The Pennsylvania State University

The Graduate School

Information Sciences and Technology

EVOLUTIONARY-BASED FEATURE EXTRACTION FOR GESTURE RECOGNITION USING A MOTION CAMERA

A Dissertation in

Information Sciences and Technology

by

Eun Yeong Ahn

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2012

The dissertation of Eun Yeong Ahn was reviewed and approved* by the following:

John Yen
University Professor of Information Sciences and Technology
Dissertation Advisor
Chair of Committee

Dinghao Wu
Senior Lecturer and Research Scientist of Information Sciences and Technology

Dongwon Lee
Associate Professor of Information Sciences and Technology

Patrick Reed
Associate Professor of Civil Engineering

Tracy Mullen
Software Engineer
Special Member

Mary Beth Rosson
Professor of Information Sciences and Technology
Director of Graduate Programs

*Signatures are on file in the Graduate School

ABSTRACT

Gesture recognition systems have garnered increasing interest for their potential to support more natural human-computer interactions. However, compared to other human-computer interaction technologies such as speech recognition, gesture recognition has not been actively applied to personal devices such as mobile phones or laptops due to the spatial requirements when performing gestures as well as sensitivity to background noise. My research first devises a problem of recognizing speed sensitive finger gestures using a novel camera called Dynamic Vision Sensor camera, which detects the temporal luminance difference for each pixel at microsecond-level granularity and outputs a stream of on-events (brighter) and off-events (darker) to the hardware. As with other machine learning problems, the performance of a gesture classification task depends on how well the representative features are extracted. Thus the feature extraction process must consider device-specific data properties to maximize the feature recognition abilities while minimizing computational cost. My research studies two feature extraction methods, namely local and global feature extractions, which are designed to maximize the performance of the DVS camera-based gesture recognition system.

First, the local feature extraction method aims to extract a smaller number of representative features from a long sequence of the raw gesture events detected by the DVS camera using segmentation. This approach is called the local feature extraction, since the features are extracted by considering neighboring events only. Specifically, I propose bottom-up segmentation methods, where the sequence of events are first divided into segments having the same time interval, called the time-based, or the same number of events, called the event-based, and the segments are repeatedly augmented based on the event distributions of the neighboring segments. The experimental results show that the event-based initial segmentation outperforms the time-based across different classifiers, and is more robust to noise. I also found that Bayesian

network classifier is more accurate than hidden Markov model when features are well extracted using the event-based segmentation.

Second, the global feature extraction method aims to construct higher level compound features by transforming the locally extracted features. Specifically, an evolutionary algorithm is employed to find a good set of simple and compound features. This is a challenging task due to the large search space and the risks of overfitting. I define problem-specific representation, genetic operators, and evaluation methods, and analyze how the specified mutation and crossover operator controls the individual's search space. The experimental results show that the proposed EA can extract a good set of compound features that can enhance the performance accuracy with a smaller number of features. Finally, I show how my evolutionary-based feature extraction approach can serve as a knowledge discovery process in the context of gesture recognition.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful to Dr. John Yen and Dr. Tracy Mullen for advice throughout my PhD program. Their support and encouragement help me enjoy research, and challenge new ideas and professional activities. I'd like to thank my committee members Dr. Dinghao Woo and Dr. Dongwon Lee for their fruitful advice and comments. I am also grateful to Dr. Patrick Reed, who brought me to the field of evolutionary algorithm and helps me explore this field. Without the advice and acute comments from my committee members, this dissertation could not been completed.

I also like to thank Drs Jung Haeng Lee, Hyun-Suk Ryu, Keun-ju Park, and Byung Chang Kang at the Samsung Advanced Institute of Technology, where I did the summer internship in 2010. Thanks to their support, I could be exposed to the novel camera called the dynamic vision sensor camera, which played a key role in my dissertation. I also like to thank Dr. Tobi Delbruck for providing a DVS camera.

I'd like to acknowledge my lab mates in Intelligent Agents Laboratory, and my best friends, Bora Lee, Hye-jin Kim, Young Shin Lim, and Jung Eun Gwak, and Hyun-Woo Kim for their personal support. Finally, I always thank my lovely family and my fiancé, Jongmin, for their love and support.

Chapter 1

Introduction

Gesture recognition is an emerging human-computer interaction technology, and has recently been applied to games such as Kinect and Wii. However, compared to other human-computer interaction technologies such as speech recognition, the vision-based gesture recognition has not been actively applied to personal mobile phones or laptops. This might be due to the spatial requirements when performing gestures as well as sensitivity to background noise. Recently, a research group at ETH has developed a motion camera called Dynamic Vision Sensor Camera (DVS), which responds to pixels with temporal luminance differences [1], but its real-world applications have not been developed yet. In my research, I devised a new problem called speed-sensitive finger gesture recognition using a DVS camera aimed at exploring potential real-world applications to mobile devices.

As with other machine learning problems, extracting representative features is critical to the performance of a speed sensitive finger gesture recognition system. Especially when a new device, i.e., the DVS camera, is employed, the properties of this new device should be well considered during the feature extraction process. Gesture recognition using a general frame-based camera detects movement by comparing consecutive frames and uses detected changes as classification features. In contrast, when the DVS camera is used for gesture recognition, a frame-based comparison is not necessary since the movements of an object are detected at the hardware level. Instead, since the DVS camera outputs a long sequence of events sampled at a microsecond-level granularity, winnowing the number of events while not losing the overall gesture pattern is key for good performance.

The first part of my dissertation aims to extract a few representative features while preserving information about the patterns of the gesture through segmentation. To capture small but important movement changes, the local approach extracts features using similarity metrics between the neighboring events. Extracted features from this approach are called simple features. However, I expect that better performance can be achieved if higher-level abstract features can be constructed by transforming the locally extracted simple features to the difference space (e.g., from the Cartesian coordinate space to the polar space) or by finding the interactions between locally extracted simple features (e.g., the average speed of a gesture). In addition, higher level compound features may reduce the number of features, and thereby reduce the computational cost for learning and classification. Furthermore, they can serve as a knowledge discovery process by finding the underlying relationships between the features. The second part of my dissertation considers how to find a set of good simple and compound features.

1. Research Questions

My research aims to tackle two main research questions: (Q1) how to extract smaller number of representative local features from a long sequence of events and (Q2) how to construct abstract compound features from the locally extracted features (i.e., the simple features). Since the first research questions directly deals with the raw data from a DVS camera, the properties of the DVS camera must be carefully considered.

My approach to Q1 uses segmentation to extract local features; a sequence of gesture events are divided into segments from which simple features are locally extracted. Thus, the related sub-questions are (Q1-1) how to segment a sequence of events so that the patterns of gestures are well represented while being robust to noise and (Q1-2) how to extract features from those segments.

To construct high level compound features and to find a set of good simple and compound features, I use an evolutionary algorithm (EA) as a search algorithm. To use evolutionary algorithms, the representation method, reproduction operators, and fitness evaluation must first be well defined (Q2-1). The difficulties of using an EA for feature transformation include exploring the large search space and the risks of overfitting to the training data. Thus, I address the following sub-questions: (Q2-2) how to efficiently search for a good set of simple and compound features, and (Q2-3) how to reduce the overfitting problem. Finally, my research aims to provide qualitative analysis on the abstract features found in the EA (Q2-4). The main research questions and the related sub-questions are summarized in Table 1-1.

Table 1-1. The Research Questions

	RESEARCH QUESTIONS	**RELATED QUESTION**
Q1	How to extract simple features so that spatial and temporal information is well preserved?	Q1-1. How to perform segmentation? Q1-2. How to extract features from segments?
Q2	How to construct higher-level compound features?	Q2-1. How to design EA to find a good set of simple and compound features? Q2-2. How to make the search process more efficient? Q2-3. How to reduce overfitting? Q2-4. Can compound features detect useful underlying relationships between the features?

2. Dissertation Organization

Chapter 2 describes the properties of the DVS camera in detail, and summarizes related work on gesture recognition; segmentation; feature selection and construction; and the evolutionary-based feature extraction methods. Chapter 3 defines my problem domain of the speed-sensitive finger gesture recognition using a DVS camera and the two feature extraction problems, namely local and global feature extractions. In Chapter 4, I propose a segmentation method to locally extract features, and two initial segmentation schemes (time-based and event-based) are evaluated. I quantitatively evaluate the two schemes in terms of performance accuracy of classifiers and qualitatively analyze the segmented gesture data. Chapter 5 proposes an evolutionary algorithm for feature construction and selection to extract features from a more global perspective. The representation, reproduction operators, and fitness evaluation methods specified to my gesture recognition problem are proposed and evaluated in terms of prediction accuracy and the complexity of the extracted features. I also analyze contextual meaning of the extracted features. Finally, Chapter 6 discusses the contributions of my research, and concludes the paper with an overview of future work.

Chapter 2

Related Work and Backgrounds

This Chapter describes the properties of a Dynamic Vision Sensor (DVS) camera, and reviews related work on my main research questions. The first Section describes the properties of the DVS camera, and Section 2 describes the overview of previous work on gesture recognition to understand where my research is located in the field of gesture recognition. In Section 3, the hidden Markov model (HMM) [2], which is the most frequently used learning algorithm for gesture recognition, is summarized. Since segmentation is used to extract simple features locally, Section 4 describes related work on time-series data segmentation. Section 5 describes feature selection and construction methods, and finally Section 6 reviews previous work on the evolutionary algorithms for selecting and constructing compound features.

1. Dynamic Vision Sensor (DVS) Camera

Dynamic Vision Sensor (DVS) camera was recently developed by Delbruck's group at ETH in Zurich [3]. As opposed to conventional frame-based cameras, the DVS camera responds to pixels with temporal changes of luminance intensity. For example, if a moving object is brighter than the background, the pixels where the object appears (i.e., the pixels getting brighter) have on-events, and the pixels where the object disappears (i.e., the pixels getting darker) have off-events. The pixels with no temporal change in luminance have no event. The output of DVS camera is a sequence of events, where each event is represented by i) x and y coordinates within a frame having 128 by 128 pixels, ii) timestamp at a microsecond granularity, and iii) the type of

event (on or off). Figure 2-1 shows an example of a set of events that occur within a 20ms interval, where white and black points represent on- and off- events, respectively. This figure is depicted by jAER software that was also developed by the same team [1].

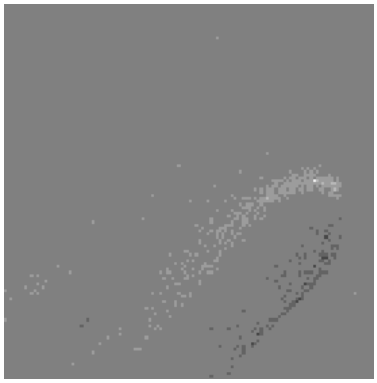
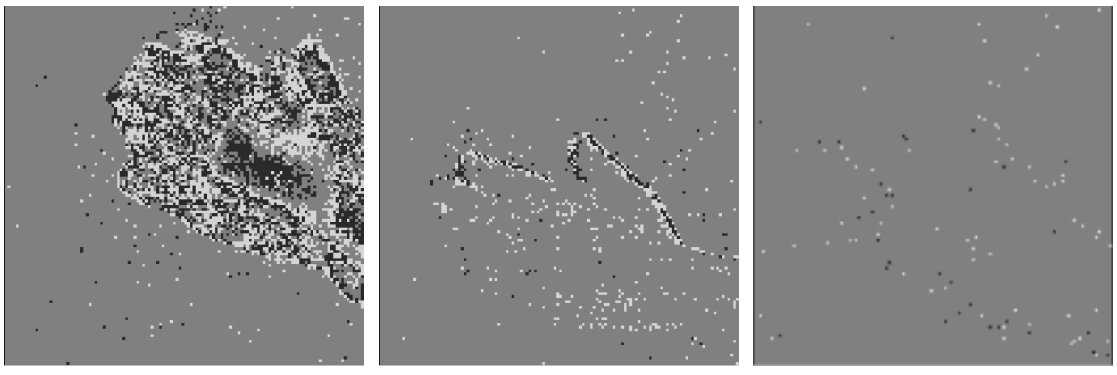


Figure 2-1. A set of events for a moving finger in 20 ms

Figure 2-2 show a sequence of frames of a hand when it stops moving after the downward movement. Each frame size is 20ms. During the hand movement, which is shown in Figure 2-2 (a), there are many events in a frame, but the number of events reduces as the hand stops moving; and finally only a few events occur when the hand does not move (Figure 2-2 (c)).



(a) (b) (c)

Figure 2-2. A sequence of frames of a moving hand

In previous work on the hand gesture recognition systems using conventional cameras, a still object is generally detected by using color information, and its movement is detected by comparing consecutive frames [2, 4, 5]. When a DVS camera is used, detecting still objects becomes nontrivial, because the DVS camera neither outputs an event for the still object, nor provides color information. However, motion detection becomes much easier computationally, since temporal changes of the pixels are detected at the hardware level. In addition, by producing events asynchronously, the DVS camera can have varying time intervals, thereby preserving accurate time information about events.

2. The Gesture Recognition Problems

Previous work on gesture recognition systems can be divided into two categories: vision-based and sensor-based systems [4]. Whereas vision-based gesture recognition depends solely on the camera [6-9], a sensor-based system requires users to wear a glove or other sensor device [6, 10, 11]. Although wearing additional devices can enhance the performance accuracy of the recognition system, wearing devices can be cumbersome and costly [12]. On the other hand, a vision-based gesture recognition system requires a camera only, but achieving the necessary performance accuracy is challenging due to the system's sensitivity to background noise. Since I use DVS camera only, this research can be categorized as the vision-based gesture recognition system.

Depending on the application of the gesture recognition, different parts of the body are used to perform a gesture. Body or hand gesture recognition has been studied for controlling remote devices [13, 14], sign language recognition [15], letter input system [16], etc. Although widely studied, gesture recognition has not been actively used to personal devices such as mobile

phones or laptops. This might be due to the spatial requirements when performing gestures as well as sensitivity to background noise. Recently, An et al. [11] and Tsukada and Yasumura [10] studied finger gesture-based interface for mobile devices, but work of Tsukada and Yasumura was the sensor-based approach. An et al. [10] and Oh et al. [17] used the vision-based finger gesture recognition using the camera on the mobile phone, but the number of gestures the user can perform was very limited to control a mobile device. As I will discuss in the next Chapter, this paper devises the vision-based speed-sensitive finger gesture recognition system, which can provide more number of gestures, and more importantly, which can provide additional semantics such as go quickly or slowly. To the best of my knowledge, the vision-based speed-sensitive finger gesture system has not been studied before.

There are two main problems in gesture-recognition systems: *gesture spotting* and gesture classification. *Gesture spotting* is the task of finding the starting and ending point of a gesture [2, 4, 18]. Kang et al. [18] describes *gesture spotting* as the process that distinguishes meaningful gestures from unintentional movements. The spotted gesture is used for gesture classification. Those two problems are tightly related, since enhancement in gesture spotting can improve the performance accuracy of gesture classification by providing more clear gesture data to a classifier, and sometimes they are performed in a single task [2]. My research simplifies the gesture spotting process and more focuses on gesture classification.

3. Hidden Markov Model for Gesture Recognition

Vision-based gesture recognition systems have often used hidden Markov model (HMM) classifiers with promising results [2, 4, 5, 19]. The HMM [19] operates by requiring a user to specify the number of hidden states. The HMM learns an initial state distribution, state transition probabilities, and emission probabilities during training. Following the notations used in [19], let

$\boldsymbol{S} = \{S_1, S_2, \ldots, S_M\}$ be the set of hidden states where M is the number of hidden states that the user specifies, $\boldsymbol{V} = \{v_1, v_2, \ldots, v_N\}$ be the set of possible N outcomes, which can be either discrete or continuous, and q_t is the state at time t. The initial state distribution is the probability of being at state S_i at the initial time ($t = 1$), which is denoted as $\pi_i = P[q_1 = S_i]$. The state transition probability is the probability of transferring to state S_j at time $t + 1$ from state S_i at time t, which is represented as $A = \{a_{ij}\}$, where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, and $1 \leq i, j \leq M$. The emission probability is the probability of emitting observation symbol v_k given a state S_j, which is denoted as $B = \{b_j(k)\}$ where $b_j(k) = P[v_k | S_j]$. In general, the HMM is specified by $\lambda = (A, B, \pi)$, and the Baum-Welch method is used to learn those probabilities [20].

Gesture-recognitions systems often employ left-to-right HMMs. A left-to-right HMM starts from an initial state and terminates at the final state after traversing intermediate hidden states. During traversing intermediate states, the left-to-right HMM only allows forward transition toward the final state. The left-to-right HMM has been shown to perform better than the fully connected model, which allows transitions from one state to any other state, by restricting structure of an HMM and reducing the number of variables to learn [16]. If there are C classes, a HMM (λ_c) is built for each class c. To classify a new sequence of observations, $\mathbf{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_{N_{obs}}\}$, where $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{N_{obs}} \in \boldsymbol{V}$, is fed to each HMM and the probability that the sequence is sampled from the HMM ($P[\mathbf{O}|\lambda_c]$, where $c \in C$) is calculated. Finally, a new data is classified as a class based on the largest $P[\mathbf{O}|\lambda_c]$.

As a concrete example of using an HMM in hand-gesture recognition with a frame-based camera, an observation can be the temporal difference of x- and y- coordinates of the center of a moving hand [2, 4]. To extract the temporal difference of x- and y- coordinates, or feature vectors, first the hand region and then the center of the hand must be detected using color information and prior knowledge about hand shape [21]. Next a trajectory of the hand, i.e., a sequence of the

temporal differences of *x*- and *y*- coordinates, is found by comparing the hand centers for consecutive frames [22].

When HMM is used for gesture recognition using DVS camera, the frame-based comparison will not be required and the sequence of events might be directly fed to the HMM. However, simply using the raw data is computationally infeasible due to the large number of events produced by the DVS camera, and thus a method to extract important features is needed.

4. Segmentation as Feature Extraction

To tackle my first research question, I use the segmentation to extract representative points from a sequence of events. In time-series data, segmentation refers to the task of dividing the data into intervals w.r.t. time, while discretization is the task of dividing the data w.r.t. value to find recurring states, as shown in Figure 2-3 [23]. According to [24], segmentation methods can be divided into 3 main approaches: sliding windows, top-down and bottom-up approaches. The sliding window approach segments the data if a segment exceeds a certain threshold; The top-down approach starts with considering the whole sequence as one segment and repeatedly partitions the sequence into smaller segments; The bottom-up approach starts with small segments and repeatedly agglomerates similar segments based on a certain distance measure. The sliding window segmentation can be performed online before looking at the whole data sequence, while both the top-down and the bottom-up approaches are off-line methods, since they cannot be performed without obtaining the whole sequence data.

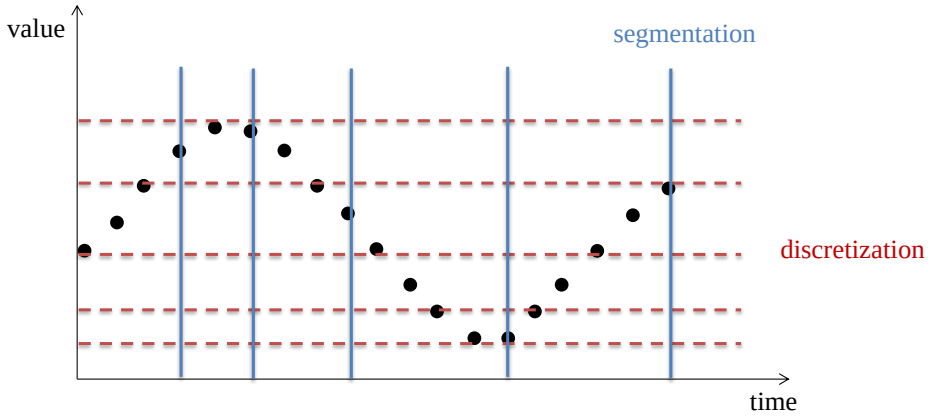


Figure 2-3. Segmentation and discretization

A key aspect of both the top-down and bottom-up methods is specifying the distance measure between two segments. One possibility to measure a distance between two segments is to perform a linear regression for each segment before and after the two segments are combined and to measure the amount of increase in the sum of errors of the linear regressions after the two segments are joined. Keogh et al. showed that the bottom-up approach performs the best and scales linearly with the size of the data, while top-down approach performs better than the sliding window approach but does not scale well [24]. As I will discuss in Chapter 3, the bottom-up approach is employed in this research, since the whole sequence of events is available for segmentation and the scalability of the segmentation is important due to the large number of events in a gesture sequence.

5. Feature Selection and Construction

In machine learning and data mining, there are mainly two categories of feature manipulation processes: feature selection and feature construction (or also called feature transformation, feature extraction or feature generation). Feature selection is the task of selecting

a subset of features from the original vector of features, while feature construction is the task of generating new features by transferring features into the new feature space. Unfortunately, there is no unique term to reference the latter feature manipulation process, but the latter process is called feature construction in the rest of this paper. The feature selection or construction can be implicitly performed as part of learning algorithms such as decision tree and support vector machine, or explicitly performed as a pre-process before the learning process [25, 26]. Since the explicit feature selection and construction method can be applied to any learning algorithm, and the extracted features can be analyzed, I use the explicit approach.

With regard to feature subset selection, the stepwise forward feature subset selection and backward feature elimination are greedy heuristic methods that add (or remove) a feature, that seems to be most promising (or useless) one at a time [27]. Although simple and fast, these greedy methods are generally insufficient to select good features. To find better feature sets, search methods have been widely applied. To search for a good feature subset, a search method and a method to evaluate a candidate solution should be determined. Since the search space is exponentially increasing as the number of original features increases, exhaustive search is prohibited and randomized search such as evolutionary algorithm is commonly used [22, 25, 29].

The intuitive way of evaluating a candidate solution, i.e., a feature set, is to estimate the accuracy of a classifier when the feature set is used for learning. This is so-called the wrapper approach, where an actual learning algorithm is incorporated during evaluation [25, 30]. Another way of evaluation is so-called the filter approach, where the statistical properties of data are used for evaluation without incorporation of any actual learning algorithm [25, 31-34]. Although the wrapper approach is computationally more expensive, it generally outperforms the filter approach [25]. Recently, hybrid methods that combine the wrapper and filter methods are proposed to reduce the computational cost of wrapper approach [35-38].

Feature construction methods aims to transfer the original features to a new feature space. The well-known method for feature transformation is principal component analysis (PCA), which linearly transforms the original data by preserving the variance in the data as much as possible [28], or the Fourier transform, which transfers data from the time domain to the frequency domain [39]. Slightly different from the conventional feature transformation methods, which transform all features from one space to the other, feature construction methods studied in [40-46] and in this research generate high level compound features by applying a sequence of mathematical (non-linear) transformations to simple features.

My research considers both feature selection and construction, since my goal is to find a set of simple features and compound features constructed from simple features. The selected set of simple and compound features may contain features from different spaces. For example, one compound feature can be generated by subtraction of the two features, which is a liner transformation, while the other compound feature can be generated by transferring to the angle space from the Cartesian product space.

6. Evolutionary-based Approach for Feature Selection and Construction

To search for a set of good simple and compound features, this paper uses the evolutionary algorithm as a search method. The evolutionary algorithm (EA) is a biologically-inspired algorithm where a population of candidate solutions called individuals or chromosomes evolves through selection and genetic reproduction processes [47]. The EA evolves by preferring fitted solutions during the selection process, and by exploring the search space through the reproduction operators such as mutation and crossover. The EA has been applied to various problems including optimization, machine intelligence, and biology, and shown successful results

[48]. Feature selection and construction is one of machine learning applications that the EA has been applied and shown promising results [29, 49-54].

In feature selection using EA, an individual is often represented by a string of bits whose size is equal to the number of original features as shown in Figure 2-4(a) [29, 49, 50, 55]. Each bit determines whether to include the corresponding feature or not. If the *i*th bit is 1, the *i*th feature is used for learning, while if the *i*th bit is 0, the ith feature is not included to the feature set which is used for learning. Reproduction and recombination are applied as in the conventional genetic algorithm (Figure 2-4(b, c)). Each individual is evaluated based on a filter, wrapper approach or hybrid method.

