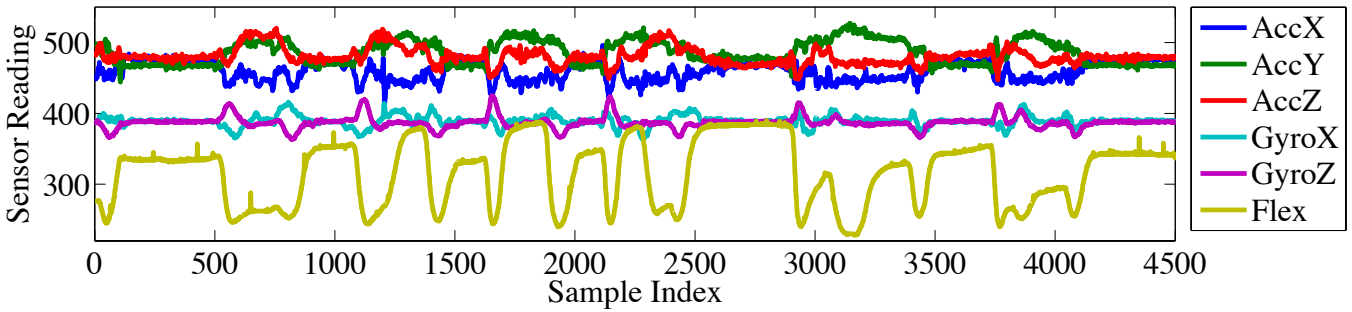


Figure 4.1: Raw data stream

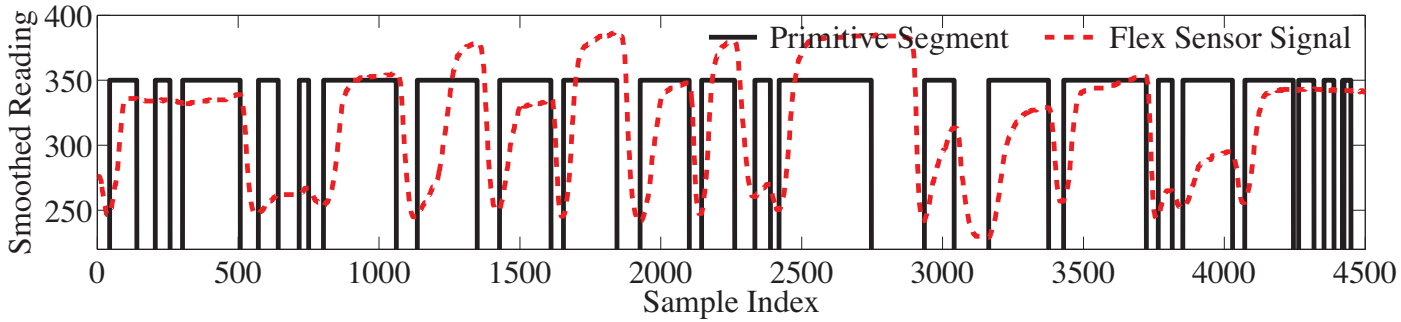


Figure 4.2: Primitive segmentation output

After primitive-wise classification and labeling, the primitive sequence is converted into a string expression. Activity recognition and classification are based on the well-developed regular expression matching method. The system picks one of the specific templates generated during the training phase and searches for matches through the primitive sequence. If matches are detected, activity recognition and classification for a specific activity class is accomplished. By rotating individual specific templates, Figure 4.3 illustrates the activity recognition and classification result of the same data stream shown in Figure 4.1, where the solid line shows the ground truth and the dotted line shows the system output.

To test the necessity and effectiveness of template adjustment, subjects were asked to intentionally perform the same set of activities slightly differently by slowing down the speed, increasing the motion range and etc. Within our expectation, the original specific templates introduce several false negatives as shown in Figure 4.4. In the same figure, we can see that the general template works pretty well by recognizing all the activity instances. Without violation of the principles presented earlier in the section of template adjustment, specific templates are edited according to the general template matching output. By employing the updated specific templates, decreased false negatives can be observed in Figure 4.4. Note that using the updated templates will not affect the system performance in processing the history data