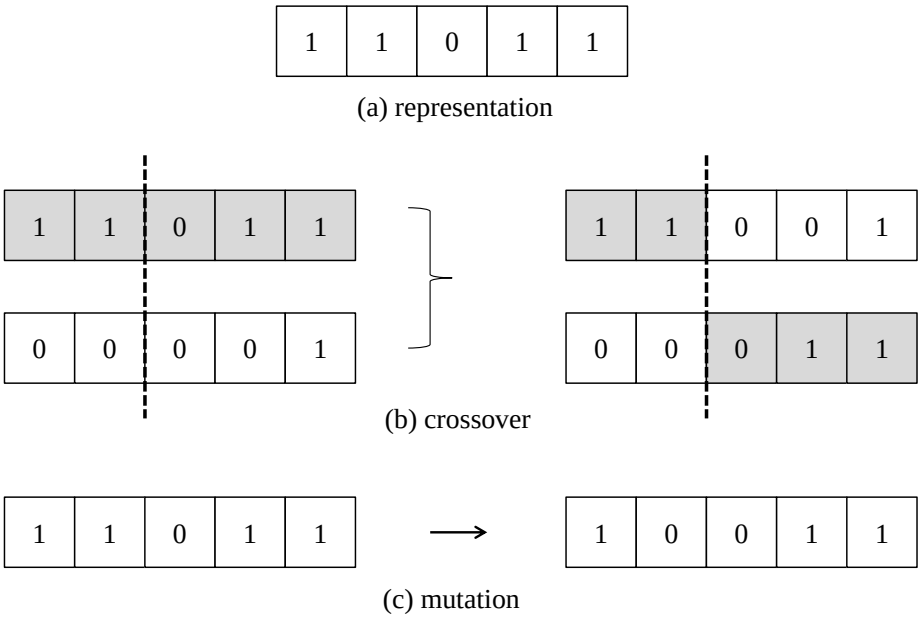
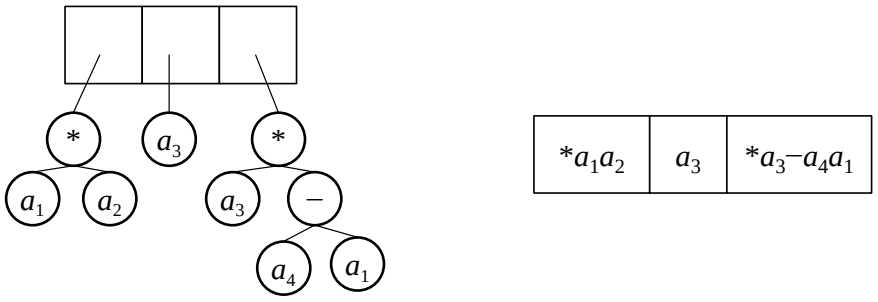


Figure 2-4. The EA for feature subset selection

Recently, EAs have also been applied to construct higher level compound features [36, 41, 44-46, 56-58]. Generally, each individual is represented by a set of features, where a feature can be either an original or a compound feature. A feature in the individual is represented by a

syntax tree or an equivalent prefix notation as shown in Figure 2-5. Internal nodes of the tree in Figure 2-5(a) are predefined transformation functions, while leaves are original simple features. It is similar to the conventional genetic programming [59, 60], where an individual is represented by a syntax tree and has an ability to classify examples based on the output of the tree. However, the EA-based feature selection and construction is different from the general genetic programming (GP) in that each individual is composed of a set of expression trees and an individual is not an actual classifier that can classify examples. As discussed in the previous Section, an individual can be evaluated using a filter or a wrapper method. Various representation-specific crossover and mutation methods have been defined. For example, in Ritthoff et al [36], crossover is performed by swapping some compound features, i.e., whole trees, of the two parents, and mutation is performed by determining whether or not to include a feature in the individual. In Krawiec [46], crossover is performed by exchanging subtrees of the two parents, and an individual is mutated by generating a new subtree at the randomly selected node. To be less destructive, they also propose the extended method, where each individual keeps an additional set of features called a hidden set. The hidden set does not evolve, but keeps the most valuable features that an individual has seen so far.



(a) tree representation (b) prefix notation

Figure 2-5. The representation of chromosome for feature construction

One of the challenging problems of the evolutionary-based feature construction and selection is overfitting, where the enhancement of the performance to the training data degrades the performance to the unseen data. The formal definition given by Michelle [61] is as follows:

> " Given a hypothesis space $\boldsymbol{H}$, a hypothesis $h \in \boldsymbol{H}$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$ such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances. "

The common approaches to reduce overfitting is (1) to use a validation set and (2) to limit the complexity of a feature set [62, 63]. When the validation set is used, the fitness of an individual is evaluated based on the training data, but the final solution that the EA returns is the solution that performs the best on the validation set, which is separated from the training data. The rationale behind the second approach is that the simpler model is preferred to more complex models by having less generalization error when they have the same predictive accuracy, which is known as Occam's Razor [64]. This is also tightly related to reducing tree-bloat in Genetic Programming [65]. To have simpler features, previous work (1) adds a complexity term to the fitness function, (2) limits the size of the individual, e.g., the numbers of operators and simple features in the individual, or (3) limits the crossover operator to be performed between the solutions of the similar size, which is called the fair-size crossover. However, Silva et al. showed that overfitting can still occur with simple models [65] and Paris et al. experimentally showed that overfitting is hardly reduced with the second and third approach [63].

Chapter 3

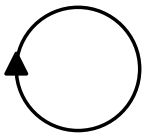
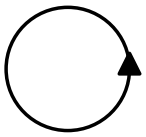
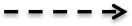
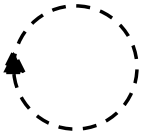
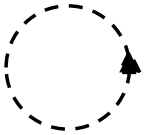
Speed Sensitive Gesture Recognition

This Chapter describes the properties of a Dynamic Vision Sensor (DVS) camera, and clearly describes my problem domain.

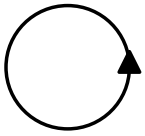
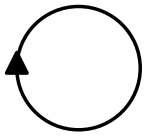
1. Finger Gesture Recognition Using a DVS camera

As discussed in Chapter 1, gesture recognition has not been actively applied to personal mobile devices partially due to the spatial requirements when performing gestures as well as sensitivity to background noise. In my research, I simplify evaluation by using a finger gesture recognition system where only one finger is used to perform gestures. While simplified, this approach still has potential real-world applications as one-finger gestures require less space and are less tiresome for users than hand gestures [10, 11]. In addition, one-finger gestures can be less prone to background noise, because other parts of our body are less likely to be shaken during making gestures.

To provide the potential for additional semantics such as go quickly or slowly, I use speed-sensitive finger gestures. Having the speed sensitive property can also increase the number of gestures that can be performed by a single finger. This research specifically uses eight finger gestures where four pairs of gestures share the same trajectories but differ in speed as shown in Figure 3-1(a). I assume the tip of finger is pointing at the front of the camera. Thus, the direction of the horizontal movement that the camera observes is opposite to the actual direction as shown in Figure 3-1(b). For example, the *x*-coordinates of events will decrease as we move our finger from left to right (movement 2).

slow				
class #	1	2	3	4
	slow right-to-left	slow left-to-right	slow clock-wise	slow counter clock-wise
fast				
class #	5	6	7	8
	fast right-to-left	fast left-to-right	fast clock-wise	fast counter clock-wise

(a) The actual finger movements

slow				
class #	1	2	3	4
fast				
class #	5	6	7	8

(b) The movements observed by the camera

Figure 3-1. The finger gestures

As discussed in Chapter 2, the gesture recognition is generally performed in two steps: gesture spotting and gesture classification. The gesture spotting is the task of finding the starting and ending point of a gesture while gesture classification is a task of determining the class of the spotted data [2, 18]. Since the major focus of this research is feature extraction for gesture classification, I simplify the gesture spotting problem by assuming that there is a short idle time between the consecutive gestures. If there is no movement during the small time gap, the DVS camera will produce a few (noisy) events, and thereby making it easier to predict the starting and ending point of a gesture. However, since completely separating the non-gesture events from the gesture events is infeasible, the spotted data can still contain non-gesture events such as a few noisy events just before and after the main gesture. The proposed feature extraction methods described in Chapter 4 and 5 assume that gesture is already spotted with a simple gesture spotting process, and the sequences of gesture events that are ordered in time are available. My simple gesture spotting process will be described briefly in Chapter 4 as part of the experimental setup process.

2. Local and Global Feature Extraction

Although motion is detected at the hardware level, the raw data from the DVS camera cannot be directly used to learn classifiers due to the large number of events in a gesture and noisy events. For example, the simple gesture of moving an index finger from right to left slowly produces more than 30,000 events. Thus, feature extraction is inevitable to make the gesture recognition computationally feasible and to achieve necessary performance accuracy. The feature extraction method should extract important patterns of the movement with a small number of representative points. Thus, my first research question can be stated as: given a sequence of the events for a gesture, how to extract the important patterns of the gesture using a small number of

points while being robust to noise. To extract fine-grained movement of the finger, the feature extraction needs to be performed from a local perspective by considering the similarity between the neighboring events. The extracted features are called simple features in the rest of the paper.

The second research question aims to find a set of simple and compound features each of which is constructed by applying a sequence of mathematical transformations to simple features using an EA. Through the evolutionary-based feature selection and construction algorithm, I expect to improve the performance accuracy of the classifiers and to reduce the computational cost by using a smaller number of features. More importantly, the compound features are expected to discover the underlying relationships between the features. In contrast to the simple feature extraction method, this approach extracts features from a more global perspective.

These two main problems are depicted in Figure 3-2. The black box at the bottom contains a sequence of events for a gesture, where the arrow indicates the moving direction of a finger. The red box next to the black box represents the feature space after the local feature extraction (Q1 in Table 1-1). Lastly, the blue box at the top shows the compound features constructed from the simple features (Q2 in Table 1-1). Note that the raw events and simple features are in the Cartesian coordinate space, but the compound features are not necessarily in the Cartesian coordinate space.

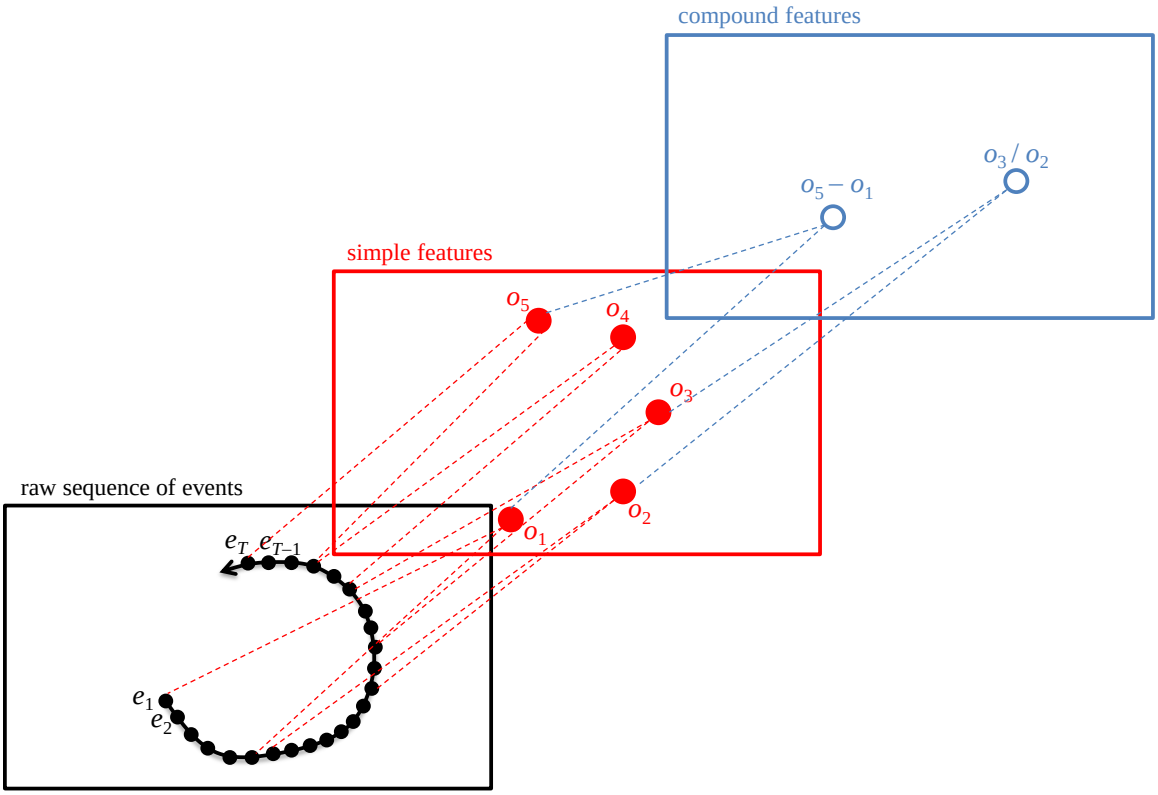


Figure 3-2. Feature extractions from the raw data

Chapter 4

Local Feature Extraction Using Segmentation

As mentioned in Chapter 2, building a classifier using the raw gesture data is computationally very expensive and prone to noise. To tackle this problem, I segment the data as a feature extraction process. Features are extracted in three steps: (1) divide the raw sequence of events into smaller segments; (2) find a representative point for each segment; and (3) extract features from the representative points. In the first two Sections, I propose segmentation methods for a sequence of gesture events, and analyze the complexity of the proposed algorithm. The third Section discusses extracting representative points and features from which classifiers are built. Finally, I show a simple method for data collection, i.e., the gesture spotting process, and show the preliminary results. This Chapter concludes with a brief summary.

1. Data Segmentation

Since gesture classification is performed after the starting and ending points of the gesture are found, a whole sequence of events for a gesture is available to the segmentation process. This allows us to use the offline segmentation approaches such as bottom-up or top-down segmentations. In addition, since the number of events in the gesture is very large, my method uses the bottom-up segmentation approach, which scales well. A sequence of events for an input gesture is denoted as $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_T\}$, where $\boldsymbol{e}_i$ is the ith event represented as a vector of x-coordinate (e_{ix}), y-coordinate (e_{iy}), and timestamp (e_{it}) as shown in Equation 4-1, and T is the index of last event. To simplify my segmentation process, my segmentation method does not take event type (on- or off-) information into consideration. However, event type information

can be used for the segmentation process by simply applying the proposed approach to on- and off-events separately. My segmentation method first segments $\boldsymbol{E}$ into the user-specified N_{init} segments, $\boldsymbol{s}_1, \boldsymbol{s}_2, .., \boldsymbol{s}_{N\text{init}}$, where each $\boldsymbol{s}_k$ represents a sequence of events in the kth segment. Then, at each step, two neighboring segments that have similar event distributions are selected to be agglomerated. The reason behind selecting segments with similar event distributions is that events in a gesture are likely to be continuous and the chance of having sudden jumps during the movement is very small. The segment augmentation process is repeated until the user-specified number of segments, which is denoted as N_{final}, is obtained. The overall process is depicted in Figure 4-1.

$$\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \dots, \boldsymbol{e}_T\}$$
$$, where\ \boldsymbol{e}_i = \begin{bmatrix} e_{ix} \\ e_{iy} \\ e_{it} \end{bmatrix}$$

Equation 4-1

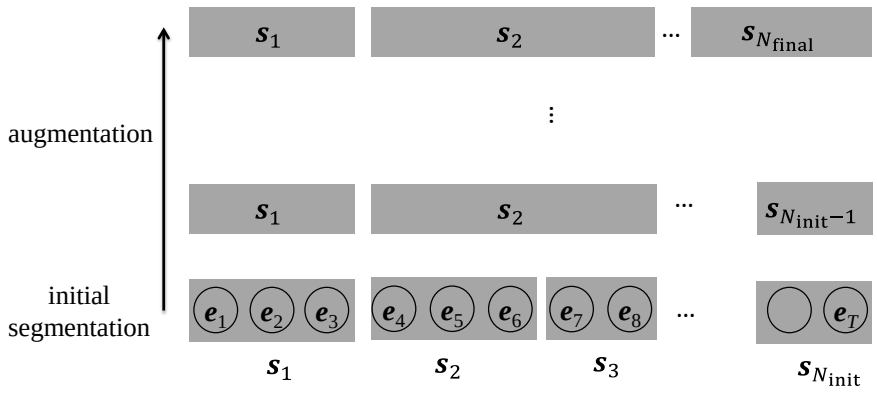


Figure 4-1. Event Segmentation and Augmentation

To detect fine-grained movement of a gesture, N_{init} should be large enough. However, the segmentation methods become more sensitive to noise as N_{init} gets larger (Hypothesis 1). When a feature vector is extracted from each final segment, an increase in N_{final} increases the number of feature vectors, thereby allowing the trajectories of a gesture to be expressed at a more fine-grained level. However, with too large N_{final}, final segments may include unnecessary, redundant, and noisy information (Hypothesis 2).

The initial segmentation can be performed in two ways: time-based and event-based. The time-based initial segmentation divides $\boldsymbol{E}$ into segments of the same time interval, whereas the event-based segmentation divides $\boldsymbol{E}$ into segments containing the equal number of events. Note that the time-based initial segmentation makes segments similar to the frames in a conventional camera, while my event-based segmentation is motivated by the nature of the data produced by DVS camera.

I hypothesize that the event-based segmentation is less prone to having a small number of noisy events before and after the main gesture (Hypothesis 3). Figure 4-2 shows y-coordinates and timestamps for the events that occur when the finger is moving from left to right. Since it is infeasible to exactly locate the starting and ending points of the gesture, the spotted gesture contains a small number of sparse events that occur just before and after the main gesture. When the time-based segmentation is performed, a small number of events will comprise the initial segments, and these segments may persist through the end of the augmentation process, since the event distributions of the initial segments may not be similar to those of the neighboring segments for the main gesture.

However, since the event-based segmentation requires a relatively large number of events to constitute an initial segment, the impact of noisy events to the event distribution of the segment can be potentially reduced by other gesture events in the same segment, and thereby making the

distribution of the segment containing noisy events similar to the event distributions of neighboring segments.

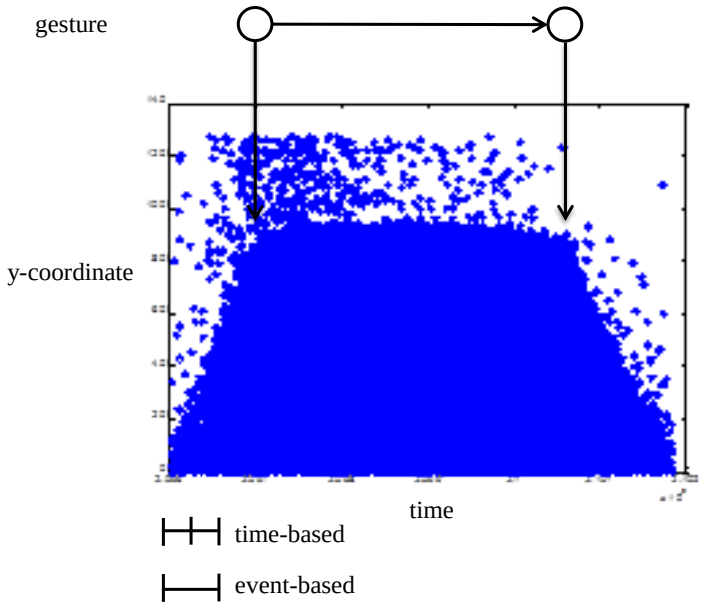


Figure 4-2. The event-based and time-based segmentations

I also hypothesize that the event-based initial segmentation can detect changes of gesture better than the time-based segmentation (Hypothesis 4). One of the reasons might be that since the speed of the finger is not constant over a gesture, segmenting a gesture with a constant time interval can overlook multiple but important changes that occur within a short time interval. On the other hand, the event-based initial segmentation method can detect those changes by segmenting gesture events based on the number of events. The four hypotheses are summarized in Table 4-1.

Table 4-1. Hypotheses

HYPOTHESIS	EXPLANATION
1	The segmentation method becomes sensitive to noise as N_{init} gets larger.
2	A larger N_{final} enables the pattern of gestures to be expressed at a more fine-grained level. However, too large N_{final} can include unnecessary and redundant information.
3	The event-based segmentation is less prone to having sparse noisy events just before and after the main gesture.
4	The event-based initial segmentation can detect changes of a gesture trajectory better than the time-based segmentation

To get the event distribution for each segment, I assume that events are sampled from a multivariate Gaussian distribution for the corresponding segment. The event distribution for the kth segment, $f_k(\boldsymbol{e})$, is obtained by estimating the mean vector ($\boldsymbol{\mu}_k$) and covariance matrix ($\boldsymbol{\Sigma}_k$) of the events within the segment, which can be written as

$$f_k(\boldsymbol{e}) = \frac{1}{(2\pi)^{rank(\boldsymbol{\Sigma}_k)/2}|\boldsymbol{\Sigma}_k|^{1/2}} exp\left(-\frac{1}{2}(\boldsymbol{e}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k{}^{-1}(\boldsymbol{e}-\boldsymbol{\mu}_k)\right)$$

Equation 4-2

, where $|\boldsymbol{\Sigma}_k|$ represents the determinant of $\boldsymbol{\Sigma}_k$, and is a 3 by 3 matrix. The $\boldsymbol{e}$ and $\boldsymbol{\mu}$ are three dimensional vectors, which consists of *x*-coordinate, *y*-coordinate and timestamp.

For each pair of neighboring segments, the distance between the two distributions is calculated using Jeffrey's information [66]. Jeffery's information is a symmetric version of Kullback-Leibler (KL) divergence, which measures the dissimilarity between the two distributions *f* and *g* as

$$I^{KL}(f;g) = \sum_{i=1}^{m} f(x_i) log\left(\frac{f(x_i)}{g(x_i)}\right) \text{ for a discrete variable,}$$
$$= \int_{-\infty}^{\infty} f(x) log\left(\frac{f(x)}{g(x)}\right) dx \text{ for a continuous variable}$$
$$I^{J}(f;\ g) = I^{KL}(f;g) + I^{KL}(g;f)$$

Equation 4-3

, where each $I^{KL}(f; g)$ and $I^{J}(f; g)$ represents the KL divergence and Jeffrey's information between the two distributions *f* and *g*, respectively.

Although computing KL divergence is non-trivial in general, the assumption that the distribution of events is multivariate Gaussian makes the computation feasible and even provides a closed-form solution. The KL divergence between the distributions of the two neighboring segments, $f_1(\boldsymbol{e})$ and $f_2(\boldsymbol{e})$, is calculated as

$$I^{KL}(f_1; f_2) = \frac{1}{2}\left\{(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T {\boldsymbol{\Sigma}_2}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + tr\left({\boldsymbol{\Sigma}_2}^{-1}\boldsymbol{\Sigma}_1\right) - log\, det\left({\boldsymbol{\Sigma}_2}^{-1}\boldsymbol{\Sigma}_1\right) - d\right\}$$

Equation 4-4

, where *tr*() returns the trace of the input matrix and *d* is the dimension of $\boldsymbol{e}$. $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_1$ are the mean vector and the covariance matrix of $f_1(\boldsymbol{e})$, and $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_2$ are of $f_2(\boldsymbol{e})$. Jeffrey's information is

obtained by simply plugging Equation 4-4 to Equation 4-3. The smaller Jeffrey's information value, the more similar the two segments are considered.

2. Analysis of Time Complexity

Determining the time complexity of my segmentation method shows that calculating Jeffrey's information between neighboring segments at every augmentation step is the most expensive procedure. If d is the dimension of $\boldsymbol{e}$ and n_k and T are the number of events in the kth segment, and the number of events in a sequence, respectively, the time complexity of calculating a mean vector and a covariance matrix for each segment is $\mathbf{O}(dn_k)$ and $\mathbf{O}(d^2n_k)$, and the time complexity of calculating mean vectors and covariance matrices for all segments is $\mathbf{O}(dT)$ and $\mathbf{O}(d^2T)$, respectively. The time complexity of calculating the KL divergence between a pair of neighboring segments given mean vectors and covariance matrices is $\mathbf{O}(d^3)$. Thus, the overall time complexity of calculating mean vectors, covariance matrices and Jeffrey's information for all pairs of the adjacent segments is $\mathbf{O}(d^2T + d^3N_j)$, where N_j is the number of segments at the jth augmentation step. By repeating this process $N_{\text{init}} - N_{\text{final}}$ times, calculating Jeffrey's information in the segmentation process is $\mathbf{O}((N_{\text{init}} - N_{\text{final}})Td^2 + (N_{\text{init}} - N_{\text{final}})(N_{\text{init}} + N_{\text{final}})d^3)$, where the first term is the time complexity for calculating means and covariance matrices while the second term is the time complexity for calculating Jeffery's information between neighboring segments. Note that since d is relatively a very small constant compared to T in our application, d can be negligible when time complexity is considered.

I might be able to reduce the time complexity by considering that only one pair of segments is combined at each step. For each step, I might update Jeffrey's information only for the augmented segments and their neighboring segments. However, calculating mean vectors and covariance matrices to get Jeffery's information is still computationally expensive when the size

of segment (i.e., the number of events in the segment) is large. This problem can be solved by storing and updating the following calculations for each step: the number of events, n_k, $\boldsymbol{Q}_k = \sum_{\boldsymbol{e}_i \in \boldsymbol{s}_k} \boldsymbol{e}_i$ and $\boldsymbol{R}_k = \sum_{\boldsymbol{e}_i \in \boldsymbol{s}_k} \boldsymbol{e}_i \boldsymbol{e}_i^T$ for each segment $\boldsymbol{s}_k$, where $\boldsymbol{e}_i^T$ is the transpose of $\boldsymbol{e}_i$. If $\boldsymbol{s}_k$ and $\boldsymbol{s}_{k+1}$ are selected for augmentation, then the mean ($\boldsymbol{\mu}'$) and covariance matrix ($\boldsymbol{\Sigma}'$) for the new segment, $\boldsymbol{s}$', is recalculated as

$$
\begin{aligned}
&\boldsymbol{Q}' = \sum_{\boldsymbol{e}_i \in s'} \boldsymbol{e}_i = Q_k + Q_{k+1} = \sum_{\boldsymbol{e}_i \in s_k} \boldsymbol{e}_i + \sum_{\boldsymbol{e}_i \in s_{k+1}} \boldsymbol{e}_i \\
&\boldsymbol{R}' = \sum_{\boldsymbol{e}_i \in s'} \boldsymbol{e}_i \boldsymbol{e}_i^T = R_k + R_{k+1} = \sum_{\boldsymbol{e}_i \in s_k} \boldsymbol{e}_i \boldsymbol{e}_i^T + \sum_{\boldsymbol{e}_i \in s_{k+1}} \boldsymbol{e}_i \boldsymbol{e}_i^T \\
&n' = n_k + n_{k+1} \\
&\boldsymbol{\mu}' = \frac{1}{|n'|} \boldsymbol{Q}' \\
&\boldsymbol{\Sigma}'(l, m) = \frac{1}{|n'|} \boldsymbol{R}'(l, m) - \boldsymbol{\mu}'(l) \boldsymbol{\mu}'(m)
\end{aligned}
$$

Equation 4-5

,where $\boldsymbol{\Sigma}'(l, m)$ and $\boldsymbol{\mu}'(l)$ represent the (l, m)th entry of the matrix $\boldsymbol{\Sigma}'$ and the lth entry of the vector $\boldsymbol{\mu}$. In this way, I can trade off memory space to reduce the computational time for updating mean and covariance matrix at each step to $\mathbf{O}(d^2)$ and the overall computational time for updating Jefferey's information at each augmentation step to $\mathbf{O}(d^3)$, which is much more efficient than $\mathbf{O}(d^2T + d^3N_j)$. The computational time for calculating Jeffrey's information in the segmentation process is reduced to $\mathbf{O}(Td^2 + (2N_{\text{init}} - N_{\text{final}})d^3)$, which is linear to T.

3. Feature Extraction

After segmentation, I extract a representative point for each segment from which features are extracted. To determine each segment's representative point, I simply use the mean of events in the segment, although other possibilities include using the median from each event segment or

applying the *k*-means algorithm to each segment to get *k* representative points. Mean value is chosen because it does not require additional computation and is expected to represent the segment well since the mean vector is used as part of the segmentation process. I denote the representative point of the *k*th segment as $\boldsymbol{z}_k$, which is the same as $\boldsymbol{\mu}_k$ of the *k*th final segment. Since $\boldsymbol{z}_k$ is three dimentional, including an *x*-coordinate, *y*-coordinate, and timestamp, $\boldsymbol{z}_k$ is denoted as

$$\boldsymbol{z}_k = \begin{bmatrix} z_{kx} \\ z_{ky} \\ z_{kt} \end{bmatrix} for\ 1 \leq k \leq N_{final}$$

Equation 4-6

similar to Equation 4-1.

A sequence of $\boldsymbol{z}_k$ can be used to train and test classifiers such as HMM, but the same gesture trajectories with different starting points can result in different classification results. Thus, all $\boldsymbol{z}$s in a sequence are translated to the starting point $\boldsymbol{z}_1$ as shown in Equation 4-7. The translated representative points are used as a feature vector $\boldsymbol{o}$, which corresponds to an observation in our explanation of HMM in Chapter 2. Since the first representation point $\boldsymbol{z}_1$ is always zero after translation, it is excluded from the feature vectors. A sequence of feature vectors for a gesture is denoted as $\mathbf{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_{N_{obs}}\}$, where N_{obs} is $N_{\text{final}} - 1$, and this is graphically shown in Figure 4-3 (a). In the rest of the paper this feature extraction method is called REL COORD.

$$\boldsymbol{o}_k = \boldsymbol{z}_{k+1} - \boldsymbol{z}_1\ for\ 1 \leq k \leq N_{final} - 1$$

$$\boldsymbol{o}_k = \begin{bmatrix} \boldsymbol{o}_{kx} \\ \boldsymbol{o}_{ky} \\ \boldsymbol{o}_{kt} \end{bmatrix}$$

Equation 4-7

Although $\boldsymbol{O}$ can be directly fed into HMM, it cannot be fed into other classifiers, such as Bayesian networks [67] and decision trees, which are not designed to learn sequential data. In this paper, I call these classifiers non-sequential classifiers. To feed my sequential gesture data into non-sequential classifiers, the following data transformation is used: a sequence of extracted feature vectors is represented as a 2-D array where each column represents $\boldsymbol{o}_k$, and then serialized so that it can be represented as a 1-D array and used in such classifiers. Specifically, all feature vectors $\boldsymbol{o}$s are concatenated to get a single feature vector, as shown in Equation 4-8, where $[\boldsymbol{v}_1, \boldsymbol{v}_2]$ is the concatenation operator of two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Note that the effect of increasing N_{obs} to non-sequential classifiers and HMM are different; Increasing N_{obs} has a similar impact to increasing the number of instances for HMM, while it increases the number of features for non-sequential classifiers.

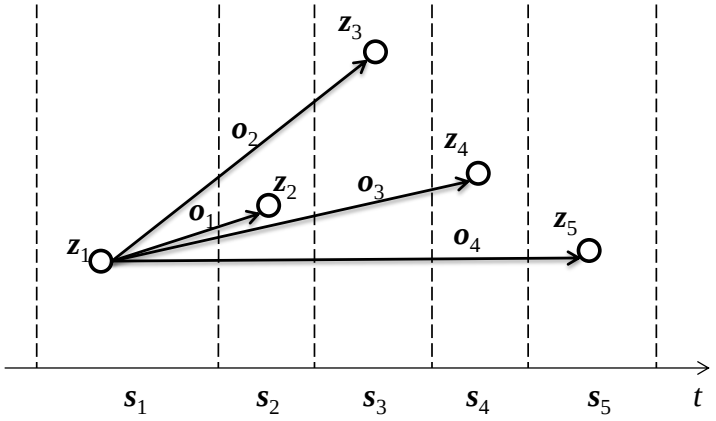
$$\begin{aligned}\boldsymbol{O} &= [\boldsymbol{o}'_1, \boldsymbol{o}'_2, \dots, \boldsymbol{o}'_{N_{obs}}] \\ &= [\boldsymbol{z}'_{1x}, \boldsymbol{z}'_{1y}, \boldsymbol{z}'_{1t}, \ \dots, \boldsymbol{z}'_{Nx}, \boldsymbol{z}'_{Ny}, \boldsymbol{z}'_{Nt}]\end{aligned}$$

Equation 4-8

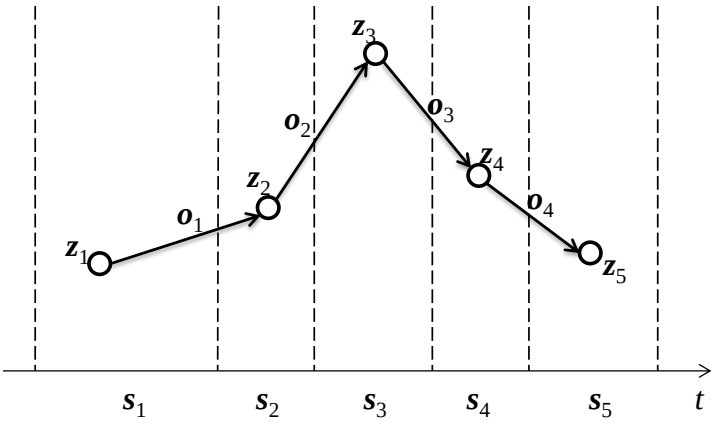
The second feature extraction method, which is called DELTA COORD, extracts the difference of the representative points of the adjacent segments, as described in Equation 4-9 and Figure 4-3 (b)

$$\boldsymbol{o}_k = \boldsymbol{z}_{k+1} - \boldsymbol{z}_k \ for\ 1 \le k \le N_{final} - 1$$

Equation 4-9



(a)



(b)

Figure 4-3. REL COORD and DELTA COORD

The last feature extraction method is VELOCITY, which divides $\boldsymbol{o}_{kx}$ and $\boldsymbol{o}_{ky}$ obtained from DELTA COORD by $\boldsymbol{o}_{kt}$ as described in Equation 4-10. VELOCITY shows promise in reducing the number of features and the correlation between the features.

$$\boldsymbol{o}_k = \begin{bmatrix} \frac{z_{(k+1)x} - z_{kx}}{z_{(k+1)t} - z_{kt}} \\ \frac{z_{(k+1)y} - z_{ky}}{z_{(k+1)t} - z_{kt}} \end{bmatrix} \; for \; 1 \leq k \leq N_{final} - 1$$

Equation 4-10

4. Data Acquisition

To collect finger gesture data, I record eight videos, each for one type of gesture. Within each video, a person repeats the same gestures ten times with a short time interval between the consecutive gestures. The finger is pointing toward the camera, which will be the case when the finger gesture recognition system is applied to mobile devices. Since each video contains multiple gestures of the same type, we need to find the start and the end points of each gesture, which is called *gesture spotting* [68]. Based on the property of DVS camera, which only produces events for the pixels with temporal difference, I check the number of events that occur within a certain time frame (e.g., 20 μs in this experiment) to determine whether an object is moving or not within that time frame. If the number of events in the frame, denoted as N_{frame}, is bigger than the predefined threshold N_{move}, the object can be considered to be moving, otherwise, not moving. If the finger is considered to be moving, events in the frame are collected until the finger is considered to be unmoving. The collected sequence of events is used for segmentation and feature extraction.

However, if a finger temporally stays unmoved for a few time frames during the movement, an incomplete sequence of events will be detected as a gesture. To avoid this problem, I allow a few unmoving frames to be included as part of a gesture. Specifically, three states are defined: unmoving, moving, and momentarily-still, where momentarily-still state is a buffer state that allows temporally short still period while making a gesture. The number of consecutive frames remaining in the momentarily-still state is denoted as c.

The gesture spotting process starts from the unmoving state. If N_{frame} is bigger than N_{move}, the current state transits to moving state, otherwise stays at unmoving state. If N_{frame} is larger than N_{move} at moving state, it stays at the same state, otherwise, transits to momentarily-still state and set c, which counts the number of frames staying at momentarily-still state, to one. If the N_{frame} is

bigger than N_{move} at momentarily-still state, the state transits to moving state. Otherwise, it stays at the same state and increases *c* by one unless *c* exceeds the number of allowable time frames at momentarily-still state, denoted as *max_count*. If *c* exceeds *max_count*, then it moves to unmoving state.

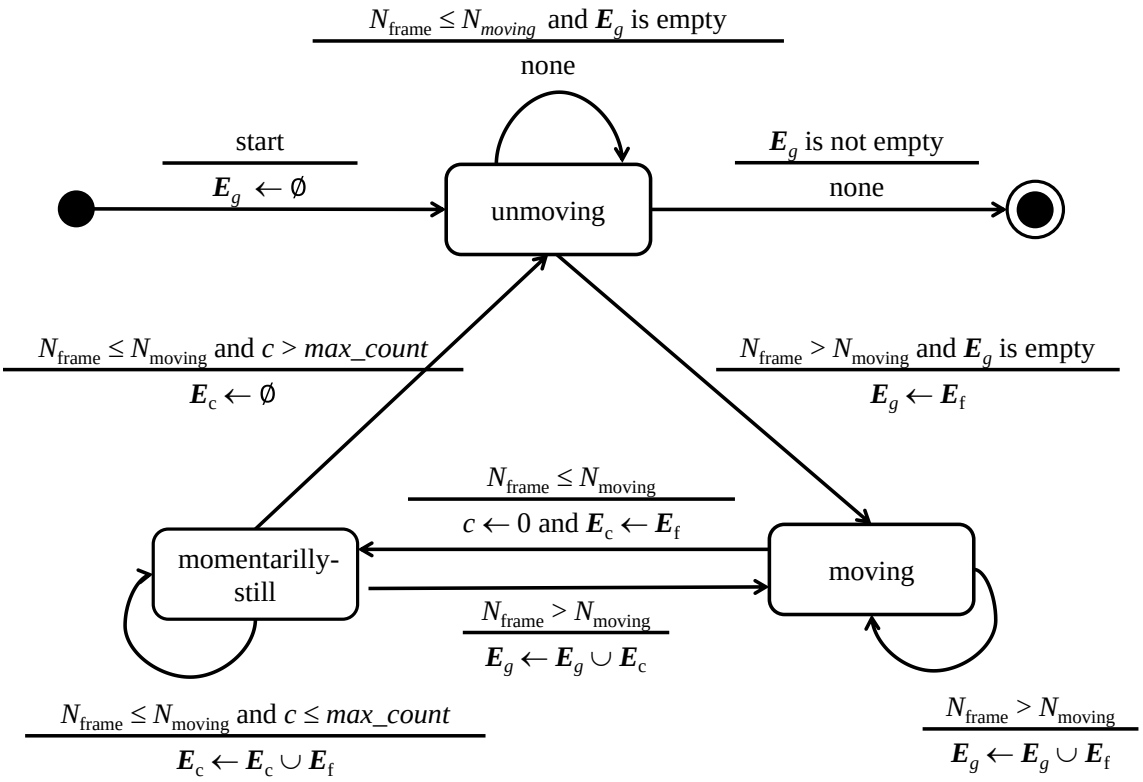


Figure 4-4. The state transition diagram of the gesture spotting process

I denote a sequence of events occurred in a frame as $\boldsymbol{E}_{\text{f}}$, and a sequence of gesture-related events as $\boldsymbol{E}_g$, which is initially an empty sequence. In the moving state, $\boldsymbol{E}_{\text{f}}$ is concatenated to $\boldsymbol{E}_g$, and in the momentarily-still state, $\boldsymbol{E}_{\text{f}}$ is temporally saved to a temporary location, $\boldsymbol{E}_{\text{c}}$. $\boldsymbol{E}_{\text{c}}$ is concatenated to $\boldsymbol{E}_{\text{g}}$ when the state transits from momentarily-still state to moving state. Finally,

when the state is back to the unmoving state, $\boldsymbol{E}_g$ is considered as a sequence of events of a gesture. Figure 4-4 shows the state transition diagram for the gesture spotting process. Note that since it is infeasible for our *gesture spotting* process to locate the exact starting point of the gesture, $\boldsymbol{E}_g$ can contain noisy events.

5. Experimental Settings

The proposed segmentation method is evaluated based on the performance of classifiers including decision tree, Bayesian network (BN) and hidden Markov model (HMM), which are trained and tested with the extracted features discussed in Section 3. Each of these classifiers is selected based on its ability to learn sequential data and relationships between features. The HMM is designed to learn sequential data; the BN is not designed to learn sequential data but it searches for dependencies between features; and the decision tree neither supports learning sequential data nor searches for feature dependencies, but it can implicitly represent feature relationships.

Weka data mining software [69] and the HMM open source package [70] are used for the classifiers. For the decision tree, I use J48 in Weka with the default parameter settings. For the BN, the K2 algorithm is used to learn the structure of the network, and a naïve Bayes as an initial network. The K2 algorithm first orders a set of features in a sequence, and then for each feature, it checks whether adding an edge from each of the preceding features to the current node improves the Bayesian score of the network. Although features can be ordered randomly, I follow the feature order in the data, since values for each *x*-coordinate, *y*-coordinate and timestamp are already ordered in time. Thus, the BN cannot have any time reversing edges. I experimentally allow each node to have a maximum of two parents. For the HMM, a left-to-right model with five

states is used. Five is a commonly used value in previous literature, and it experimentally seemed to work well for my application.

The data is normalized to avoid feature scale from dominating the classification process, and to make the calculation processes, such as calculating the determinant, more robust. In addition, data normalization can also enhance the performance of classifiers [12, 71]. The z-score normalization is performed after features are extracted, and each feature is normalized to have zero mean and unit variance. In time series data, normalization can be performed within a sequence without considering other sequences [71, 72]. However, if the normalization is done without considering the feature values of different gestures, gestures that share the same trajectories but differ in speed cannot be distinguished. Thus, feature values of different gestures are also considered in my normalization. The normalization of *x*-coordinates can be performed as shown in Equation 4-11, where $\boldsymbol{D}$ is training data consists of a set of gesture sequences before serialization; $\boldsymbol{O}$ is a sequence of feature vectors for a gesture in $\boldsymbol{D}$; N_{obs} is the number of feature vectors in a sequence; and $\boldsymbol{o}_x$' is the value of the *x*-coordinate, $\boldsymbol{o}_x$, after normalization. Similarly, the normalizations of *y*-coordinates and timestamps are performed.

$$\mu_x = \frac{\sum_{\boldsymbol{O}\in\boldsymbol{D}} \sum_{i=1}^{N_{obs}} \boldsymbol{o}_{ix}}{|D| \times N_{obs}}$$

$$\sigma_x = \frac{\sum_{\boldsymbol{O}\in\boldsymbol{D}} \sum_{i=1}^{N_{obs}} \boldsymbol{o}_{ix}^2}{\#\ of\ \boldsymbol{O}s\ in\ \boldsymbol{D} \times N_{obs}} - \mu_x^2$$

$$\boldsymbol{o}_x{}' = \frac{\boldsymbol{o}_x - \mu_x}{\sigma_x}$$

$$\text{, where } \boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_{N_{obs}}\} = \left\{ \begin{bmatrix} \boldsymbol{o}_{1x} \\ \boldsymbol{o}_{1y} \\ \boldsymbol{o}_{1t} \end{bmatrix}, \begin{bmatrix} \boldsymbol{o}_{2x} \\ \boldsymbol{o}_{2y} \\ \boldsymbol{o}_{2t} \end{bmatrix}, \dots, \begin{bmatrix} \boldsymbol{o}_{N_{obs}x} \\ \boldsymbol{o}_{N_{obs}y} \\ \boldsymbol{o}_{N_{obs}t} \end{bmatrix} \right\}$$

Equation 4-11

6. Experimental Results

A. Performance accuracy analysis

Figure 4-5, Figure 4-6, and Figure 4-7 compare the performance of the event-based and the time-based initial segmentations over different numbers of initial segments ($N_{init} \in \{100, 200, \ldots, 500\}$) and final segments ($N_{final} \in \{5, 10, \ldots, 75\}$) for each classifier when REL COORD feature extraction method is used. For the HMM, N_{final} is set to be equal or greater than 15. Recall that increasing/decreasing N_{final} has similar impact to increasing/decreasing the number of instances for the HMM. Empirically, the training data obtained from $N_{final} = 5$ and 10 seems to be too small to learn many parameters in the HMM. Each point in Figure 4-5, Figure 4-6 and Figure 4-7 is the average *F*-measure obtained from 10 five-fold cross-validations for each classifier. The 10 five-fold cross-validation means that the five-fold cross-validation is performed ten times with different random seeds, and *F*-measure is the harmonic mean of recall and precision [73]. The bigger the *F*-measure is, the better the performance of the classifier is. The *x*-axis represents N_{final}, and the *y*-axis represents the average *F*-measure. The smaller N_{final}, the more augmentation steps are needed. The dotted lines represent the event-based initial segmentation while the solid lines represent the time-based initial segmentation. The numbers followed by time or event in the legend represent N_{init}.

Overall the event-based segmentation outperforms the time-based segmentation. I believe that the event-based approach might be able to extract the overall patterns of gestures better than the time-based segmentation due to being less sensitive to sparse noisy events and by detecting important changes in trajectories of a finger gesture (Hypotheses 3 and 4). However, the performance difference of the event-based and time-based segmentations is not as significant in

the HMM as it is for the other classifiers. One possible reason is the HMM has an ability to implicitly combine noisy segments through the hidden states and emission probabilities.

Table 4-2, Table 4-3, and Table 4-4 show the result of statistical tests evaluating the performance of the event-based and the time-based segmentations. I test the performance comparison between the time-based and the event-based initial segmentations for each combination of N_{final} and N_{init} using paired two-tailed *t*-tests at the significance level $\alpha = 0.05$, and the statistically outperforming method is shown in bold. Each entry of the table is the average *F*-measure, which corresponds to a point in Figure 4-5, Figure 4-6, and Figure 4-7**.**

The super script * and + in Table 4-2, Table 4-3, and Table 4-4 show the results of statistical tests between $N_{\text{final}} = 20$ and $N_{\text{final}} = 40$, and between $N_{\text{final}} = 40$ and $N_{\text{final}} = 60$, respectively. The entry with the superscript means that the entry is statistically better than the corresponding entry with different N_{final} based on a paired two-tailed *t*-test at the significance level $\alpha = 0.05$. If the impact of different values of N_{final} to the performance of BN is considered in Figure 4-6 and Table 4-3, I find that there is a certain N_{final} where the performance is maximized for each N_{init}. For example, in the event-based segmentation with $N_{\text{init}} = 400$, the performance is maximized when N_{final} is 40, but the performance degrades slightly as N_{final} gets larger than 40. Recall that increases in N_{final} increases the number of features acquired from the data for the BN. As the number of features increases, the trajectories of a gesture can be expressed at a more fine-grained level, thereby enhancing the performance of the classifier. However, too large number of features may include non-important features, such as outliers and redundant features, which makes it hard for the BN classifier to learn the structure of the network (Hypothesis 2). In the HMM, on the other hand, the increase in N_{final} has the effect of providing more data to the classifier. Thus, the performance of the HMM improves and gets more robust as N_{final} increases as shown in Figure 4-7. The poor performance of the HMM with a small N_{final} might be caused by insufficient number of data to learn the parameters for the HMM.

When different settings for N_{init} are considered, the impact of N_{init} on the time-based segmentation is larger than the impact of N_{init} on the event-based segmentation in BN. The performance of the time-based segmentation degrades significantly when N_{init} exceeds a certain threshold (300), but the event-based segmentation is less sensitive to the large N_{init}. As discussed in Section 4.1, the sparse noisy events that occur just before and after the main gesture are more likely to form initial segments by themselves in the time-based segmentation than the event-based segmentation, and these segments are likely to persist all the way to the end of the segmentation process, thereby producing poor quality features (Hypotheses 1 and 3). Compared to BN, HMM seems to be robust to the different values of N_{init}. As stated previously, it may results from the ability to implicitly recombine the segments through hidden states.