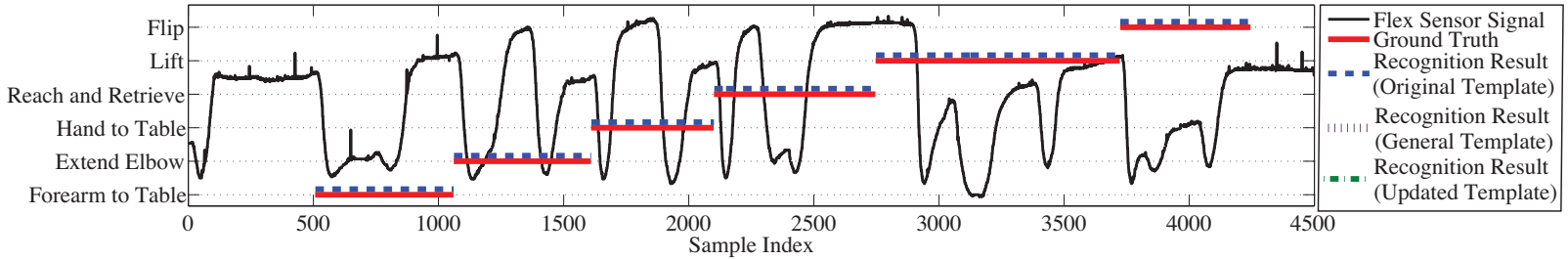


Figure 4.3: Activity recognition

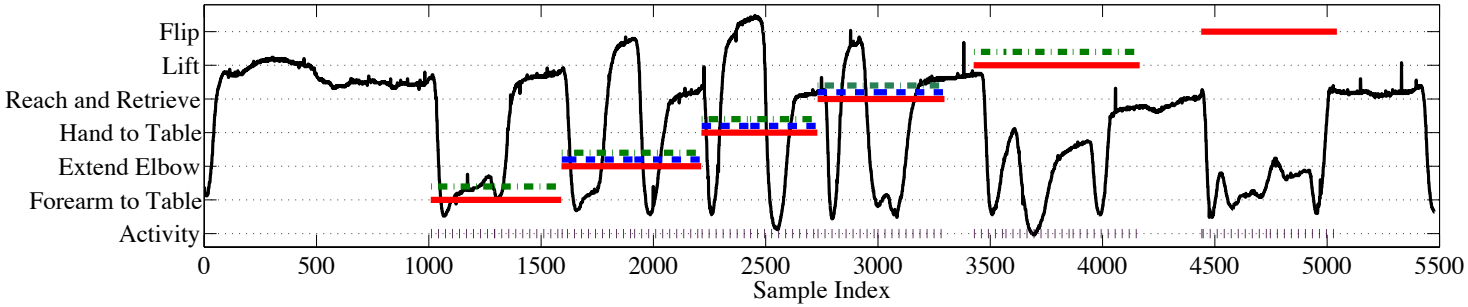


Figure 4.4: Improved accuracy with template adjustment

4.2 Classification Result

The experiments described above were conducted among 5 subjects whose heights varying from 160cm to 185cm. The rationale behind choosing height as a variable is based on the assumption that the lengths of human limbs are proportional to human heights. Table 4.1 shows the overall testing results. We use precisions and recalls as the system has both false positives and false negatives. Precision calculates the fraction of recognized activity instances that are actually belonging to the category represented by the current template among all the recognized instances while recall calculates the fraction of relevant activity instances recognized by the current template among all the relevant instances. We use three sets of templates, specific templates, general templates and updated specific templates to test the algorithm on two sets of data, data of normal activity performance and data of varied activity performance for individual subjects.

Table 4.1: Overall testing result

Template Set	**Normal Instance**		**Varied Instance**	
	Precision	**Recall**	**Precision**	**Recall**
Specific Template	100%	96.67%	90.48%	63.33%
General Template	100%	100%	100%	100%
Updated Template	100%	100%	92.86%	86.67%

In the table, we can see that the intra-subject templates are quite effective to detect the normal activity instances. The high precision is due to the principle we employ during the template generation. The template extracted for each activity class should be unique. Any pattern that brings false positives among the training data set will not be incorporated into the general or

specific templates. Even though the principle for template generation is more inclined to give high precision, the rate of recall is still acceptable, which indicates that the primitive based upper limb activity recognition and classification is applicable. For intentionally varied activity instances, the pre-generated specific templates result in a tremendous decrease in both precision and recall. By integrating the returned string segments of the general template matching algorithm, the updated specific templates achieve a much higher accuracy especially in terms of recall.