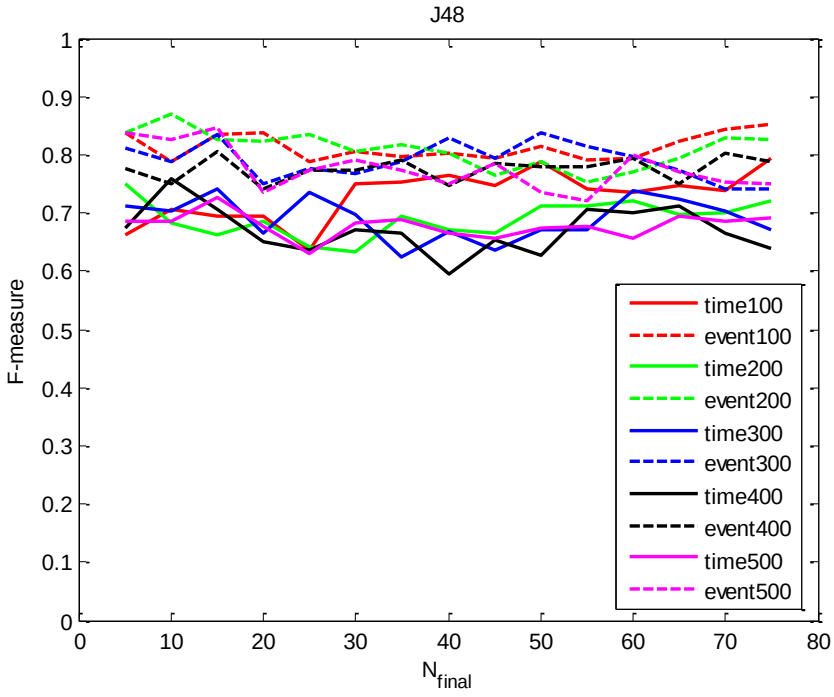


Figure 4-5. The average *F*-measure of the J48 with different N_{init} and N_{final}

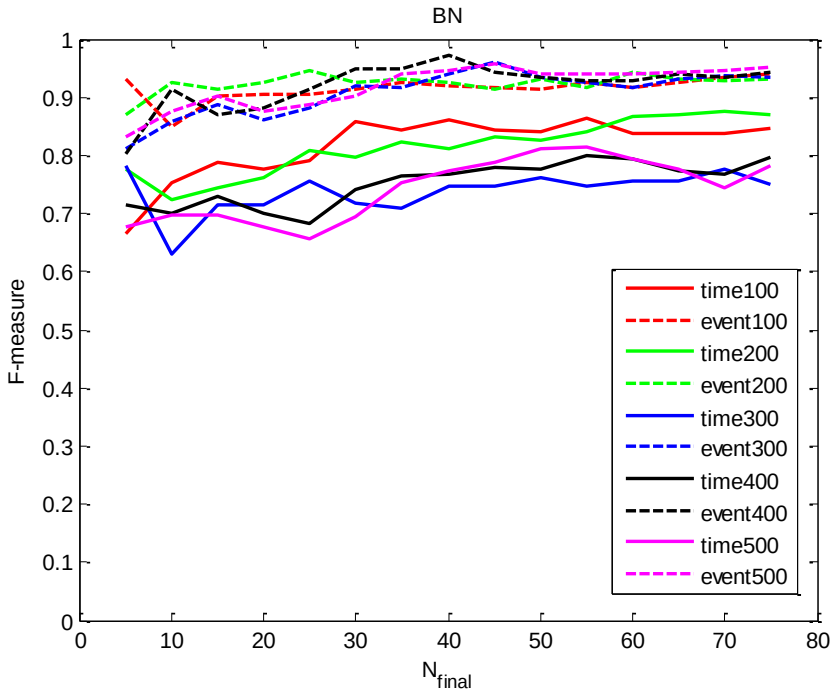


Figure 4-6. The average *F*-measure of the BN with different N_{init} and N_{final}

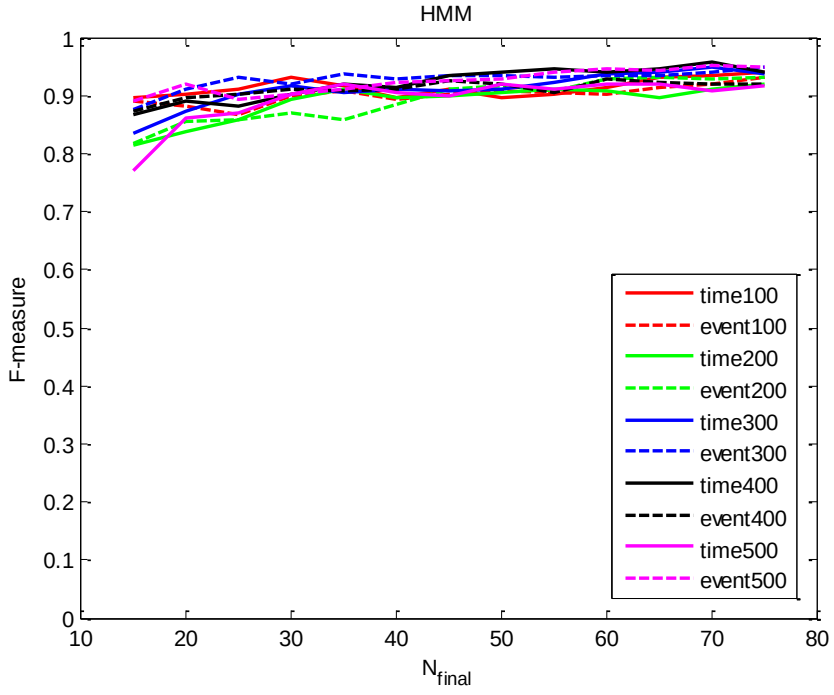


Figure 4-7. The average *F*-measure of the HMM with different N_{init} and N_{final}

Table 4-2. The average *F*-measure of J48 with statistical tests

	N_{final}=20		N_{final}=40		N_{final}=60	
N_{init}	time	event	time	event	time	event
100	0.695	**0.837***	0.764^{*+}	**0.804**	0.734	**0.793**
200	0.685	**0.824**	0.672	**0.803**$^{+}$	0.720^{+}	**0.770**
300	0.666	**0.729**	0.668	**0.829**$^{*+}$	0.737^{+}	**0.796**
400	0.652*	**0.740**	0.595	**0.747**	0.700^{+}	**0.795**$^{+}$
500	0.677	**0.736**	0.666	**0.749**	0.656	**0.799**$^{+}$

Table 4-3. The average *F*-measure of Bayes Net with statistical tests

	N_{final}=20		N_{final}=40		N_{final}=60	
N_{init}	time	event	time	event	time	event
100	0.777	**0.906**	0.860^{*+}	**0.920***	0.838	**0.916**
200	0.763	**0.925**	0.812*	**0.926**	0.866^{+}	**0.944**$^{+}$
300	0.714	**0.861**	0.746*	**0.941***$^{+}$	0.757	**0.918**
400	0.699	**0.882**	0.769*	**0.971***$^{+}$	0.795^{+}	**0.929**
500	0.678	**0.877**	0.773*	**0.946***	0.793^{+}	**0.940**

Table 4-4. The average *F*-measure of HMM with statistical tests

	N_{final}=20		N_{final}=40		N_{final}=60	
N_{init}	time	event	time	event	time	event
100	**0.902**	0.882	0.895	0.894	0.914^{+}	0.903
200	0.839	0.854	0.897^{*}	0.885^{*}	0.907	$\mathbf{0.928}^{+}$
300	0.873	**0.912**	0.911^{*}	$\mathbf{0.928}^{*}$	0.937^{+}	0.935
400	0.889	0.895	0.915^{*}	0.910^{*}	0.940^{+}	0.927^{+}
500	0.860	**0.919**	0.905^{*}	**0.922**	0.918^{+}	$\mathbf{0.945}^{+}$

B. Qualitative analysis of the extracted features

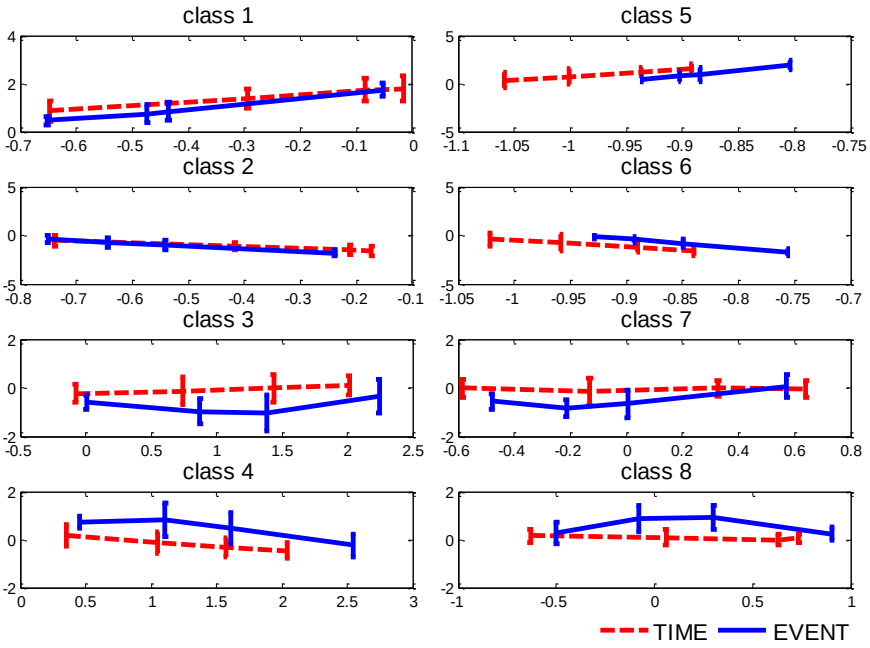
To have better insight to the features extracted from REL COORD feature extraction method, Figure 4-8 plots the feature values for each class after normalization. Recall that REL COORD extracts a sequence of feature vectors where each feature vector has *x*-coordinate ($\boldsymbol{o}_{kx}$), *y*-coordinate ($\boldsymbol{o}_{ky}$), and timestamp ($\boldsymbol{o}_{kt}$). In Figure 4-8(a), the horizontal and vertical axis represents the timestamp and *x*-coordinate respectively in the gesture space. Similarly, in Figure 4-8(b), the horizontal axis represents the timestamp in the gesture space, but the vertical axis represents the *y*-coordinate. The *k*th points in Figure 4-8 (a) and (b) represents ($\boldsymbol{o}_{kt}$, $\boldsymbol{o}_{kx}$) and ($\boldsymbol{o}_{kt}$, $\boldsymbol{o}_{ky}$). Each point is the average of the ten gestures of the same type, and the bar represents the standard deviation. In these figures, N_{init} and N_{final} are set to 100 and 5, and the class number is the gesture number described in Figure 3-1. Each solid and dotted line represents the event-based and the time-based method, respectively. When class 3, 4, 7, and 8 (the clock-wise and counter clock-wise

movements) are considered, the event-based segmentation can extract the patterns of gestures better than the time-based segmentation with smaller number of feature vectors, and no significant difference between the standard deviations of the two segmentation methods. This explains the superior performances of the event-based segmentation over the time-based segmentation as shown in the previous results (Figure 4-5, Figure 4-6, and Figure 4-7**)**, and supports my hypothesis 4.

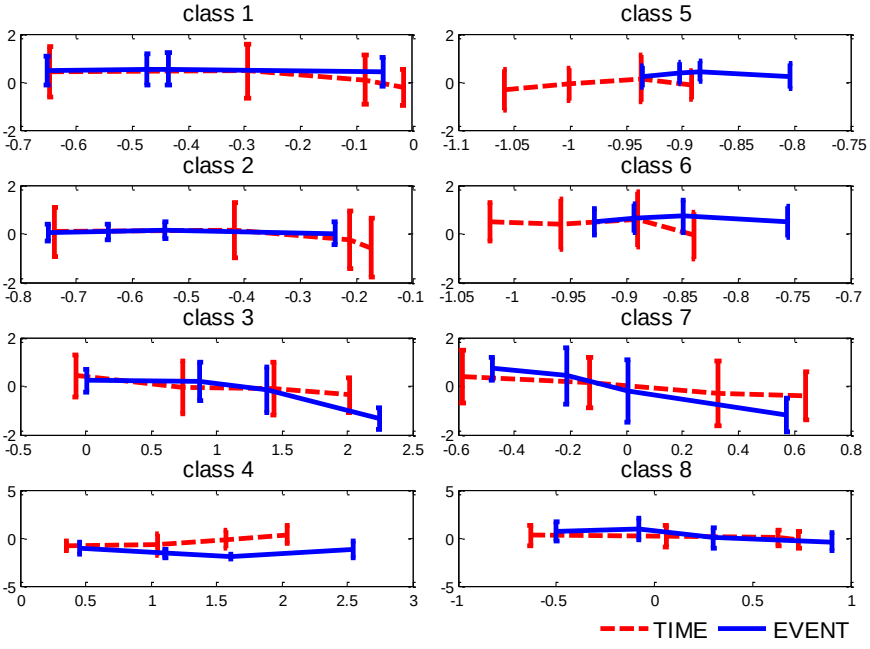
Figure 4-9 and Figure 4-10 show the normalized features when larger N_{final} (20 and 75 respectively) are used. For readability of the graphs, the bars representing the standard deviation are not shown. As expected, increasing N_{final} can describe the gesture trajectories at a fine-grained level, but more fluctuations are found. This supports my results shown in Figure 4-6 and Table 4-3 that a too large N_{final} degrades the performance of BN and thus supports the hypothesis 2.

When the two initial segmentation methods are compared, the time-based seems to be more prone to noise than the event-based segmentation. For example, when *y*-coordinates of the two initial segmentation methods are compared in class 1, 2, 5 and 6 in Figure 4-10(b), the time-based segmentation is very sensitive to noisy events especially those that occur at the end of gestures by segmenting the sparse points with the equal time interval. This supports our hypotheses 3 and 4 that the event-based is less sensitive to sparse noisy events and better extract the patters of gestures. It can be also said that the event-based segmentation is more robust to different settings of N_{final}.

To understand the impact of N_{init}, Figure 4-11 show the normalized data when N_{init} increases to 500 and N_{final} is set to 20. Compared to Figure 4-9 which uses a smaller number for N_{init}, the graph fluctuates more in Figure 4-11 both for the event-based and the time-based segmentations. This supports my first hypothesis that too large N_{init} makes the segmentation process more prone to noise. However, the amount of fluctuation is much bigger in the time-based than the event-based, which supports my hypothesis 3 and 4.

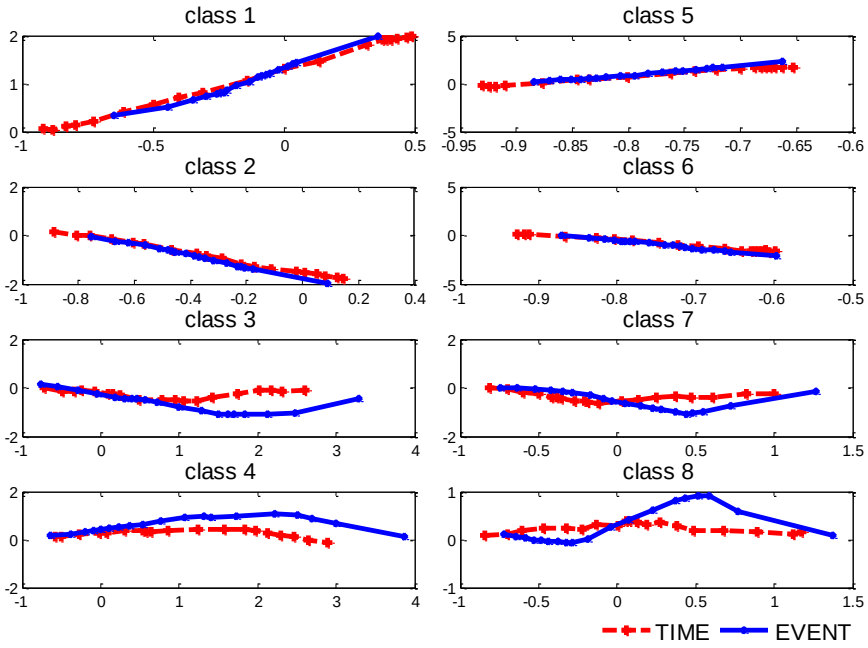


(a) x-coordinates and timestamps

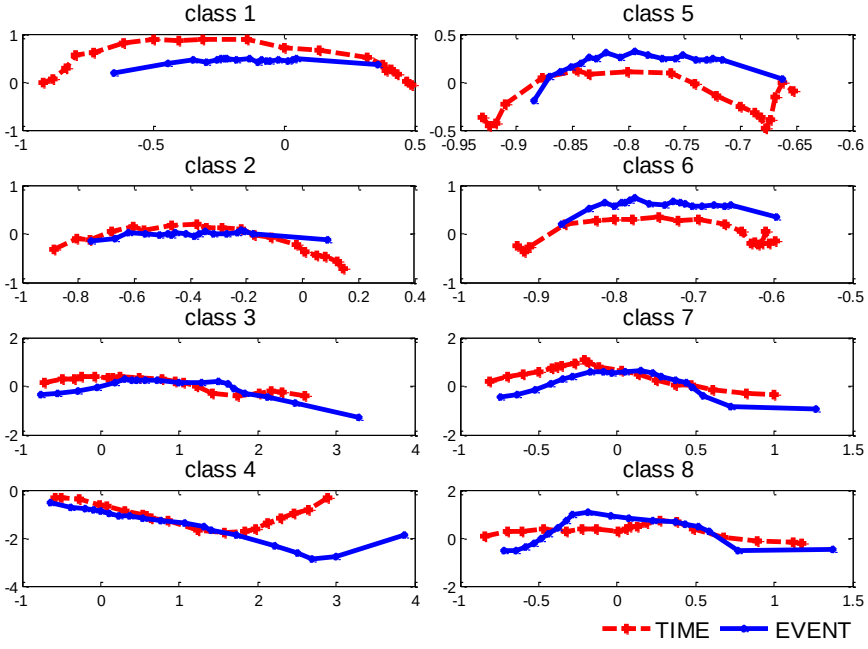


(b) y-coordinates and timestamps

Figure 4-8. The normalized features when N_{init} = 100 and N_{final} = 5

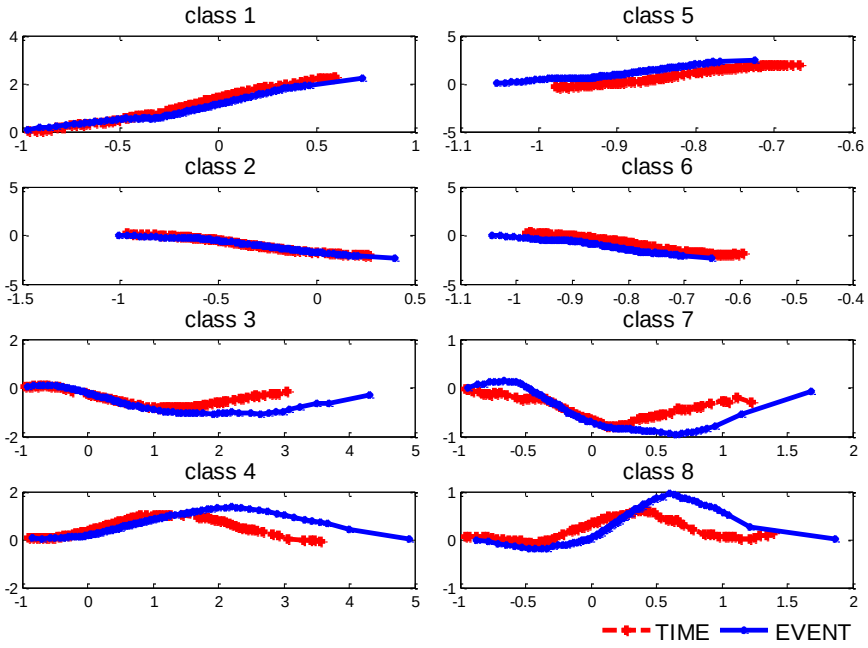


(a) x-coordinates and timestamps

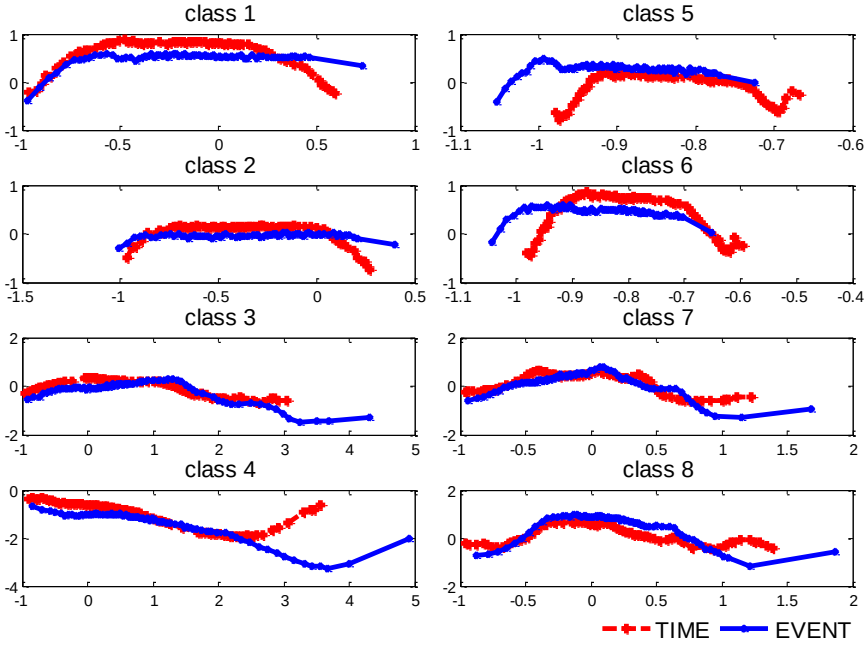


(b) y-coordinates and timestamps

Figure 4-9. The normalized features for the small number of final segments (N_{init} = 100, N_{final} = 20)

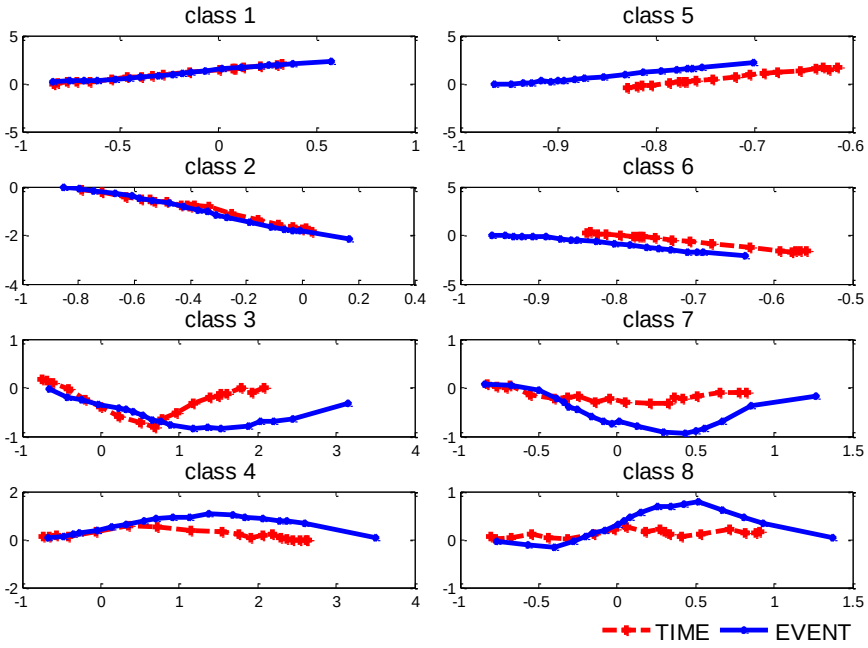


(a) x-coordinates and timestamps

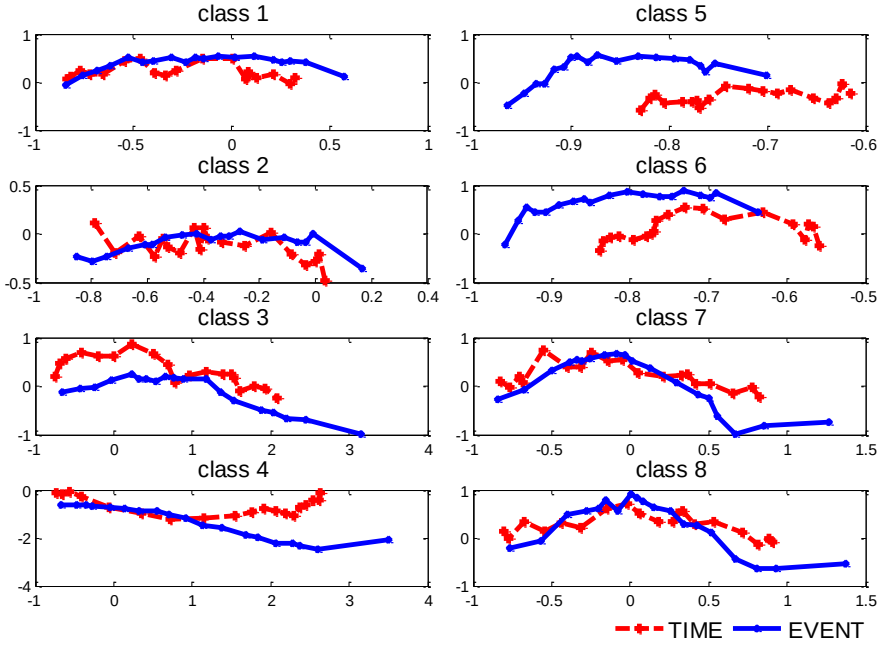


(b) y-coordinates and timestamps

Figure 4-10. The normalized features for the large number of final segments (N_{init} = 100, N_{final} = 75)



(a) x-coordinates and timestamps



(b) y-coordinates and timestamps

Figure 4-11. The normalized features for the large number of initial segments (N_{init} = 500, N_{final} = 20)

C. Comparisons of different feature extraction methods

To see the performance of different feature extraction methods, each Figure 4-12 and Figure 4-13 shows the performance of different classifiers for each DELTA COORD and VELOCITY feature extraction method. The *y*-axis represents the average *F*-measure of the 10 five-fold cross-validations, and the *x*-axis represents N_{final}. The number followed by the feature extraction method in legend represents N_{init}. In Figure 4-12, HMM performs better than other classifiers and the performance of BN degrade as the N_{init} increases even in the event-based segmentation. Considering my result that the impact of large N_{init} is relatively small in the event-based segmentation when REL COORD is used, DELTA COORD seems to be more sensitive to noise than REL COORD. The noise might be exaggerated in DELTA COORD by comparing two consecutive segments and focusing on the changes of the gesture trajectories too locally.