Chapter 5

Conclusion

5.1 Conclusion

In this paper we presented a novel system combining inertial sensors and a flex sensor to monitor and evaluate complex upper limb activity performance, under the scenario of WMFT, of neurological patients going through rehabilitation. We presented a flex sensor based primitive segmentation method. This method uses the permutation of primitives to generate templates for activity recognition and classification. To incorporate the instance variation, the regular expression based pattern matching method is employed in the recognition and classification procedure.

The effectiveness of the algorithm is proven via an initial experimental campaign involving 5 healthy subjects. The results show its special success when incorporating the procedure of template adjustment.

However, The current system is still suffering from several shortcomings: (1) Trials among a large scale community with participations of stroke patients have not been proceeded to evaluate the system robustness; (2) The system capacity is still limited by the scenario of the WMFT; (3) The complete protocol of template adjustment has not been proposed

5.2 Future Work

We plan to extend the application of the presented system to the classification of the upper limb activities during the free-living conditions among a larger group of subjects. As such activities will become increasingly complex, we plan to extend the presented system with wearable system attached to the forearm so that we can monitor simultaneously the movements of both shoulder joint and elbow joint [29].

The extended system will enhance the types of features and primitives that can be extracted to most precisely characterize activities in the community. Besides, 3D motion tracking techniques will be incorporated to break through the restriction of targeted activity set.

Appendix I

Code for Primitive Segmentation

```
function primitive_segmentation

data = load('../data/training.mat');

field = fieldnames(data);

window1 = 19;

window2 =29;

for i = 1:length(field)

    raw_data = data.(char(field(i)));

    len1 = length(raw_data(:,6))-window1;

    median_data.(char(field(i))) = zeros(len1,6);

    for j = 1:len1

        median_data.(char(field(i)))(j,:) = median(raw_data(j:j+window1,:));

    end

    len2 = len1-1;

    derivative.(char(field(i))) = diff(median_data.(char(field(i)))(:,6));
```

```
    pattern.(char(field(i))) = zeros(len2,7);
    flag = 0;
    for k = 1:len2
        if ((derivative.(char(field(i)))(k) ~= 0) && ((derivative.(char(field(i)))(k) > 0) ~= flag) &&
(k ~= len2))
            if ( k+window2 > len2)
                tendancy1 = sum((derivative.(char(field(i)))(k:len2) > 0));
                tendancy2 = sum((derivative.(char(field(i)))(k:len2) < 0));
            else
                tendancy1 = sum((derivative.(char(field(i)))(k:k+window2) > 0));
                tendancy2 = sum((derivative.(char(field(i)))(k:k+window2) < 0));
            end
            if (tendancy1 < tendancy2)
                flag = 0;
                disp(k);
                disp(flag);
            elseif (tendancy1 > tendancy2)
                flag = 1;
```

```
            disp(k);
            disp(flag);
        end
    end
    pattern.(char(field(i)))(k,1) = flag * 600;
end
pattern.(char(field(i)))(1:len2,2:7) =  raw_data(1:len2,:);
figure(i);
plot(pattern.(char(field(i))));
title(char(field(i)));
saveas(gcf,strcat('../figure/',char(field(i)),'.fig'));
end
for i = 1:length(field)
activity = pattern.(char(field(i)));
flag = -1;
index = 0;
for j = 1:length(activity(:,1))
```

```
        if (activity(j,1) ~= flag)
          flag = activity(j,1);
          index = index + 1;
          name = strcat(char(field(i)),'_',num2str(index));
          k = 1;
        else
           k = k + 1;
        end
        primitive.(name)(k,:) = activity(j,2:7);
    end
end
save('../data/primitive.mat','-struct','primitive');
end
```