For example, Figure 4-14 shows the representative points for each segment ($\boldsymbol{z}_k$, $\boldsymbol{z}_{k+1}$, and $\boldsymbol{z}_{k+2}$), where $\boldsymbol{z}_{k+1}$ is the noisy point. The solid lines ($\boldsymbol{o}_{1r}$ and $\boldsymbol{o}_{2r}$) represent feature vectors extracted from REL COORD, while the dotted lines ($\boldsymbol{o}_{1d}$ and $\boldsymbol{o}_{2d}$) represent the feature vectors extracted from DELTA COORD. Compared to $\boldsymbol{o}_{1r}$, $\boldsymbol{o}_{1d}$ is more influenced by the noise. In addition, more number of feature vectors is influenced by the noisy point $\boldsymbol{z}_{k+1}$ in DELTA COORD; In DELTA COORD both $\boldsymbol{o}_{1d}$ and $\boldsymbol{o}_{2d}$ are influenced by $\boldsymbol{z}_{k+1}$, but in REL COORD only $\boldsymbol{o}_{1r}$ is influenced.

As shown in Figure 4-13, VELOCITY results in poor performance across different classifiers. Although VELOCITY can reduce the dimension of the features, dividing the difference of each *x*-coordinate and *y*-coordinate by the difference of the time may make the feature extraction doubly sensitive to noise.

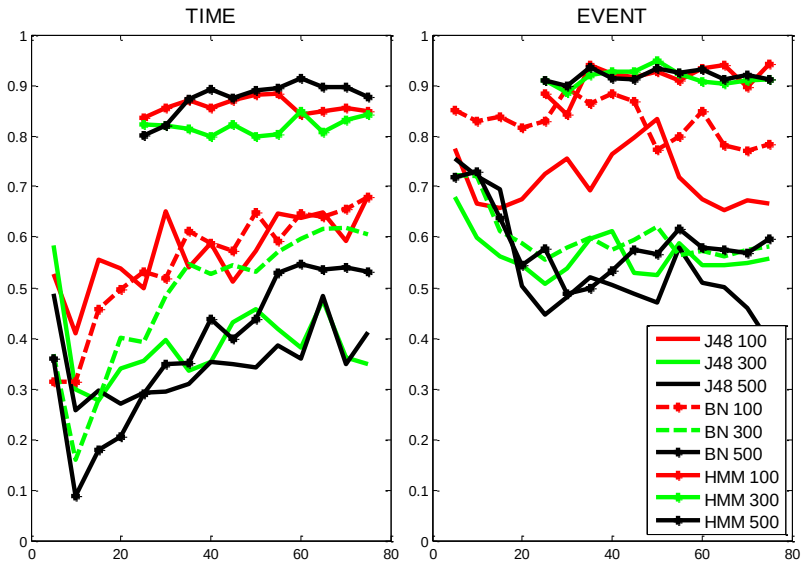


Figure 4-12. The average *F*-measure of DELTA COORD

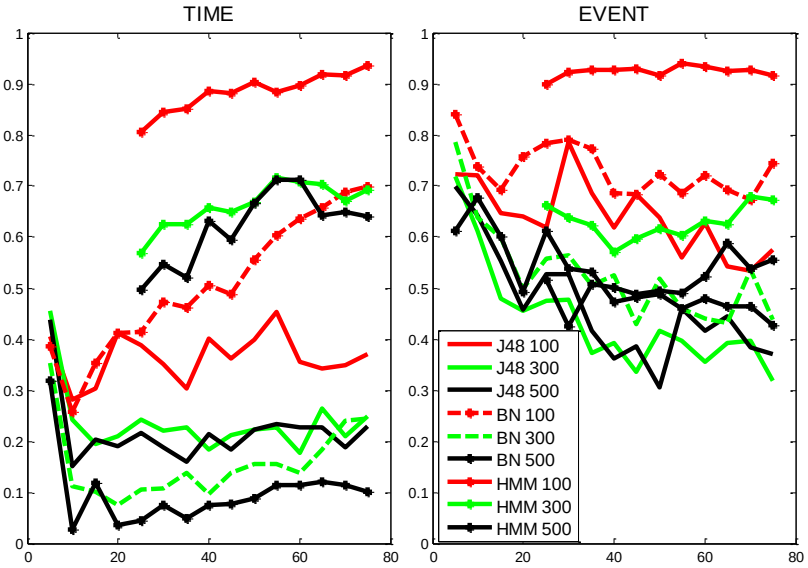


Figure 4-13. The average *F*-measure of VELOCITY

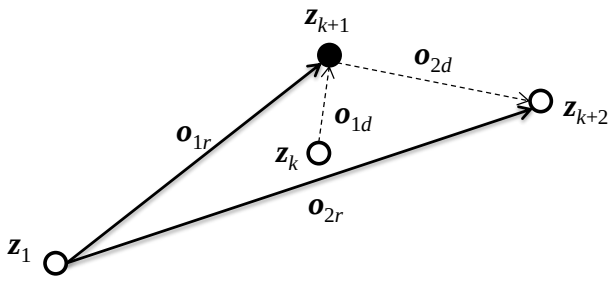


Figure 4-14. Comparisons of REL COORD and DELTA COORD

D. Performances of different classifiers

Lastly, one of J48s (in Figure 4-15 and Figure 4-16) and BNs (in Figure 4-17and Figure 4-18) for the time-based and the event-based initial segmentation are depicted when N_{init} and N_{final} are set to 100 and 5, respectively, and REL COORD is used for feature extraction. For readability, $\boldsymbol{o}_{kx}$, $\boldsymbol{o}_{ky}$ and $\boldsymbol{o}_{kt}$ are denoted as xk, yk and tk, where $1 \leq k \leq 4$. Although smaller numbers of features are used by the J48 built from the event-based segmentation, the event-based segmentation performs better than the time-based segmentation in J48 (Figure 4-5 and Table 4-2). It seems that the event-based segmentation helps the classifier to build general classifiers by representing the patterns of the gestures better than the time-based segmentation. In Figure 4-16, it is interesting to see that J48 first classifies data by the trajectory of the gesture, and then distinguish two gestures of the same trajectory by their speed using a time feature.

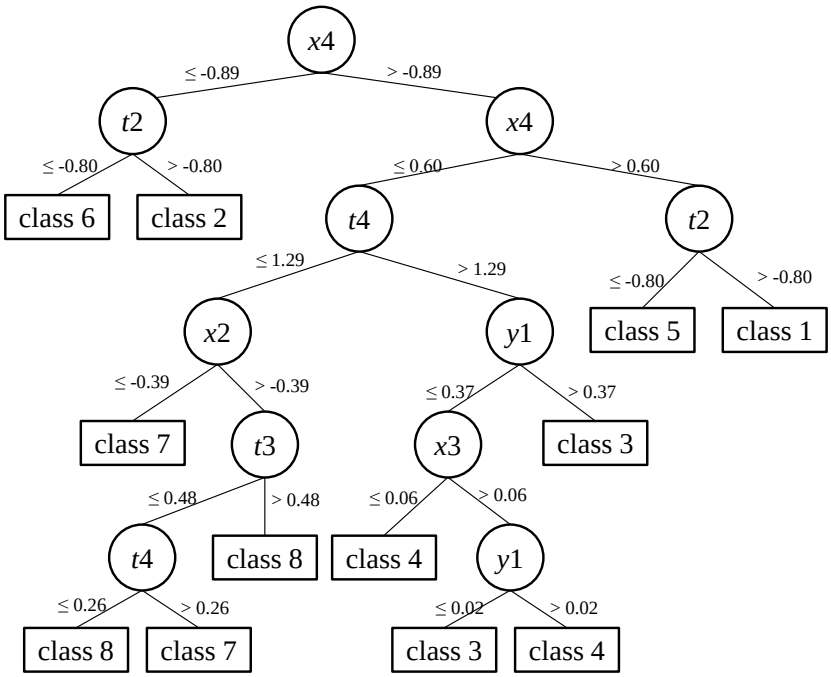


Figure 4-15. The J48 from the time-based segmentation

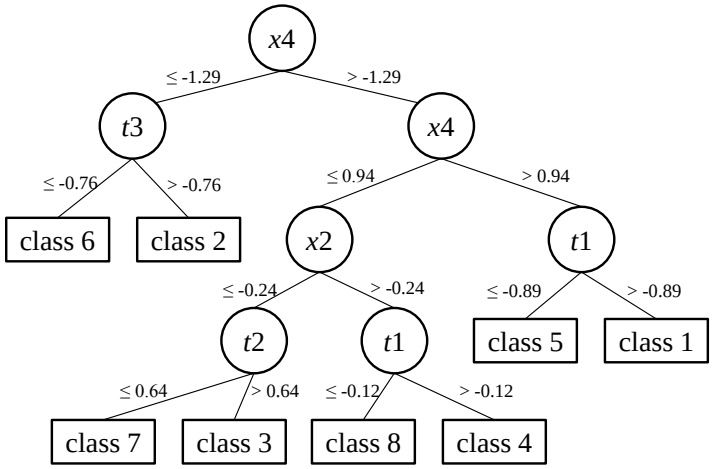


Figure 4-16. The J48 from the event-based segmentation

In Figure 4-17, BN built from the time-based segmentation hardly finds the dependencies between the coordinates. However, in Figure 4-18 the BN built from the event-based

segmentation can find dependencies between *x*-coordinates and *y*-coordinates as well as timestamps, and more dependencies are found between features that occur in similar time frames than for those that are far apart in time. The simple heuristic method for learning the structure of network, K2, seems to be able to find many temporal relationships between features. In contrast from the HMM, where temporal relationships are presented via hidden states, a BN can represent the dependencies between the features not only that occur in different time slots but also those that occur in the same time slot. This might also explain my result (Figure 4-6, Figure 4-7, Table 4-3, and Table 4-4) that BN can outperform HMM in some cases.

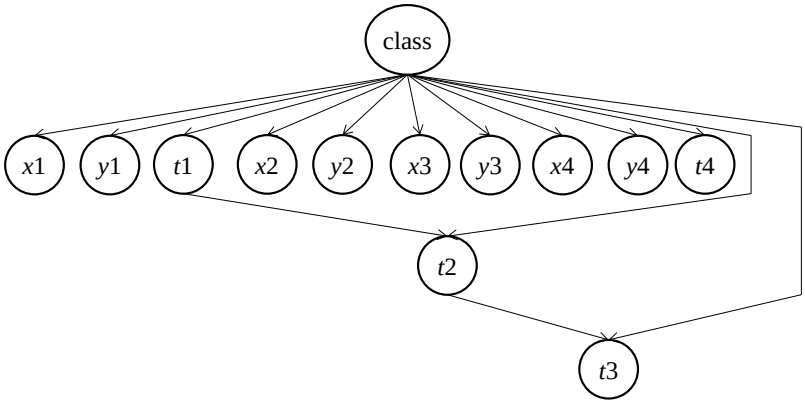


Figure 4-17. The BN from the time-based segmentation

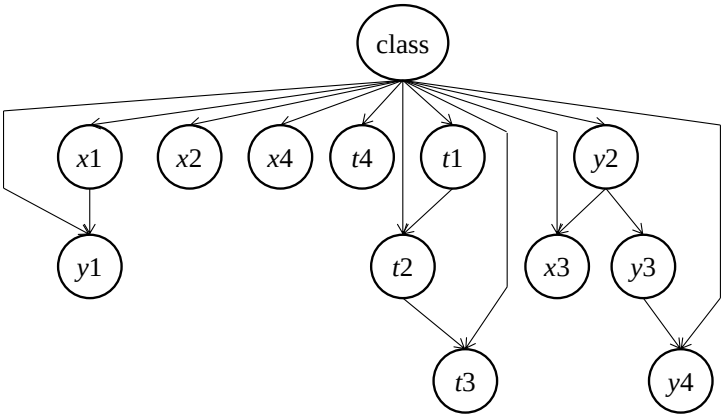


Figure 4-18. The BN from the event-based segmentation

To summarize, the performance accuracy of the classifiers and the analysis of the extracted data show that the event-based segmentation outperforms the time-based segmentation by better extracting important patterns of gestures and being more robust to noise. Table 4-5 summarizes how each hypothesis described in Table 4-1 is supported by my results. Among the different feature extraction methods, REL COORD performs the best and is more tolerable to noise than DELTA COORD and VELOCITY. The HMM seems to be most useful when features contain noise, since noise can be absorbed in emission probability of the hidden state, and the BN is useful when features are well extracted using the event-based segmentation and N_{final} is small.

Table 4-5. The summary of supporting results for the hypotheses

Hypothesis	Supporting Results
H1 (N_{init})	Comparisons of different lines in Figure 4-5, Figure 4-6, and Figure 4-8 Comparisons between Figure 4-9 and Figure 4-11
H2 (N_{final})	Comparisons of different values of x-axis in Figure 4-5, Figure 4-6, and Figure 4-8 The result of statistical tests shown with superscripts in Table 4-2, Table 4-3, and Table 4-4 Comparisons of Figure 4-8, Figure 4-9, and Figure 4-10
H3 and H4 (event vs time)	Comparisons between the dotted lines and the solid lines in Figure 4-5, Figure 4-6, and Figure 4-8 The result of statistical tests shown with bold face in Table 4-2, Table 4-3, and Table 4-4 Comparisons between the dotted lines and the solid lines in Figure 4-8, Figure 4-9, Figure 4-10, and Figure 4-11

7. Summary

This Chapter proposes a novel gesture-inspired segmentation method for feature extraction. Since a single gesture can have a large number of events detected by the DVS camera at a microsecond granularity level, I first segment a sequence of events to get a smaller number of representative points, from which features are extracted. The proposed segmentation method first divides a sequence of events into small segments to have the same number of events (event-based) or the same length of time interval (time-based), and then repeatedly aggregates neighboring segments with similar event distributions. The distance between two neighboring segments is measured by Jefferey's information. After a user-specified number of final segments is obtained, the mean vector of the events for each segment is used as a representative point of each segment.

Features are extracted from the representative points using three methods: REL COORD, DELTA COORD, and VELOCITY. REL COORD translates a sequence of the representative points in a gesture to the first representative point in the sequence, and uses the translated representative points except the first point as the feature vectors. In DELTA COORD, feature vectors represent the difference between the representative points of the consecutive segments, and VELOCITY divides the *x*-coordinate and *y*-coordinates obtained from DELTA COORD by the corresponding timestamp.

The experimental results using decision tree (J48), Bayesian network (BN), and hidden Markov model (HMM) show that the event-based initial segmentation outperforms the time-based segmentation across different classifiers and is more robust to noise and various settings for the number of the initial segments. The HMM shows robust performance most of the time, but the BN can perform better than the HMM when features are well extracted using the event-based segmentation. The HMM seems to be more tolerant to noise than the BN and J48 classifiers by

absorbing noise via hidden states. On the other hand, the BN seems to perform better when there are a small number of feature vectors.

Chapter 5

Global Feature Extraction Using an Evolutionary Algorithm

The REL COORD feature extraction method discussed in Chapter 4 can be considered as a local feature extraction, since the feature vector is extracted from a segment composed from a sequence of events that occur in a narrowly defined time and space. The DELTA COORD and SPEED feature extraction methods attempt to find higher level features by considering the relationship between representative features in the adjacent segments, but they fail to provide good results due to their sensitivity to noise. In addition, they are still relatively local feature extraction methods since only the interactions between representative points of neighboring segments are considered. This Chapter aims to construct higher level compound features by transforming simple features extracted from REL COORD, and to find a good set of simple and compound features, which can enhance the performance accuracy of a classifier and explain the underlying relationships between the features. Note that this global feature extraction approach has two implicit sub-problems: constructing higher level compound features, and finding the proper number of compound features. The examples of such compound features are the difference in x-coordinates of the starting and the ending points or the average speed of the horizontal movement during the gesture. By using compound features, I expect (1) to improve performance accuracy of classifier by reducing noisy data; (2) to reduce the number of features, and (3) to provide insight to the underlying relationships between the features in the data.

A compound feature can be constructed by applying a sequence of mathematical transformations to simple features [43]. As discussed in Chapter 2, the feature construction and selection process has often employed evolutionary algorithms (EAs) as a search algorithm and has shown promising results [41-46, 56]. This research also employs an EA, but the

representation of an individual, the genetic operators, and the fitness evaluation method are specified to the gesture classification problem. The major characteristics of my gesture data are that it is a time-series data with variable time interval and the number of instances for each class is very limited.

Good EA design for feature extractions requires: (1) good design of genetic operators; (2) accurate estimation of the fitness of an individual, i.e., a feature set; (3) a mechanism for reducing overfitting; and (4) an efficient search method for the large search space. Genetic operators including mutation and crossover should be well defined so that promising candidate solutions can be generated. Operators must provide enough power for exploration while not being too destructive. Since the actual fitness landscape is unknown, the fitness, i.e., accuracy, of an individual needs to be estimated. Although the performance of a feature set for unseen data can be estimated more accurately when there is enough data, it is non-trivial given a very limited amount of data for each class. In addition, the fitness evaluation should be able to well discriminate among different but similar solutions, and be computational tractable. Since we cannot perfectly estimate the performance of a feature set for unseen data, the best-so-far solution found using the training data may not perform well for unseen data, which is known as overfitting. A mechanism to reduce overfitting also needs to be considered.

Computational feasibility requires considering the search space, where the number of possible subsets of simple features is 2^n, if there are n simple features. Since compound features are also considered in this research, the number of subsets of features is 2^{n+m}, where m is the number of compound features. If a compound feature is constructed by applying one mathematical operator, such as subtraction, to two different simple features as operands, then the number of compound features, m, is n choose 2, which is the order of n^2. However, since there are multiple transformation operators and these operators can be applied to compound features that

have been already generated, the search space will be even larger than 2^{n+n^2}. Thus, efficiently searching for a good feature set in this large search space is a key concern in my research.

The next Section describes an evolutionary algorithm for feature extraction specified to my gesture classification problem. Representation, fitness evaluation, mutation, and crossover operator are first defined and next the overall procedure of the proposed EA for feature construction and selection is described. Section 2 proposes a Bayesian network-based mutation scheme, which aims to make the search process more efficient with a guidance of relational information about the features presented in a Bayesian network. Section 3 shows the experimental results including a quantitative analysis on classification accuracy and complexity, and compares the performance of different mutation schemes. Section 4 analyzes the semantic meanings of the extracted compound features, and finally Section 5 concludes this Chapter with a summary.

1. An Evolutionary-based Global Feature Construction and Selection

This section describes an EA for feature construction and selection (EAFCS). For clarification, some notations are first defined. A set of simple features is represented as $\boldsymbol{A} = \{a_1, a_2, \dots, a_k\} = \{o_{1x}, o_{1y}, o_{1t}, \dots, o_{(N_{\text{final}}-1)x}, o_{(N_{\text{final}}-1)y}, o_{(N_{\text{final}}-1)t}\} = \{x_1, y_1, t_1, \dots, x_{N_{\text{final}}-1}, y_{N_{\text{final}}-1}, t_{N_{\text{final}}-1}\}$ where a_i is the ith simple feature which is locally extracted by using the REL COORD feature extraction method; o_{1x}, .., $o_{(N_{\text{final}}-1)t}$ are the notations used in Equation 4-7; and x_i, y_i, t_i correspond to o_{ix}, o_{iy} and o_{it}, respectively.

A. Representation and fitness evaluation

An individual I represents a set of features, i.e., $\boldsymbol{F} = \{f_1, f_2, \dots, f_l\}$, where f_i is either a simple feature or a compound feature. A variable length string is used to represent each individual

as shown in Figure 5-1. In the example, there are three features, i.e., three *f*s, in the chromosome where f_1 consists of a_1; f_2 consists of a division operator and two simple features a_4 and a_1; and f_3 consists of a subtraction operator and two simple features a_5 and a_2 as operands.

a_1	a_4 / a_1	$a_5 - a_2$

Figure 5-1. The representation of an individual

Although one of my goals is to reduce the number of features, reducing the number of features should not degrade the predictive performance of a classifier. Thus, the fitness of an individual is evaluated by estimating the predictive performance of a classifier when the individual's feature set is used for learning, and the simpler individual is preferred only when the two individuals have the same fitness. If there is enough data, the simplest approach to measure the predictive performance might be using the average *F*-measure over different classes.

Specifically, the predictive performance of an individual *I* can be estimated as follows: (1) transform the data *D* according to the feature transformation expressions defined in *I* to produce a transformed data *D*'. (2) *D*' is divided into training data, D'_T, and validation data, D'_V. Note that examples in *D* and *D*' are all labeled. (3) A classifier is built using D'_T, and then the classifier is used to predict the class label of D'_V assuming that the class labels of data in D'_V are hidden. By comparing the predicted class labels with the actual class labels attached to data in D'_V, the average *F*-measure across different classifier can be obtained.

To estimate the accuracy of a classifier more accurately, and to make the EA less prone to overfit to a particular D'_T and D'_V, a *k*-fold cross-validation is used. According to the *k*-fold cross-validation, the data set *D*' is divided into *k* folds, i.e., $D'_1, \ldots, D'_k$. For each *i*th fold, the

model is built using $D' - D'_i$, and tested with D'_I [73]. Although the common values for k are 2, 5 or 10, k = 4 is used in my experiment considering the number of data available for the fitness evaluation. The fitness can be obtained by averaging the F-measures of classifiers over k folds. This process is described in Figure 5-2.

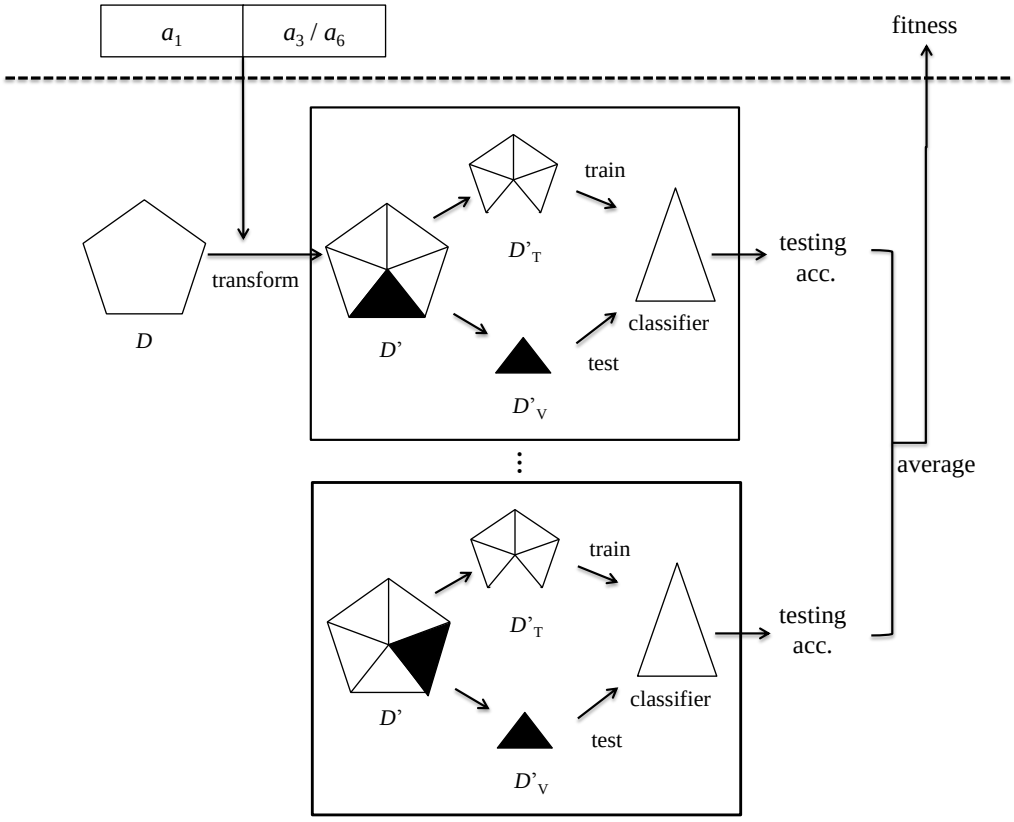


Figure 5-2. The fitness evaluation

However, considering the fact that the number of training data for an EA is very limited, I doubt that the performance of an individual can be accurately estimated with a single cross-validation, and also doubt that the two individuals with similar performance can be discriminated well due to the limited number of possible values for the fitness. Thus, the average F-measure obtained from a cross-validation may not be a good evaluation method to well guide the EA

toward the better solutions, and more fine grained fitness evaluation method is needed. One possibility to tackle this problem may be to use a *k*-fold cross-validation multiple times with different splits of data, but repeating cross-validation multiple times makes the fitness evaluation computationally too expensive.