Appendix II

Code for Primitive Classification

```
function primitive_classification

data = load('../data/primitive.mat');

field = fieldnames(data);

file = strcat('../data/parameter','.mat');

para = load(file);

obj = gmdistribution(para.mean, para.covariance, para.component);

for i = 1:length(field)

    label = char(field(i));

    loc = findstr(label,'_');

    label1 = label(1:loc(2)-1);

    index = str2double(label(loc(2)+1:end));

    primitive = data.(label);

    if (length(primitive(:,1)) >= 2)

        feature.(label1)(index,:) = feature_extraction(primitive);
```

```
        class = cluster(obj,feature.(label1)(index,:));
        script.(label1){index} = char('A'-1+class);
    else
        script.(label1){index} = 'Z';
    end
end
file = strcat('../data/script','.mat');
save(file,'-struct','script');
disp(script);
end


function feature = feature_extraction(data)
dimension = 9;
feature = zeros(1,length(data(1,:))*dimension);
for i=1:length(data(1,:))
    index = (i-1)*dimension;
    feature(index+1) = max(data(:,i))-min(data(:,i));
```

```
    feature(index+2) = data(end,i)-data(1,i);

    feature(index+3) = max(diff(data(:,i)));

    [Y I] = max(data(:,i));

     feature(index+4:index+dimension) = data(I,1:6);

end

end
```

Appendix III

Code for Test Item Classification

```
function sequence_test(index)

testing = load('../data/testing.mat');

sequence = testing.(strcat('testing', '_',num2str(index)));

record = load(strcat('../data/record_',num2str(index),'.mat'));

plot(record.(char(label(i)))(1):record.(char(label(i)))(2),(record.(char(label(i)))(1):record.(char(la
bel(i)))(2))*0+350,'r');

window1 = 19;

window2 = 29;

len1 = length(sequence(:,6))-window1;

median_sequence = zeros(len1,6);

for i = 1:len1

   median_sequence(i,:) = median(sequence(i:i+window1,:));

end

len2 = len1-1;
```

```
derivative = diff(median_sequence(:,6));

pattern = zeros(len2,7);

flag = 0;

for k = 1:len2

   if ((derivative(k) ~= 0) && ((derivative(k) > 0) ~= flag))

      if ( k+window2 > len2)

         tendancy = sum(derivative(k:len2));

      else

         tendancy = sum(derivative(k:k+window2));

      end

      if (tendancy < 0)

         flag = 0;

      elseif (tendancy > 0)

         flag = 1;

      end

   end

pattern(k,1) = flag * 350;
```

```
end

pattern(1:len2,2:7) =  median_sequence(1:len2,:);

flag = pattern(1,1);

dimension = 9;

k = 1;

period(k,1) = 1;

for i = 1:length(pattern(:,1))

   if (pattern(i,1) ~= flag)

      period(k,2) = i-1;

      primitive = pattern(period(k,1):period(k,2),2:7);

      for j=1:length(primitive(1,:))

         index = (j-1)*dimension;

         feature(k,index+1) = max(primitive(:,j))-min(primitive(:,j));

         feature(k,index+2) = primitive(end,j)-primitive(1,j);

         feature(k,index+3) = max(diff(primitive(:,j)));

         [Y I] = max(primitive(:,j));

         feature(k,index+4:index+dimension) = primitive(I,1:6);
```

```
        end
        flag = pattern(i,1);
        k = k+1;
        period(k,1) = i;
    end
end
period(k,2) = length(pattern(:,1));
primitive = pattern(period(k,1):period(k,2),2:7);
if (length(primitive(:,1)) >= 2)
    for j=1:length(primitive(1,:))
        index = (j-1)*dimension;
        feature(k,index+1) = max(primitive(:,j))-min(primitive(:,j));
        feature(k,index+2) = primitive(end,j)-primitive(1,j);
        feature(k,index+3) = max(diff(primitive(:,j)));
        [Y I] = max(primitive(:,j));
        feature(k,index+4:index+dimension) = primitive(I,1:6);
    end
```

```
end

para = load('../data/parameter.mat');

obj = gmdistribution(para.mean, para.covariance, para.component);

script = cell(1,length(feature(:,1)));

for i = 1:length(feature(:,1))

    class = cluster(obj,feature(i,:));

    script{i} = char('A'-1+class);

end

string = char(script)';

disp(string);

template.('forearm') = '(G|B)(O|A)+P*E*M+Q';

template.('extend') = '(B|G)L+(D|I)+Q';

template.('hand') = 'C(L|F)+(D|I)+Q';

template.('reach') = '(C|R)(L|F)+K+P+(M|E)+Q';

template.('lift') = '(R|C)(L|I)+H+H+D+Q';

template.('flip') = 'C(I|O)+N+I+J*D+Q';

template.('general')

='(B|G|C|R)(A|D|E|F|H|I|J|K|L|M|N|O|P)(A|D|E|F|H|I|J|K|L|M|N|O|P)(A|D|E|F|H|I|J|K|L|M|N|O|P)
```

```
*Q';

disp(template);

color = [0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0];

figure(2);

plot(sequence(:,6));

hold on;

label = fieldnames(record);

for i = 1:length(label)

plot(record.(char(label(i)))(1):record.(char(label(i)))(2),(record.(char(label(i)))(1):record.(char(la

bel(i)))(2))*0+230+i*25,'r');

end

label = fieldnames(template);

for i = 1:6

    disp(char(label(i)));

    expressn = template.(char(label(i)));

    [matchstart,matchend,matchstring] = regexp(string,expressn,'start','end','match');

    disp(matchstring);

    for j = 1:length(matchstart)
```

```
plot(period(matchstart(j),1):period(matchend(j),2),(period(matchstart(j),1):period(matchend(j),2)
)*0+230+i*25+5,'b');
    end
end
expressn = template.('general');
[matchstart,matchend,matchstring] = regexp(string,expressn,'start','end','match');
disp(matchstring);
for j = 1:length(matchstart)
plot(period(matchstart(j),1):period(matchend(j),2),(period(matchstart(j),1):period(matchend(j),2)
)*0+230);
end
template.('forearm') = '(G|B|R)(O|A)+P*E*M+Q';
template.('extend') = '(B|G)(L|I)+(D|I)+Q';
template.('hand') = 'C(L|F)+(D|I)+Q';
template.('reach') = '(C|R)(L|F|I)+K+P*(M|E)*Q';
template.('lift') = '(R|C)(A|D|E|F|H|I|J|K|L|M|N|O|P)+H+H+D+Q';
template.('flip') = 'C(I|O)+N+I+J*D+Q';
label = fieldnames(template);
```

```
for i = 1:6
    disp(char(label(i)));
    expressn = template.(char(label(i)));
    [matchstart,matchend,matchstring] = regexp(string,expressn,'start','end','match');
    disp(matchstring);
    for j = 1:length(matchstart)
plot(period(matchstart(j),1):period(matchend(j),2),(period(matchstart(j),1):period(matchend(j),2)
)*0+230+i*25+10,'b');
    end
end
hold off;
end
```