Instead, this paper uses Area Under Curve (AUC) metric specified for multi-class problems [74]. The AUC represents the area under the Receiver Operating Characteristics (ROC) curve, which shows the tradeoffs between the true positive rate, depicted on the *y*-axis, and the false positive rates, depicted on the *x*-axis, of a classifier across different settings of the classifier's parameters. In binary classification problem, AUC can be calculated as in Equation 5-1 where c_0 and c_1 represents positive and negative class, n_0 and n_1 represent the numbers of positive and negative instances, and r_i represents the rank of the *i*th positive instance when instances from both classes are sorted in ascending order based on the estimated probability of belonging to the positive class [75]. S_0 is the sum of ranks for the positive instances. AUC can be interpreted as the probability that the estimated probability of a randomly selected negative example belong to a positive is lower than the estimated probability of a randomly selected positive example belong to a positive.

$$auc(c_0, c_1) = \frac{S_0 - \frac{n_0(n_0+1)}{2}}{n_0 n_1}$$

$$S_0 = \sum r_i$$

Equation 5-1

The AUC was designed to measure the performance of classifiers for binary classification problems, but Hand and Till [75] extended the AUC metric to multi-class problems. The AUC for a multi-class problem is measured by the average AUC of all pairs of classes as shown in

Equation 5-2, where c is the number of classes and $auc(c_i, c_j)$ is calculated as in Equation 5-1. Huang and Ling [74] showed that the AUC is consistent with accuracy but can discriminate similar classifiers better than simple accuracy measure.

$$AUC = \frac{2}{n_c(n_c - 1)} \sum_{i<j} AUC'(c_i, c_j)$$
$$AUC'(c_i . c_j) = \frac{auc(c_i, c_j) + auc(c_j, c_i)}{2}$$

Equation 5-2

Since my fitness evaluation method incorporates the actual learning algorithm during evaluation, it is a wrapper approach. Any learning algorithm can be used, but it should be selected considering the computational cost for learning and testing. To use AUC for the fitness evaluation, a classifier needs to produce probability estimations for each class. In my experiment, Naïve Bayes classifier is used. When the individuals are compared, the higher fitness is the better. If the fitness of the two individuals is the same, the less complex individual is considered to be better based on Occam's Razor [64].

Employing the idea of Lam and Bacchus [76], the complexity of an individual is measured by the ratio of the minimum description length (MDL) required to encode an individual's feature set to the MDL required to encode the simple features as shown in Equation 5-3, where K and N_{opr} represent the numbers of simple features and pre-defined operators, and n_{oprnd} and n_{opr} represent the total numbers of simple features and operators in the individual's feature set. To encode a simple feature and a transformation operator, $\log_2 K$ and $\log_2 N_{opr}$ bits are required. For example, the MDL of the second compound feature a_4/a_1 in Figure 5-1 is $2\log_2 K + \log_2 N_{opr}$. The MDL of an individual is the sum of the MDLs of all features in the individual.

$$complexity(I) = \frac{MDL\ for\ I}{MDL\ for\ simple\ features} = \frac{n_{oprnd} log_2 K + n_{opr} log_2 N_{opr}}{K log_2 K}$$

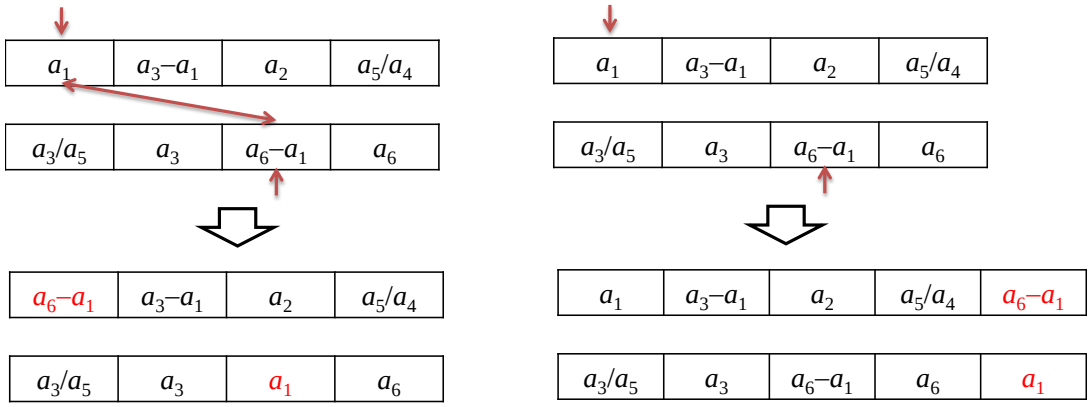
Equation 5-3

The general approach to measure the complexity of an individual is to count the number of mathematical operators and simple features in the individual [26]. My approach differs from the general approach in that the amount of contribution that a simple feature and an operator made is different. In my EA, the number of operators is relatively smaller than the number of simple features. Thus, adding two simple features increases complexity more than adding one simple feature and one operator. For example, if there are an individual with three simple features, a_1, a_2 and a_3, and an individual with a_1/a_2, their complexities are the same based on the general complexity measure, but my approach favors the latter individual. This approach will give more penalties to the individuals with more number of *f*s when the sums of the number of simple features and the number of operators in the two individual's feature sets are the same. In my problem where the number of operators is smaller than the number of simple features, this seems to be reasonable since the larger number of compound features (*f*s) may make the learning and classification tasks computationally more expensive.

B. Crossover

The crossover method randomly selects a feature from each of the two parents, and either exchanges the selected features as shown in Figure 5-3(a) or adds the selected feature from the other parent as shown in Figure 5-3(b). The first approach is called crossover-by-exchange, while the latter approach is called crossover-by-addition. Whenever crossover is performed, one crossover scheme is randomly selected. If the crossover operator only performs crossover-by-exchange, it becomes hard to construct a larger set of features from the simpler feature set. On the

other hand, if crossover-by-addition is only performed, the number of compound features in the individual will get too large. The alternative approach for crossover might be selecting a crossover point for each parent and exchanging a set of features as shown in Figure 5-4, but this method destructs the parents' feature sets too much and seems to be inadequate to produce promising offspring.



(a) crossover-by-exchange (b) crossover-by-addition

Figure 5-3. Crossover

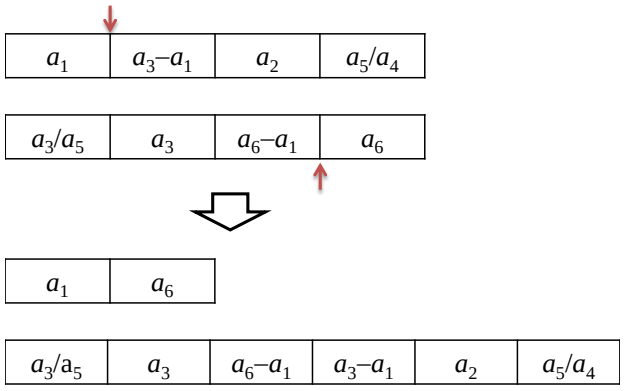


Figure 5-4. The alternative approach for crossover

The two offspring created from crossover are included in the next generation. Although the EA can select one superior offspring for the next generation, this method empirically shows the loss of important compound features. Since an individual is evaluated based on the performance of the individual's feature set not on the performance of a single feature, important compound features that can potentially perform well with other features can be lost simply based on the poor individual evaluation results.

C. Mutation

Mutation is performed in two steps as shown in Figure 5-5. First, a mutation operator is randomly selected from a set of predefined mutation operators shown in Table 5-1. Next, the appropriate operands are selected. When the inclusion operator is selected, a simple feature that is not included in the current individual is randomly selected, and added to the individual. When the exclusion operator is selected, a random compound feature *f* in the individual is deleted. The replacement operator selects a random feature from an individual and replaces it with the same type of simple feature, which does not exist in the current individual. For example, if the selected feature is *x*-coordinate, then the *x*-coordinate feature is selected for the replacement. The detail description on feature types will be discussed shortly.

Using two-operand operators, such as plus, minus, division, min and max, means that the operator will randomly select the first feature f_1 from the current individual and the second feature f_2 from a set of simple features and generate a new compound feature $f_1 + f_2$, $f_1 - f_2$, f_1 / f_2 , min(f_1, f_2) and max(f_1, f_2). The new feature replaces f_1, because simply adding the new compound feature may increase the redundancy between the features in the individual. A slightly different method for selecting the operands is selecting both operands from the current individual, and replacing the two operands by the new feature. The first method that selects the second operand from a set

of simple features is called EAFCS-EX while the latter method is called EAFCS-IN. I developed EAFCS-IN based on the assumption that it has the potential to construct higher level features more quickly than EAFCS-EX by combining two compound features. However, EAFCS-IN can be less explorative than EAFCS-EX, since new features will be less frequently introduced to the population in EAFCS-IN. In addition, replacing two features by one feature can reduce the diversity of features in the population.

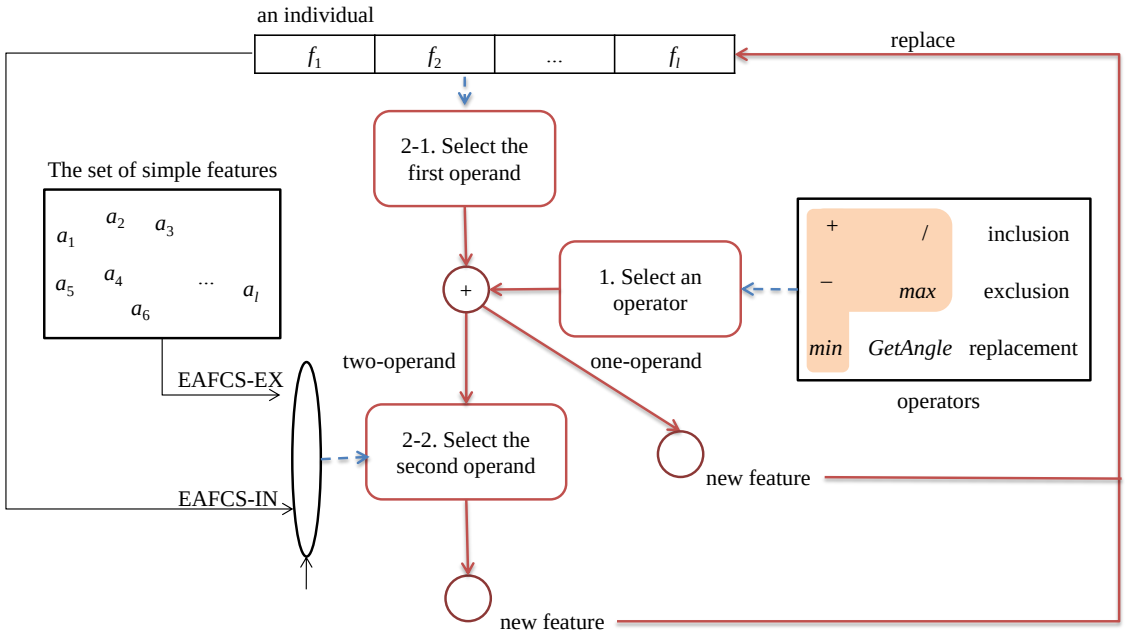
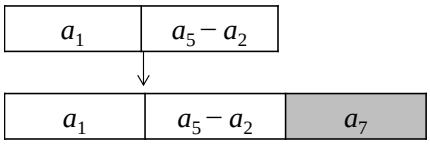
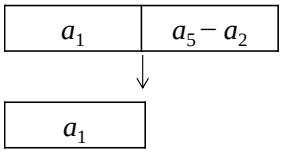
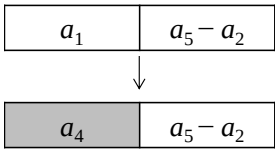
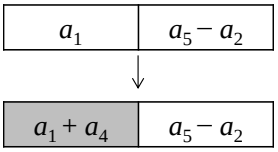
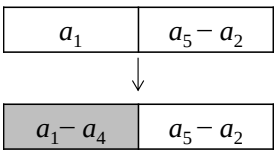


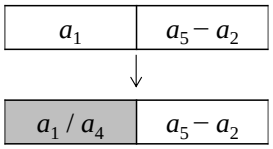
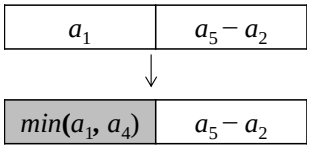
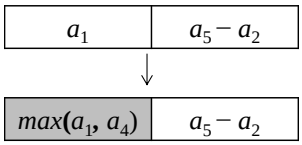
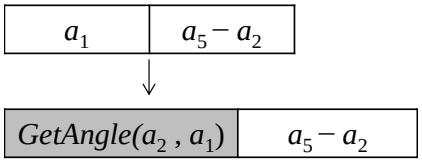
Figure 5-5. Mutation

Lastly, the *GetAngle* mutation operator selects a simple *x*- or *y*-coordinate feature from the current individual and replaces it with the angle at that point. The angle is scaled in 0-360 degrees, as shown in Figure 5-6. The *GetAngle* operator can be performed by adding $\tan^{-1}$ as a mutation operator, but this requires that the division operator must first be applied to *x*-coordinates and *y*-coordinates that occur at the same time. In addition, the compound feature x / y must be good enough to be selected as an operand of the $\tan^{-1}$ operator in the future generation.

To avoid these problems, this paper adds the *GetAngle* function as a mutation operator, as opposed to simply adding the $\tan^{-1}$ operator.

Table 5-1. The mutation operators

OPERATOR	**EXAMPLES**
inclusion	
exclusion	
replacement	
plus	
minus	

division	
min	
max	
GetAngle	

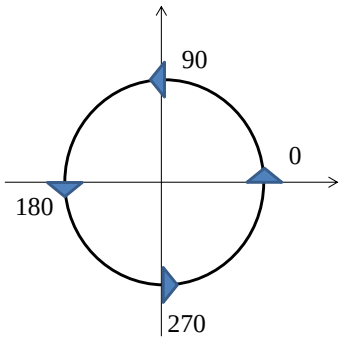


Figure 5-6. *GetAngle* measurements in degrees

To generate contextually sound compound features some restrictions are added to the operand selection process. For example, a compound feature generated by subtracting a time

feature from an *x*-coordinate is not contextually meaningful. As a step towards defining restrictions, I first define eight feature types: *x*, *y*, *t*, *x*, *y*', *t*', θ and *other*. The *x*, *y* and *t* are the *x*-coordinate, *y*-coordinate, and timestamp of the simple features, and the *x*', *y*', and *t*' are compound features but ones that remain in the *x*-coordinate, *y*-coordinate and time domain. For example, a compound feature generated by subtracting x_5 from x_1 is considered as *x*'. The angle θ represents the angle generated from the *GetAngle* operator. Feature types that are not of these seven canonical types are classified as *other*. The following restrictions are added: the plus and minus operators are only applied to features of the same domain (i.e., between features in the *x*-domain, *y*-domain, timestamps, and angles); the division operator can be applied between any types of features, but *x*, *x*', *y*, and *y*' cannot be selected as the second operand for the first operand θ (i.e., dividing the angle by a location coordinate is not generally meaningful in this domain). The min/max operator can be applied to features from the same domain but cannot be applied to *t* or *t*' since $t_i < t_j$ for all $i < j$, and min(t_i, t_j) is the same as replacing t_i by t_j. The restrictions are summarized in Table 5-2. Restricting operands also reduces the search space, and thereby makes the search process more efficient.

Table 5-2. Allowable feature types for each operator

OPERATOR	**OPERAND 1 TYPE**	**OPERAND 2 TYPE**	**OUTPUT TYPE**
inclusion	*x*, *y*, *t*	N/A	
exclusion	any		

replacement	x, x' y, y' t, t'	x, x' y, y' t, t'	x, x' y, y' t, t'
plus/minus	x, x' y, y' t, t' θ	x, x' y, y' t, t' θ	x' y' t' θ
division	x, y, t, x', y', t' θ	x, y, t, x', y', t' t, t'	other
min/max	x, x' y, y' θ	x, x' y, y' θ	x' y' θ
GetAngle	x, y	N/A	θ

D. The overall process for EAFCS and selection for the final solution

The EA for feature construction and selection is summarized as follows: (1) Each individual in an initial population P_0 is generated by randomly selecting a simple feature. Thus, all the initial individuals have one feature. (2) Fitness of the individuals in P_0 is evaluated. (3) To generate the next population P_{i+1}, a mating pool is created through binary tournament selection from P_i, where the size of the mating pool is the same as the size of population N_{pop}. (4) The population for the next generation P_{i+1} is generated by repeating the following two steps: randomly selecting an individual from the mating pool; and performing crossover with the probability P_{co}. If the individual is selected to be applied the crossover operation, another parent is randomly selected from the mating pool, and the two offspring generated from the crossover is

added to the next generation. Otherwise, the individual is added to the next generation without any change. Crossover is performed by randomly selecting one of the following two crossover schemes: crossover-by-exchange or crossover-by-addition. (5) Mutation is performed for each individual with the mutation rate P_{mut}. Due to the specified mutation method, the mutation rate is applied at the level of individual not at the level of gene. Using elitism, the best individual is kept in the next generation. The overall process of the EAFCS is depicted in Figure 5-7.

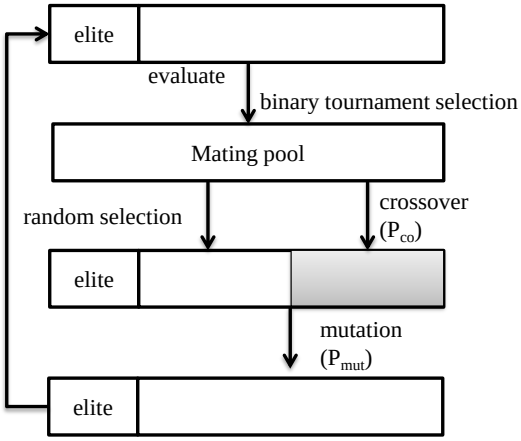


Figure 5-7. The overall process of the EA for feature extraction

Another approach for initialization is to select a subset of simple features as an individual's feature set. However, including too many features not proven to be good features can keep the EA from the efficient search, since the EA will try to construct higher level features on top of the unproven features. In addition, this can also makes the search more difficult by providing too many possible ways of generating offspring. For example, the number of ways to select the first operand increases as an individual includes more features. Thus, my EA initializes each individual with a simple feature, and gradually constructs and adds compound or simple features to individuals.

As discussed in Chapter 2, the individual with the highest fitness is not guaranteed to perform well for the unseen data due to overfitting. One way to reduce the overfitting is to select the final solution from the best-so-far solutions based on the performance on a separate validation set, which is not involved during the fitness evaluation. However, given the limited amount of data for the EA, separating a validation set from the fitness evaluation set can impede the accurate estimation of an individual's fitness. Instead, in this research, the final solution is selected based on using 4-fold cross-validation with the same data as for the fitness evaluation, but using a different random seed for data partitioning. In this way, overfitting to the specific split of the data (i.e., D'_T and D'_V) can be partially avoided. One drawback of this approach is the computational cost arising from the additional cross-validations. However, compared to the total number of evaluations over a course of an EA run, the additional cost for selecting the final solution is very small. For example, if an EA with the population size 100 is run for 100 generations, the total number of fitness evaluations is 10,000, while the number of evaluations for selecting the final solution will be less than 100, since the best-so-far solution will not be updated at every generation.

2. Bayesian Network-guided Evolutionary Search for Compound Feature

I expect that information about the feature relationships might be useful in constructing promising compound features and making the search process more efficient. Thus, in addition to two mutation schemes: EAFCS-EX and EAFCS-IN, this Section proposes a mutation scheme that considers feature relationships in selecting the second operand. Since the structure of Bayesian network (BN) can represent the conditional independence relationship between the features, I use the BN built from a set of original simple features to select the second operand. This approach is called EAFCS-BN.

I assume that if there is a functional relationship between the features a_i and a_j, then a_i will be in the Markov blanket of a_j, and vice versa. Tsamardinos and Aliferis [77] show that the Markov blanket (MB) consists of strongly relevant features in any faithful[1] Bayesian network, and recently Markov blankets have been used in feature selection to consider causality between features [78-80]. In a faithful Bayesian network, the Markov blanket of node a_i includes the parent, children, and children's parent (spouse) nodes of a_i. Based on my assumption and on previous work, EAFCS-BN selects the first operand from the current individual and the second operand from the MB of the first operand f with a high probability. For example, in Figure 5-8, if a_3 is selected from the current individual as the first operand, then the second operand is randomly selected from the features in the MB of a_3, i.e., a_1, a_5, and a_7.

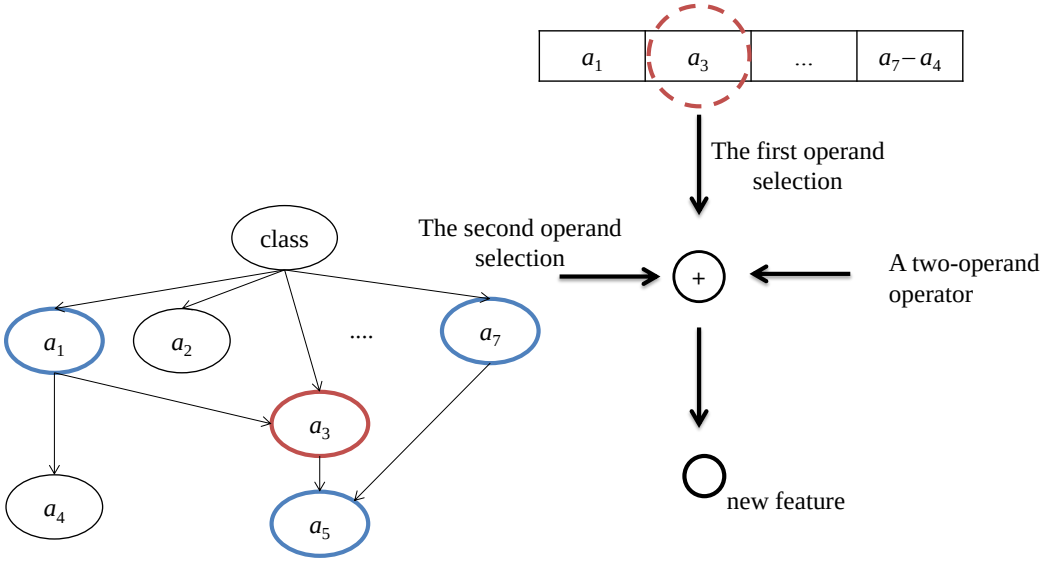


Figure 5-8. A simple example of EAFCS-BN

[1] According to Tsamardios et al, a Bayesian network is considered to be a faithful network, if the distribution where the data is sampled from embodies only independencies represented in the directed acyclic graph, i.e, the structure of the network.

If the selected first operand f_1 is a simple feature (i.e., no operator has been applied), a set of simple features $\boldsymbol{S}$ from which the second operand is selected is simply the features in the MB of f_1 except the class node. However, if f_1 is already a transformed feature such as $a_7 - a_4$, determining $\boldsymbol{S}$ becomes non-trivial. One way to solve this problem might be to rebuild a Bayesian network using both the compound features in the individual and the entire simple features in the original data, but learning a BN whenever an individual is mutated will make the EA computationally too expensive. Instead, the simple features in MBs of all simple features in f_1 are included in $\boldsymbol{S}$. If f_1 is $a_7 - a_4$ in Figure 5-8, the features in the MBs of each a_7 and a_4, i.e., a_1, a_3, and a_5, will be included in $\boldsymbol{S}$.

To probabilistically select features in $\boldsymbol{S}$, I define P_{MB} as the probability that a feature is selected from $\boldsymbol{S}$. With $(1 - P_{\mathrm{MB}})$, the second feature is selected based on EAFCS-EX. If there is no feature in $\boldsymbol{S}$ that satisfies the restrictions shown in Table 5-1, EAFCS-EX is used to select the second operand. By using heuristics to select relevant features, I expect that EAFCS-BN can generate more promising offspring than EAFCS-EX.

3. Factors Affecting the Individual's Search Space

To provide better insight to my algorithm, I will clarify the specific components that control the explorative power of the EA. Recall that mutation is performed by selecting one operator and appropriate number of operands as shown in Figure 5-9. The explorative power of a mutation is affected by not only the mutation rate but also the individuals' search space. Each individual's search space is constrained by the following factors: (C1) the number of mutation operators; (C2) the number of features in the individual, i.e., the number of features for the first operand; and (C3) the number of candidate features for the second operand.

The number of mutation operators is predefined, but the numbers of features for the first operand and second operand vary. The number of first operands is controlled by the initialization, crossover, inclusion and exclusion mutation operators. As discussed in Section 1, initializing each individual with a single feature limits the individual's search space at the beginning of the EA. The crossover-by-addition method and the inclusion mutation operator increase the individual's search space by adding a feature to the individual, whereas the exclusion mutation operator decreases the individual's search space. The three mutation schemes: EAFCS-EX, EAFCS-IN, and EAFCS-BN generate a different search space for an individual by having different sets of candidate features as the second operand. Figure 5-9 show the flow of mutation (red boxes), candidate operands and operators, and components that affect the individual's search space.

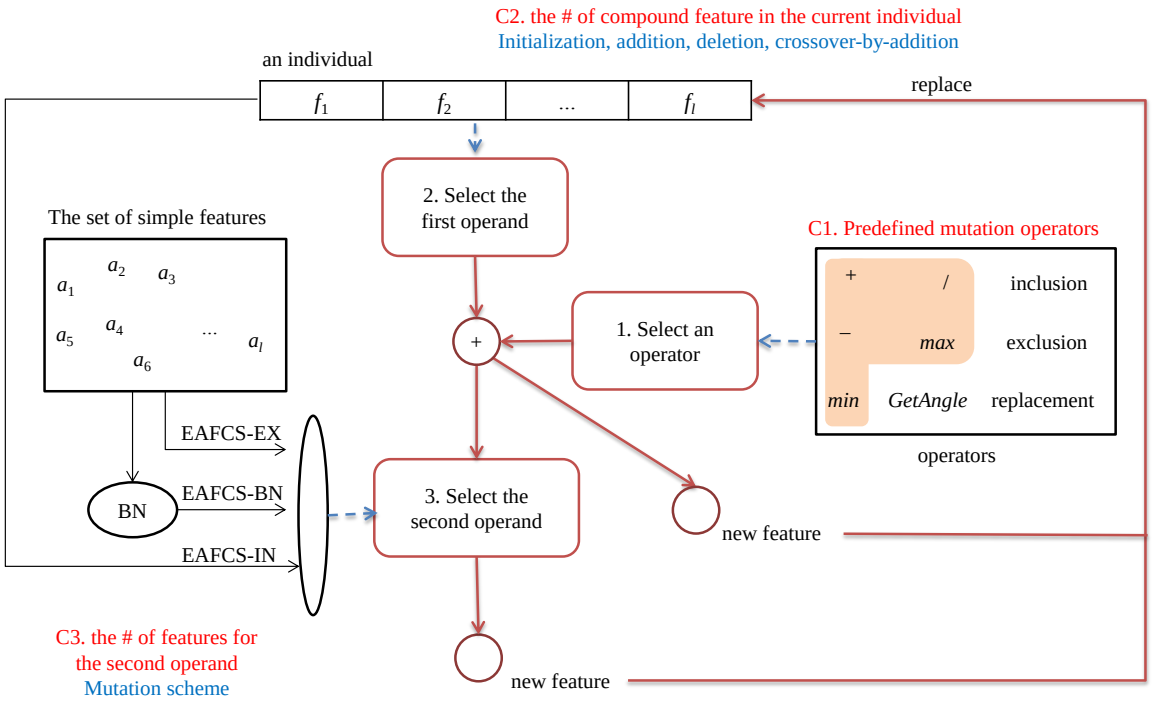


Figure 5-9. The factors that control the individual's search space

4. Experimental Results

This section evaluates the performance of the proposed EA in terms of prediction accuracy and complexity. The features extracted from the event-based segmentation with N_{init} = 400 and N_{final} = 40 are used. These values have shown the best performance with a Bayesian network classifier (Table 4-3). Thus, the gesture data contains 117 simple features and 10 instances for each class for a total of 80 instances. The performance accuracy and complexity of the EAFCS is compared with the two cases, where all simple features are used for learning, namely ALL, and the best subset of simple features selected from randomly drawn 1,000 feature subsets is used for learning, namely Random. For EAFCS-BN, the Bayesian network is built using K2 algorithm, and three parents are allowed at maximum. P_{MB} is set to a large value 0.9, since EAFCS-EX will be used to select the second operand if there is no feature that satisfies the restrictions shown in Table 5-1 using EAFCS-BN.

The first subsection aims to find the proper population size, and the second subsection studies the behavior of the specified crossover and three mutation schemes. The third subsection studies the performance of EAFCS under various combinations of crossover and mutation rates, and compares the accuracy and complexity of the extracted features with ALL and Random.

A. Experimental settings

One of the difficulties in using an evolutionary algorithm is that parameters, such as the size of population (i.e., the number of individuals in a population), mutation and crossover rate, need to be carefully tuned to balance exploration and exploitation [81, 82]. Although a large population size may enhance the performance of an EA, it requires a more number of fitness evaluations for each generation, and takes more time for convergence [83]. By setting the

mutation rate and crossover rate to 0.1 and 0.4 respectively, whose values were selected based on my pilot study, I compared the performance of the EAs, i.e., EAFCS-EX, EAFCS-IN, and EAFCS-BN, as the population size increases (e.g., 30, 50, 100, and 200). All EAs runs for 200 generations. The naïve Bayes classifier is used for both training and testing.

Five 5-fold cross-validations are performed, and for each fold an EA runs for five times with different random seeds. Thus, a 5-fold cross-validation is performed 25 times in total. Table 5-3 shows the average *F*-measures of five-fold cross-validations for each population size and three different mutation methods. I increase the population size until there is no significant improvement in all mutation schemes. The bold entries represents that there is a significant difference between the smaller and larger population size, i.e., between 30 and 50, 50 and 100, and 100 and 200, based on the two-tailed *t*-test at $\alpha = 0.05$. From the results shown in Table 5-3, it appears that using 100 as the population size provides a reasonable tradeoff between performance improvement and computational cost. Given 117 simple features in the data, an initial population with 100 individuals will contains approximately 85% of the original simple features.

Table 5-3. The average *F*-measures with different population sizes

POPULATION SIZE	EAFCS-EX	EAFCS-IN	EAFCS-BN
30	0.9432	0.8973	0.9660
50	0.9622	**0.9522**	0.9691
100	0.9736	**0.9717**	**0.9821**
200	0.9834	0.9761	0.9777

To check the convergence of EAs, Figure 5-10 shows the fitness of the best-so-far solutions for three population sizes, i.e., 50, 100, and 200. The evolution of the EAFCS-BN is

represented since it has shown the best performance when P_{co} = 0.4 and P_{mut} = 0.1, but the EAFCS-EX and the EAFCS-IN show the similar results. The EA with the population size of 200 can converge to a slightly better fitness than the EA with the population size of 100, but this small improvement in fitness does not seem to result in significant performance improvement for unseen testing data as shown in Table 5-3. Also, the EAs can converge within 100 generations.

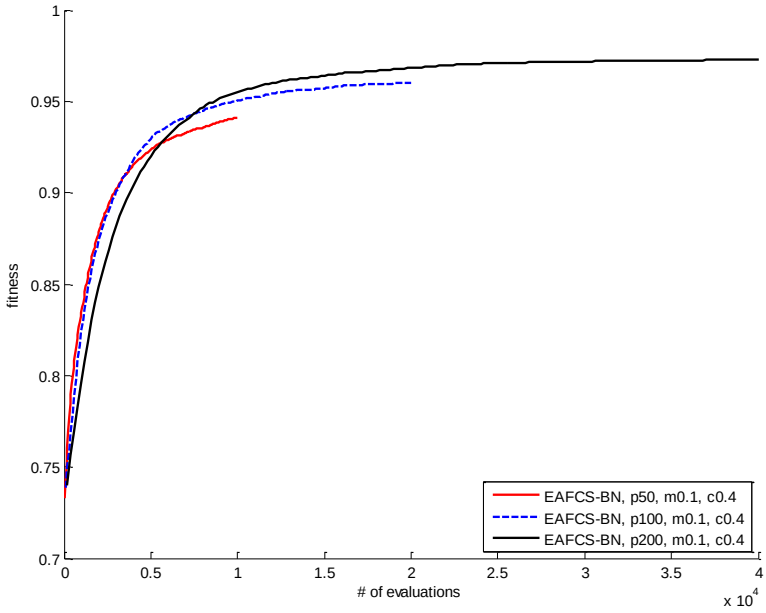


Figure 5-10. The fitness of the best-so-far solutions of EAFCS-BN

Therefore, further analysis will use 100 as the population size, and the EA will terminate after 100 generations. This requires 10,000 fitness evaluations in total. Using a machine with Intel i5 2.5GHz CPU and 6GB memory, the fitness evaluation using 4-fold cross-validation based on naïve Bayes classifier takes approximately 45.4ms, and the total evaluation time for an EA run will be approximately 7.6 minutes. Thus, the computational time for these evaluations seems to be affordable.

B. Analysis of different mutation methods

To see the behavior of three mutation schemes and crossover operator, each EAFCS-EX, EAFCS-IN and EAFCS-BN is performed using four combinations of P_{mut} = {0.1, 0.5} and P_{co} = {0.1, 0.5}. Figure 5-11, Figure 5-12 and Figure 5-13 show the box plots for the average *F*-measures of the EAs with different combinations of P_{mut} and P_{co}. Table 5-4 shows the mean values of the average *F*-measure for each of the three mutation schemes. The bolded entry indicates that the performance is significantly better than the corresponding EAFCS-IN, and * marks represent the performance of the entries are significantly different (better) from other mutation methods under the same settings for P_{mut} and P_{co}. The paired two-tailed *t*-test at a significant level $\alpha = 0.05$ is used to test the difference. EAFCS-EX and EAFCS-IN show very poor performance when P_{co} is small, while EAFCS-BN is less sensitive to P_{co}. Although the best performance achieved by EAFCS-EX and EAFCS-BN is similar, the performance of EAFCS-IN is inferior to other two mutation schemes.

Table 5-4. The average *F*-measure of different mutation methods

P_{MUT}	P_{CO}	EAFCS-EX	EAFCS-IN	EAFCS-BN
0.1	0.1	**0.8814**	0.8346	**0.9353***
0.1	0.5	**0.9850**	0.9672	**0.9831**
0.5	0.1	**0.8719**	0.8075	**0.9443***
0.5	0.5	0.9650	0.9683	**0.9774***

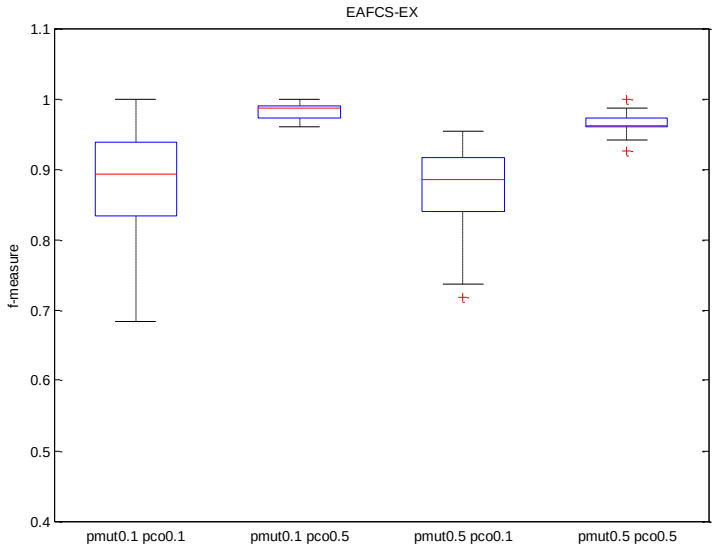


Figure 5-11. The impact of mutation and crossover to EAFCS-EX

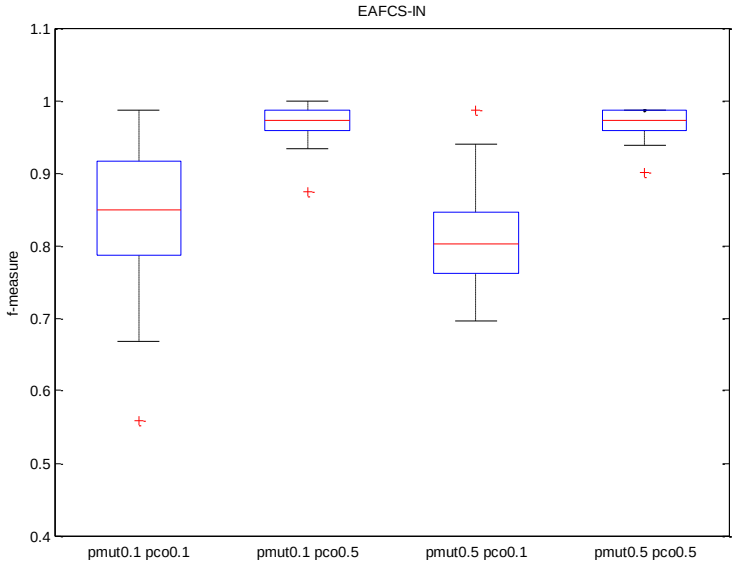


Figure 5-12. The impact of mutation and crossover to EAFCS-IN

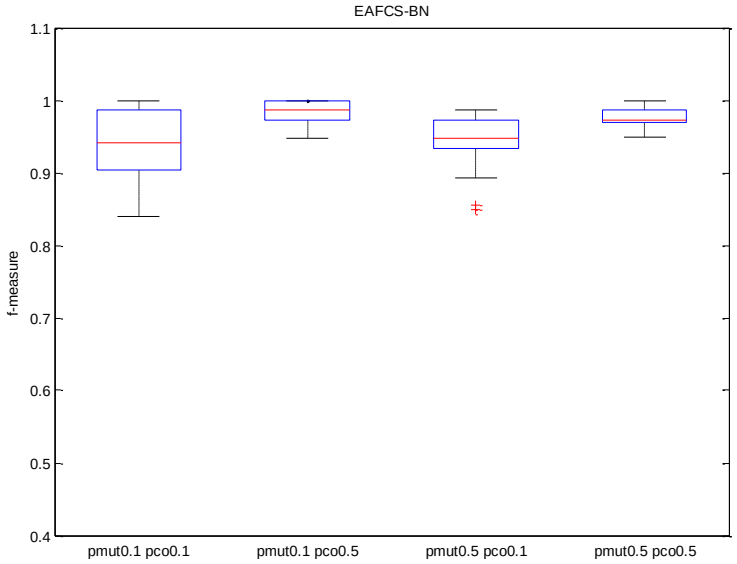


Figure 5-13. The impact of mutation and crossover to EAFCS-BN

To better understand the poor performance of EAFCS-IN, Figure 5-14 plots the evolution of the best-so-far solutions when $P_{mut} = 0.5$ and $P_{co} = 0.5$. The large value is used for P_{mut} to see the impact of different mutation schemes. It seems that the EAFCS-IN converges to a lower fitness than the EAFCS-BN and the EAFCS-EX. The reason might be that the EAFCS-IN introduces a smaller number of new simple features to the population than the EAFCS-EX and the EAFCS-BN because it selects two operands from the current individual, not from the set of original simple features. Also, by replacing two operands from the individual by a single compound feature, the two-operand mutation operators might have reduced the potential operands available within the individual and thereby reduced the diversity of the features in the population.

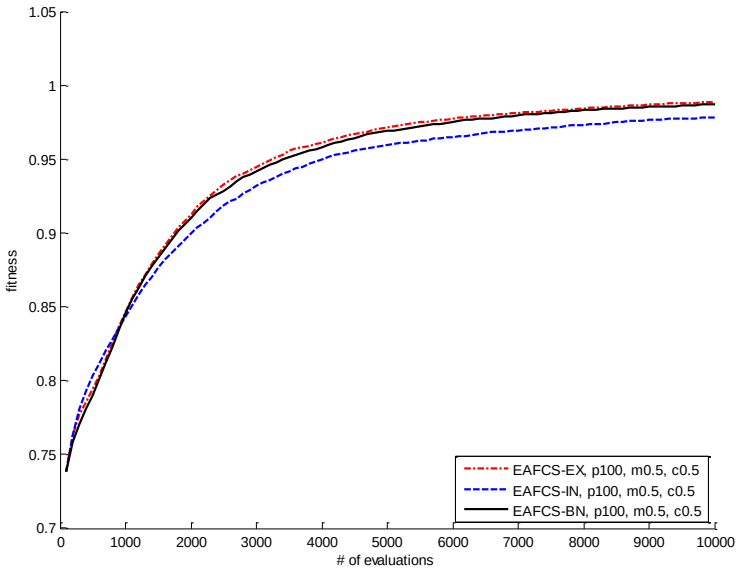


Figure 5-14. The fitness of three mutation schemes

To understand the poor performance of the EAFCS-EX and the EAFCS-IN when P_{co} is small, Figure 5-15 depicts the fitness of the best-so-far solution for three mutation schemes when P_{mut} = 0.5 and P_{co} = 0.1. In contrast to my expectation that EAFCS-BN would quickly find promising solutions by using the information about the relationship between the features, the result shows that EAFCS-BN's fitness improves very slowly at the beginning of the run.

The reason might be that the features, which are important but independent from the features in the current individual, are less likely to be included to the individual in EAFCS-BN than in EAFCS-EX or EAFCS-IN. In EAFCS-EX and EAFCS-IN, independent features can be selected as the second operand of the two-operand operators. However, in EAFCS-BN, independent features can only be included in the individual through the inclusion or replacement operators, since they will not be included by the MBs of any feature in the individual. Since there are more number of two-operand operators than the inclusion or replacement operators in my

mutation operators, EAFCS-BN might have not been able to quickly include those independent features to the individuals.

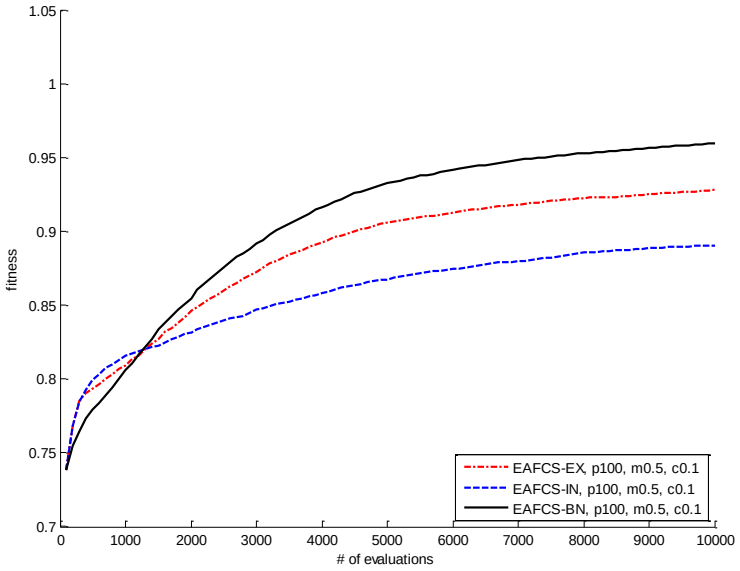


Figure 5-15. The evolution of three mutation schemes when P_{co} is small

For example, assume that the simple features a_1 and a_2 are independent but both are important features for classification. Also assume that an individual currently has one feature a_1 (Figure 5-16 (b)), and the Markov blanket of a_1 does not contain a_2, and vice versa (Figure 5-16 (a)). The ideal individual might be the one that contains two features, a_1 and a_2, as independent two *f*s (Figure 5-16 (c)). Since there are more two-operand operators than inclusion or replacement operators, EAFCS-EX and EAFCS-IN are more likely to select a_2 as an operand for the two-operand operator than for the inclusion or replacement operator (e.g., Figure 5-16 (d)). However, since a_2 is not in the MB of a_1, EAFCS-BN will not be able to select a_2 as the second operand of a_1. Instead, EAFCS-BN will select a_2 as an operand for inclusion or replacement operator as shown in Figure 5-16 (c).

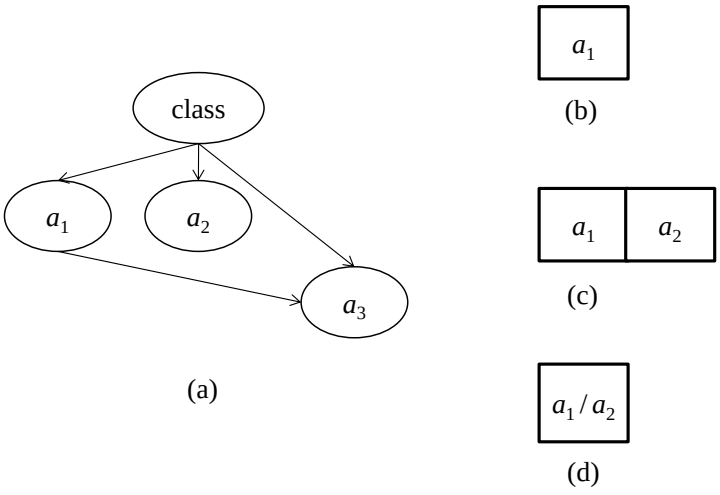


Figure 5-16. An example of including a useful yet independent feature in EAFCS-BN and EAFCS-EX

Although this slows downs the convergence of the EAFC-BN, this can provide a potential for the further improvement in future generations as shown in Figure 5-15. In other words, combining useful yet independent features can be beneficial in the short-term as shown in EAFCS-EX and EAFCS-IN, but it may not be favorable in the long-term. Since EAFCS-BN includes independent features only through inclusion or replacement operators, individuals will have the independent and important features (e.g., a_1) as independent *f*s rather than as parts of compound features. This can help the EA to construct more promising features on top of the new simple feature. More importantly, by increasing the number of *f*s in the individual, the individual's search space can be expanded.

Figure 5-17 shows the number of compound features (*f*s) in the best-so-far solution. The number of compound features of EAFCS-BN is larger than that of EAFCS-EX and EAFCS-IN. Considering that the inclusion is the only mutation operator that increases the number of compound features (*f*s) in the individual, the result indicates that the inclusion operator is more

effective in EAFCS-BN than EAFCS-EX and EAFCS-IN, albeit the three mutation schemes use the same method to select a simple feature for the inclusion operator. I hypothesize that important yet independent features can be already included as parts of compound features in EAFCS-EX, and this interferes such important yet independent features to be (re-)included in the individual as a new feature, *f*.

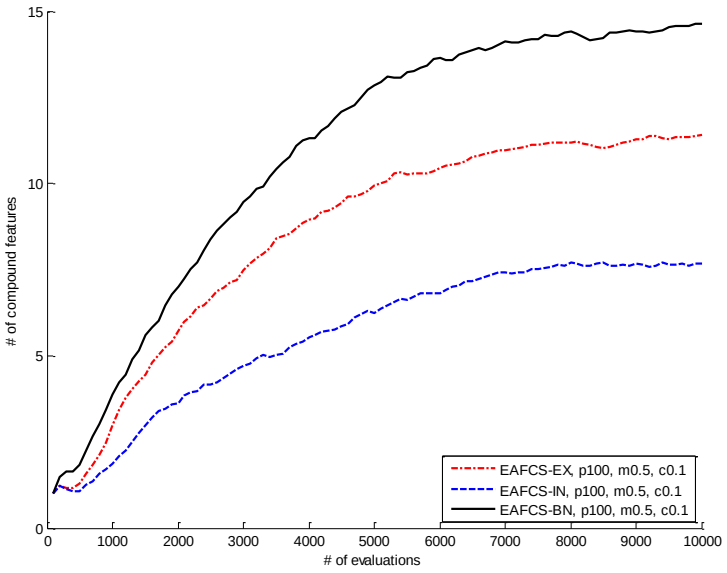


Figure 5-17. The number of compound features in the best-so-far solutions

To support my hypotheses, Figure 5-18 depicts each of the following cases: (1) including a feature that exists in the current individual's compound features results in success, namely EXISTING SUCCESSFUL; (2) including an existing feature results in failure, namely EXISTING FAILURE; (3) including a new feature that does not exist in the current individual's compound features results in success, namely NEW SUCCESSFUL; and (4) including a new feature results in failure, namely NEW FAILURE. The mutation is considered to be successful when it generates offspring with higher fitness. The *x*-axis represents the number of generations,

while the y-axis represents the ratio of the number of corresponding cases to the total number of inclusion mutations. The results are collected from 25 runs of EAFCS-BN and EAFCS-EX with $P_{mut} = 0.5$ and $P_{co} = 0.1$. The blue 'x's and the red 'o's represent EAFCS-BN and EAFCS-EX, respectively.