Reference

[1] Lloyd-Jones D, Adams RJ, Brown TM, et al, "Heart Disease and Stroke Statistics – 2010 update. A Report from the American Heart Association Statistics Committee and Stroke Statistics Subcommittee," Circulation, 2010, 121:e1-e170.

[2] Manjila S, Masri T, Shams T, et al, "Evidence-based Review of Primary and Secondary Ischemic Stroke Prevention In Adults: A Neurosurgical Perspective," Neurosurg Focus, 2011 Jun, 30(6):E1.

[3] The University Hospital, University of Medicine & Dentistry of New Jersey, Newark, New Jersey, "Stroke Statistics," http://www.theuniversityhospital.com/stroke/stats.htm.

[4] David M. Morris, Gitendra Uswatte, Jean E. Crago, et al, "The reliability of the Wolf Motor Function Test for assessing upper extremity function after stroke," Archives of Physical Medicine and Rehabilitation, Volume 82, Issue 6, June 2001, Pages 750-755.

[5] Stroke Engine, "In Depth Review of the Wolf Motor Function Test (WMFT)," http://www.medicine.mcgill.ca/strokengine-assess/module_wmft_indepth-en.html.

[6] Ghasemzadeh, H, Jafari, R, Prabhakaran, B, "A Body Sensor Network With Electromyogram and Inertial Sensors: Multimodal Interpretation of Muscular Activities," Information Technology in Biomedicine, IEEE Transactions on, vol. 14, no. 2, pp. 198-206, March 2010.

[7] Xiaoyu Xu, Maxim A. Batalin, William J. Kaiser, et al, "Robust Hierarchical System for Classification of Complex Human Mobility Characteristics in the Presence of Neurological Disorders," BSN, pp.65-70, 2011 International Conference on Body Sensor Networks, 2011.

[8] Roozbeh Jafari R., Li W., Bajcsy R., et al, 2007, "Physical Activity Monitoring for Assisted Living at Home," In IFMBE Proceedings, 4th International Workshops on Wearable and Implantable Body Sensor Networks, pp. 213-219.

[9] Hassan Ghasemzadeh, Vitali Loseu, Roozbeh Jafari, "Collaborative Signal Processing for Action Recognition in Body Sensor Networks: A Distributed Classification Algorithm Using Motion Transcripts," IPSN 2010: 244-255.

[10] Soon Mook Jeong, Tae Houn Song, Hyun Uk Jeong, et al, 2009, "Game Control Using Multiple Sensors," In Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia. ACM, New York, NY, USA, 632-636.

[11] Overduin, S. A., Zaheer, F., Bizzi, E., et al, 2011, “An instrumented glove for small primates,” Journal of Neuroscience Methods 187, 100-104.

[12] Luinge, H.J. and Veltink, P. H., 2005, “Measuring Orientation of Human Body Segments Using Miniature Gyroscopes and Accelerometers,” Medical and Biological Engineering and Computing, 43 (2), pp. 273-282.

[13] Hyde RA, Ketteringham LP, Neild SA, et al, “Estimation of Upper-limb Orientation Based on Accelerometer and Gyroscope Measurements,” IEEE Trans Biomed Eng. 2008 Feb, 55(2 Pt 7): 746-54.

[14] Demonstration of the Wolf Motor Function Test by Occupational Therapist Veronica Rowe and stroke survivor volunteer, “WMFT with Descriptions part 1 Items 1-9,” http://www.youtube.com/watch?v=tHsRfx3MbEM.

[15] Demonstration of the Wolf Motor Function Test by Occupational Therapist Veronica Rowe and stroke survivor volunteer, “WMFT with Descriptions part 2 Items 10-17,” http://www.youtube.com/watch?v=kb-q3VRynv4.

[16] “Logomatic v2 Serial SD Datalogger datasheet,” http://www.sparkfun.com/products/8627.

[17] “Three Axis Low-g Micromachined Accelerometer datasheet,” http://www.freescale.com/files/sensors/doc/data_sheet/MMA7361L.pdf.

[18] “Dual axis pitch and yaw analog output gyroscope datasheet,” http://www.sparkfun.com/datasheets/Sensors/IMU/lpy5150al.pdf.

[19] “Flex sensor 4.5” datasheet,” http://www.sparkfun.com/products/8606.

[20] “TLC25M4CN datasheet,” http://www.alldatasheet.com/view.jsp?Searchword=TLC25M4CN.

[21] “ADM8829 datasheet,” http://www.analog.com/static/imported-files/data_sheets/ADM8828_8829.pdf.

[22] Salem Saleh Al-amri, N. V. Kalyankar, S.D. Khamitkar, “A Comparative Study of Removal Noise from Remote Sensing Image,” International Journal of Computing Science Issues, IJCSI, Vol. 7, Issue 1, No. 1, January 2010.

[23] Stephen J Preece, John Y Goulermas, Laurence P J Kenney et al, “Activity identification using body-mounted sensors – a review of classification techniques,” 2009 Physiological Measurement, vol 30, number 4.

[24] Douglas Reynolds, “Gaussian mixture models,” MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02140, USA.

[25] In Jae Myung, 2003, “Tutorial on Maximum Likelihood Estimation,” Journal of Mathematical Psychology, Volume 47, Issue 1.

[26] ChengXiang Zhai. 2007, “A Note on the Expectation-Maximization (EM) Algorithm,” Department of Computer Science, University of Illinois at Urbana-Champaign.

[27] Nishida, M., Kawahara, T., “Unsupervised Speaker Indexing Using Speaker Model Selection Based on Bayesian Information Criterion,” Acoustics, Speech, and Signal Processing, 2003 IEEE International Conference on, pages I -172 – 175 vol.1.

[28] Ken Thompson, “Regular Expression Search Algorithm,” Communications of the ACM, vol. 11, no. 6, pp 419-422.

[29] Zhiqiang Zhang, Lawrence W C. Wong, Jian-Kang Wu, “3D Upper Limb Motion Modeling and Estimation Using Wearable Micro-sensors,” BSN ’10 Proceedings of the 2010 International Conference on Body Sensor Networks.