It appears that adding a new feature is more likely to be successful in mutation than adding an existing feature by providing new information, which was not presented in the current individual's feature set. When EAFCS-BN and EAFCS-EX are compared in NEW SUCESSFUL and NEW FARILURE, new features are more effective in EAFCS-BN than EAFCS-EX. The new features in EAFCS-BN might be more predictive feature than the new features in EAFCS-EX, and such good features might have been already included in the current individual of EAFCS-EX as parts of compound features. Since adding existing feature is likely to result in failure, the important yet independent features that are already included as parts of compound features are hard to be re-included to the individual as a new *f*. This result supports my hypotheses. By adding useful independent features through successful mutations, EAFCS-BN appears to have an ability to increase the number of compound features in the individual to the proper size for good classification.

However, as shown in Table 5-4, the crossover operator can compensate for the limitations of EAFCS-EX and EAFCS-IN. Recall that the crossover operation is performed in one of two ways: crossover-by-exchange or crossover-by-addition. Among the two crossover operations, crossover-by-addition might have helped EAFCS-EX and EAFCS-IN to expand the search space originally restricted by the set of features in the current individual, and to find the proper size for the feature set.

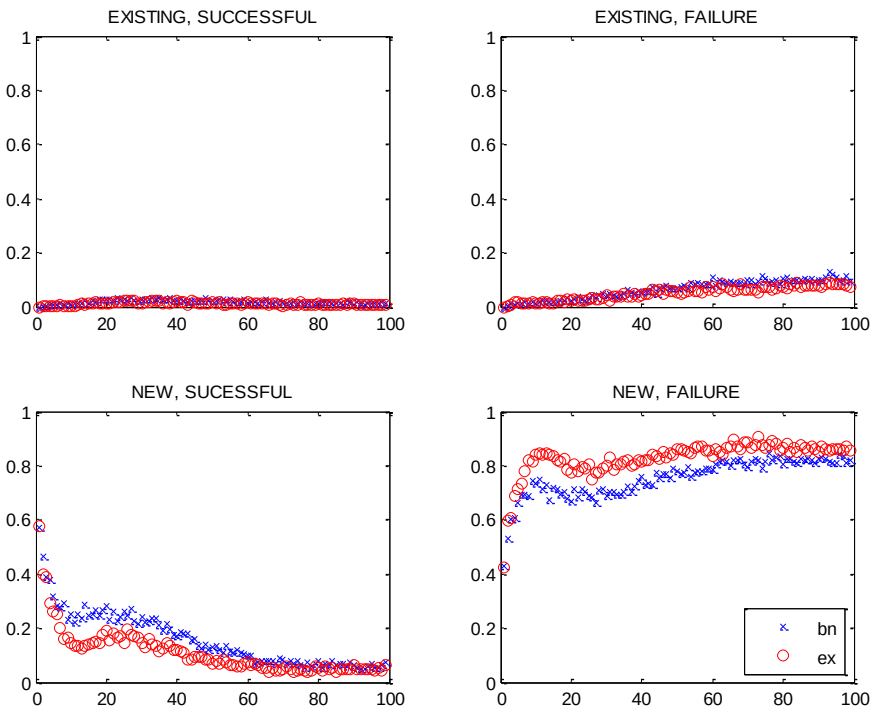


Figure 5-18. The results of including independent simple features

C. Impact of mutation and crossover rate

To show the performance of the EAs with different values of P_{mut} and P_{co}, Figure 5-19 plots the average *F*-measure of the 5-fold cross-validations of EAFCS-EX across different values of P_{mut} and P_{co} and Table 5-5 shows the value for each point. For each setting, five 5-fold cross-validations are performed with five different random seeds for a total of 25 5-fold cross-validations. The comparison between EAFCS-EX and Random is depicted in Figure 5-19, since Random performs better than ALL, which will be shown shortly. The red and blue mesh represent the performance of EAFCS-EX and Random, respectively, and the red (blue) star marks represent that EAFCS-EX (Random) is significantly better than Random (EAFCS-EX) based on the two-tailed paired *t*-test at $\alpha = 0.05$. The red marks are shown in bold in Table 5-5. EAFCS-EX

can perform significantly better than Random, and this good performance can be achieved when P_{mut} is small and P_{co} is large. EAFCS-EX performs especially well when P_{mut} is around 0.1 and P_{co} is larger than 0.5. It seems that new features are gradually added to the EA with small P_{mut}, while the crossover operators find good combinations of features in a population with large P_{co}. By exchanging or adding only one feature at a time, my crossover operator makes the parents' feature set less destructive but limits big jumps as compared to a general crossover operator. Thus to compensate for the low explorative power of my crossover operator, P_{co} might need to be set with a large value. The best average *F*-measure of the naïve Bayes classifier, 0.985, is achieved when P_{mut} = 0.1 and P_{co} = 0.5. The average *F*-measure of the naïve Bayes classifier achieved by the Random is 0.954.

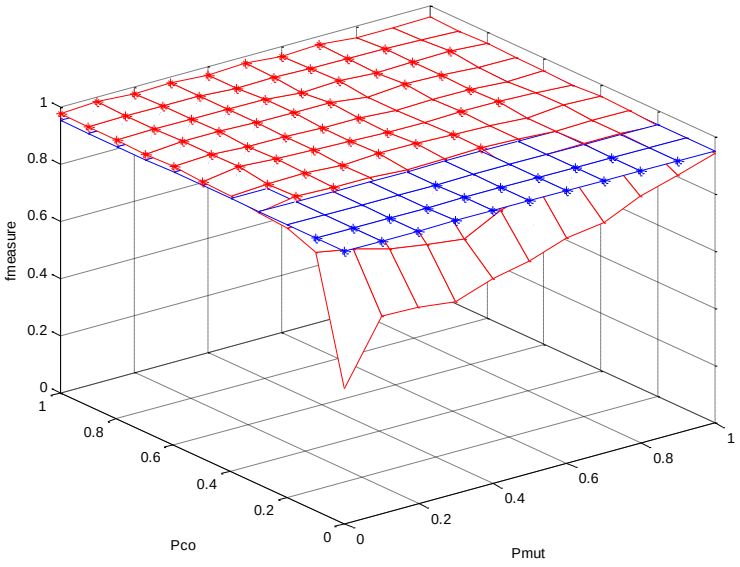


Figure 5-19. The *F*-measure of EAFCS-EX (red) and Random (blue) for different values of P_{co} and P_{mut} using EAFCS-EX

Table 5-5. The average *F*-measure of Naïve Bayes with statistical tests using EAFCS-EX

P_{co} \ P_{mut}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	0.471	0.691	0.687	0.669	0.716	0.741	0.790	0.807	0.862	0.903	0.946
0.1	0.904	0.881	0.846	0.824	0.809	0.872	0.918	0.904	0.940	0.954	0.953
0.2	0.941	0.938	0.950	0.933	0.924	0.937	0.944	0.953	0.957	0.956	0.949
0.3	0.949	**0.971**	**0.967**	0.967	0.958	0.962	**0.968**	0.964	0.956	0.959	0.957
0.4	0.961	**0.970**	**0.974**	**0.966**	**0.973**	**0.970**	**0.968**	0.963	0.954	0.964	0.961
0.5	**0.968**	**0.985**	**0.975**	**0.972**	**0.968**	**0.965**	0.965	**0.965**	**0.967**	0.960	0.955
0.6	**0.973**	**0.982**	**0.972**	**0.973**	**0.969**	**0.968**	0.960	0.962	**0.965**	0.956	0.960
0.7	**0.971**	**0.981**	**0.982**	**0.978**	**0.975**	**0.978**	0.962	**0.970**	**0.967**	0.957	0.959
0.8	**0.976**	**0.980**	**0.977**	**0.979**	**0.975**	**0.965**	**0.974**	**0.965**	**0.968**	**0.969**	0.963
0.9	**0.974**	**0.980**	**0.977**	**0.979**	**0.970**	**0.974**	**0.970**	**0.967**	**0.968**	0.965	0.959
1	0.977	**0.984**	**0.976**	**0.978**	**0.971**	**0.977**	**0.964**	**0.973**	0.962	0.962	0.963

Figure 5-20 and Figure 5-21 are the box plots comparing the average *F*-measures and complexity of the extracted features for ALL, Random, and EAFCS-EX with $P_{mut} = 0.1$ and $P_{co} = 0.5$. The complexity is measured by Equation 5-3, and smaller is better. The numerical values are shown in Table 5-6. Overall, EAFCS-EX appears to achieve significantly better performance accuracy than ALL and Random with a simpler feature set.

Table 5-6. The average *F*-measures and complexity of All, Random, and EAFCS-EX

	ALL	**RANDOM**	**EAFCS-EX**
Accuracy	0.9453 ± 0.0079	0.9535 ± 0.0158	0.9850 ± 0.0116
complexity	1 ± 0	0.6537 ± 0.0568	0.1827 ± 0.0159

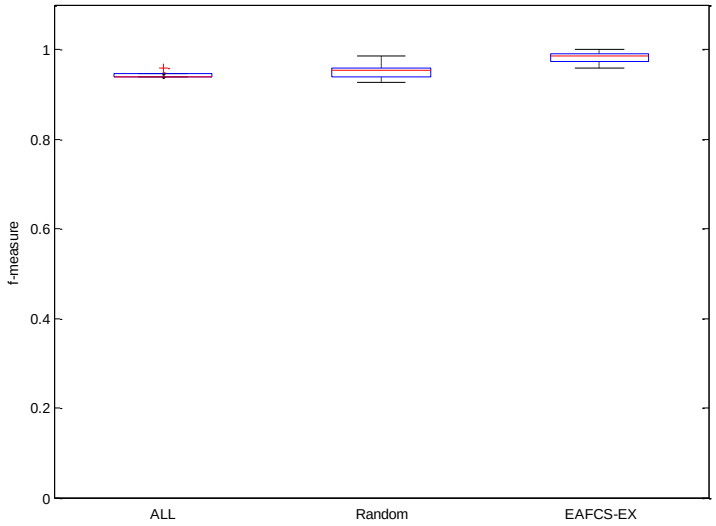


Figure 5-20. The performance accuracy of All, Random and EAFCS-EX

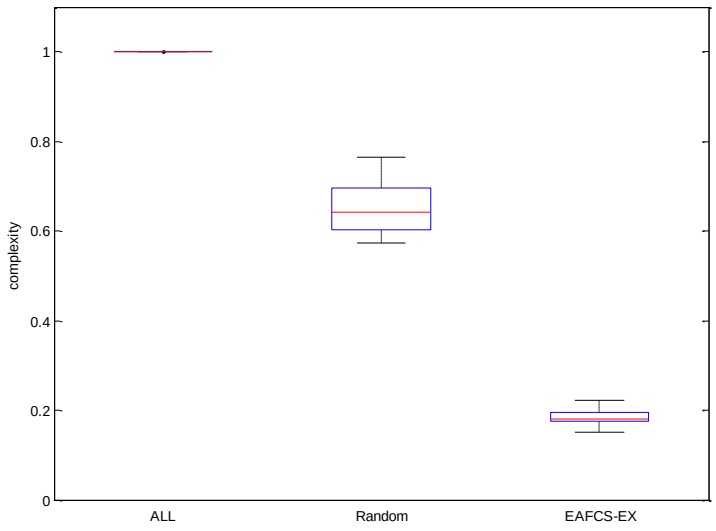


Figure 5-21. The complexities of All, Random and EAFCS-EX (measured by Equation 5-3)

Figure 5-22 shows the evolution of the number of compound features in the best so far solutions when P_{mut} = 0.1 and P_{co} = 0.5. As in Figure 5-19, each point is the average of 25 5-fold cross-validations. Note that the number of compound features converges to approximately 13, which suggests that 13 might be the appropriate number of features in this gesture classification problem.

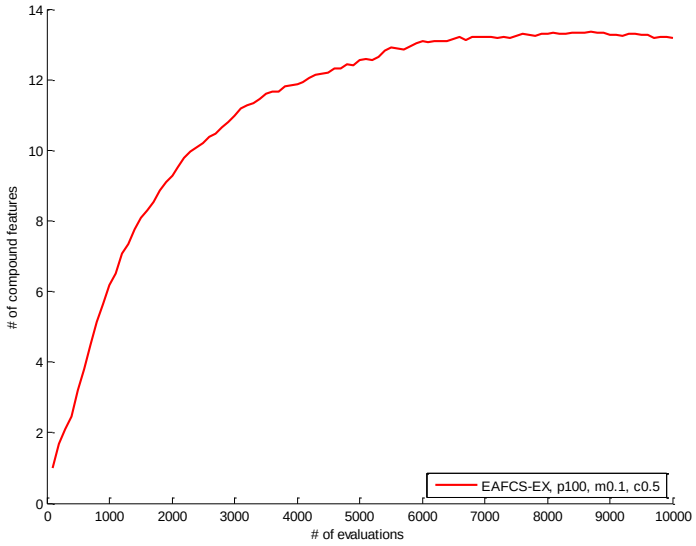


Figure 5-22. The number of compound features in the best-so-far solutions

Figure 5-23 and Table 5-7 show the performance of EAFCS-BN across different settings of P_{mut} and P_{co}. The result is similar to EAFCS-EX in that the most promising performance is achieved when P_{mut} is small and P_{co} is large. The best performance, 0.986, is achieved when P_{co} = 0.1 and P_{mut} = 0.8. There is no significant difference between the best performances achieved by EAFCS-EX and EAFCS-BN. However, EAFCS-BN seems to be less sensitive to the smaller values of P_{co} than EAFCS-EX.

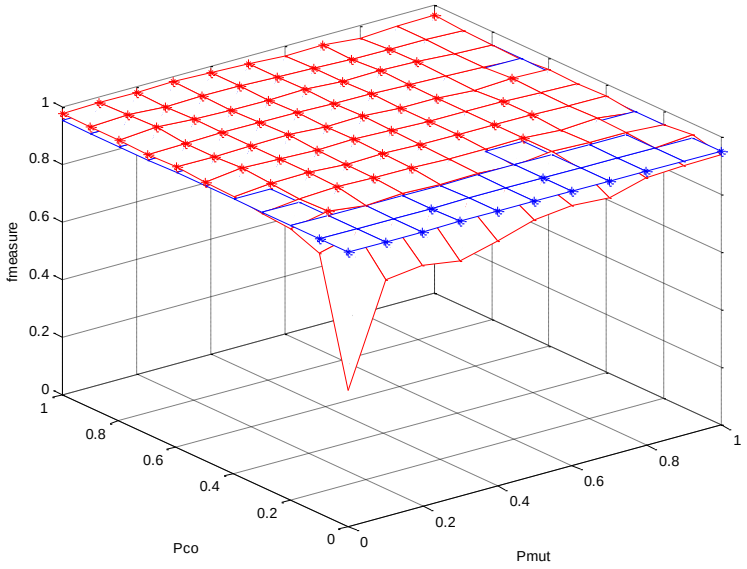


Figure 5-23. The *F*-measure of Naïve Bayes for different values of P_{co} and P_{mut} using EAFCS-BN

Table 5-7. The average *F*-measure of Naïve Bayes with statistical tests using EAFCS-BN

P_{mut} / P_{co}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	0.471	0.821	0.835	0.818	0.849	0.888	0.905	0.895	0.931	0.938	0.940
0.1	0.904	0.939	0.935	0.917	0.926	0.942	0.935	0.949	0.948	0.954	0.963
0.2	0.941	**0.969**	0.954	0.961	0.954	0.960	0.953	0.964	0.959	0.959	0.952
0.3	0.949	0.967	**0.973**	**0.970**	**0.968**	0.962	0.963	**0.965**	0.959	0.965	0.951
0.4	0.961	**0.975**	**0.976**	**0.967**	**0.972**	**0.968**	**0.969**	**0.968**	**0.967**	0.961	0.962
0.5	**0.968**	**0.981**	**0.976**	**0.974**	**0.978**	**0.973**	**0.968**	0.965	**0.968**	0.963	0.957
0.6	**0.973**	**0.974**	**0.978**	**0.979**	**0.974**	**0.975**	**0.968**	**0.968**	0.958	**0.968**	0.961
0.7	**0.971**	**0.975**	**0.979**	**0.977**	**0.976**	**0.972**	**0.971**	**0.968**	0.964	0.960	0.951
0.8	**0.976**	**0.986**	**0.979**	**0.974**	**0.973**	**0.978**	**0.969**	**0.970**	**0.967**	0.959	0.955
0.9	**0.974**	**0.981**	**0.974**	**0.977**	**0.978**	**0.975**	**0.969**	**0.964**	**0.964**	0.962	0.957
1	0.977	**0.980**	**0.979**	**0.982**	**0.978**	**0.972**	0.965	**0.968**	0.959	0.964	**0.968**

The performance difference between EAFCS-EX and EAFCS-BN for each combination of P_{mut} and P_{co} is shown in Figure 5-24, where red and blue represents EAFCS-BN and EAFCS-EX, respectively. The star marks represent that the difference is significant. As discussed in the previous subsection, EAFCS-BN can perform better than EAFCS-EX when the crossover rate is small, which indicates that EAFCS-BN might be the better mutation method.

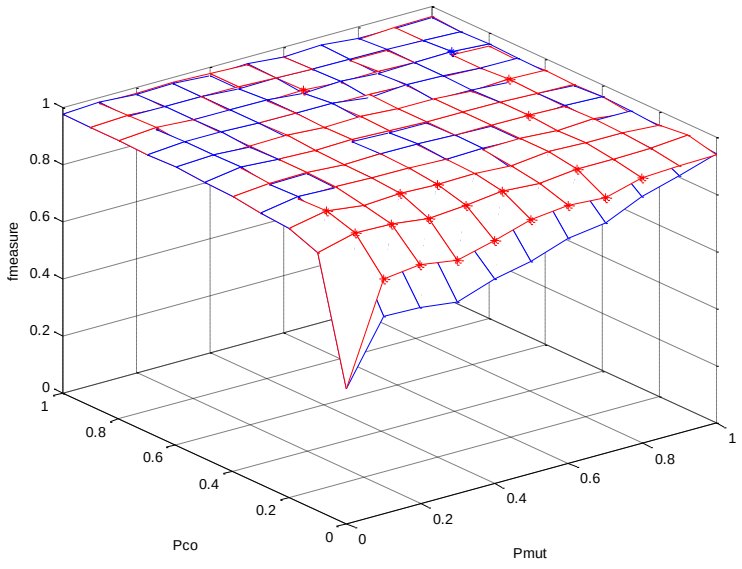


Figure 5-24. EAFCS-BN (red) and EAFCS-EX (blue)

5. Qualitative analysis of the extracted features

To understand the extracted features, I appraise the features extracted from one run of an EA to more fully understand 1) the type of gestures that the different classifiers have trouble with, and 2) potential semantic meanings that could be extracted from the compound features. Table 5-8 shows the confusion matrix for one left out fold in a five-fold cross-validation when all

features are used for learning and a naïve Bayes classifier is used. In this particular fold, two fast left to right gestures (gesture 6) are misclassified as slow left to right gestures (gesture 2), and one fast clock-wise movement (gesture 7) is misclassified as slow clock-wise gesture (gesture 3). The naïve Bayes classifier could not distinguish between the gestures with different speeds.

Table 5-8. The confusion matrix with all features

actual predicted	1	2	3	4	5	6	7	8
1	2							
2		2						
3			2					
4				2				
5					2			
6		2				0		
7			1				1	
8								2

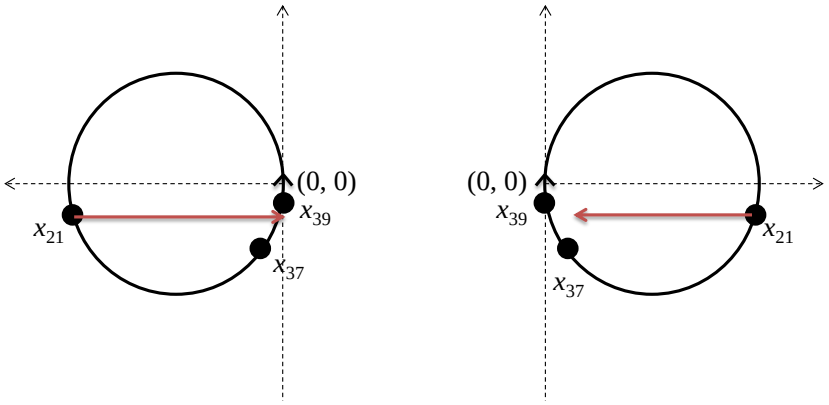
However, all instances in this left out fold can be correctly classified by using features extracted from EAFCS-EX. The extracted features are shown in Table 5-9. The extracted features include several timestamps at the end of gesture rather than the timestamps at the beginning of the gesture. The 5^{th} and 6^{th} features that are constructed by adding two timestamps at the end of gestures might make the difference between the same gestures where only the speed differs.

Table 5-9. The extracted features for a particular run of an EA

NO.	THE EXTRACTED FEATURES
1	t_{22}
2	t_{34}
3	t_{36}
4	$t_7 - t_{39}$
5	$t_{39} + t_{23}$
6	$t_{34} + t_{22}$
7	x_{25} / t_{22}
8	$max(x_{39}, x_{37}) - x_{21}$
9	$min(max(x_{39} , x_{37}) , x_{32})$
10	$max((x_{23} + x_{38}) , x_{16})$
11	$max(x_{39} , x_{37}) - x_{32}$
12	$Angle(x_{22}, y_{22})$

The 7th feature is speed feature x_{25} / t_{22}, which comes very close to being the average speed for the first half of the gesture. Although there are a number of possible features to be selected for the second operand of x_{25}, it is interesting that the selected operand t_{22} is the timestamp that occurred at a similar time point in the gesture. The 8th feature $max(x_{39}, x_{37}) - x_{21}$

represents the movement during the last half of the gesture. Its interpretation in the circular movements is shown in Figure 5-25 and their values for each class are plotted in Figure 5-26. The class numbers are the gesture numbers shown in Figure 3-1, and each point in Figure 5-26 is the transformed instance from the whole data (i.e., ten instances for each class). The 11th feature has the similar interpretation.



(a) class 3 and 7 (b) class 4 and 8

Figure 5-25. An interpretation of $max(x_{39}, x_{37}) - x_{21}$ in the circular movements

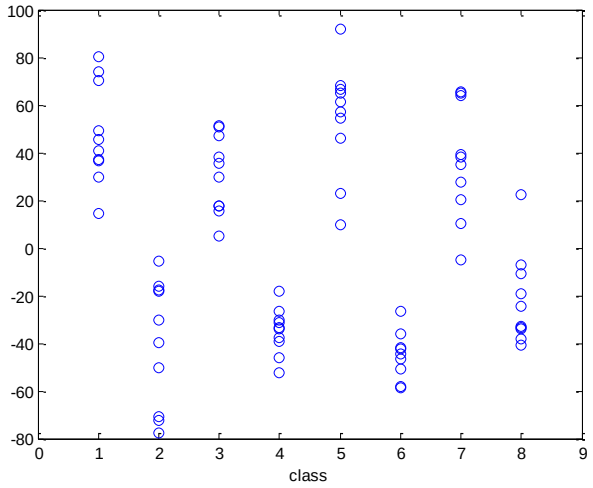


Figure 5-26. The values of $max(x_{39}, x_{37}) - x_{21}$

The 9th feature classifies the shapes of the different gestures, where the values for each feature and compound feature in the 9th feature are shown in Figure 5-27. With x_{39}, two circular movements can be clearly distinguished from right-to-left (class1 and class5) and left-to-right (class2 and class6) gestures, but two circular movements with different directions cannot be distinguished since the finger returns to the starting point at the end of the gestures. By constructing $max(x_{39}, x_{37})$, clock-wise and counter clock-wise (class 3 and 4) movements will be classified slightly better than simply using x_{39}. However, it is still insufficient to classify two circular movements with different directions. The finally constructed feature $min(max(x_{39}, x_{27}), x_{32})$, seems to well distinguish four different shapes of gestures. A simple feature x_{32} can distinguish two circular movements with different direction, but can confuse circular movements with straight movements.

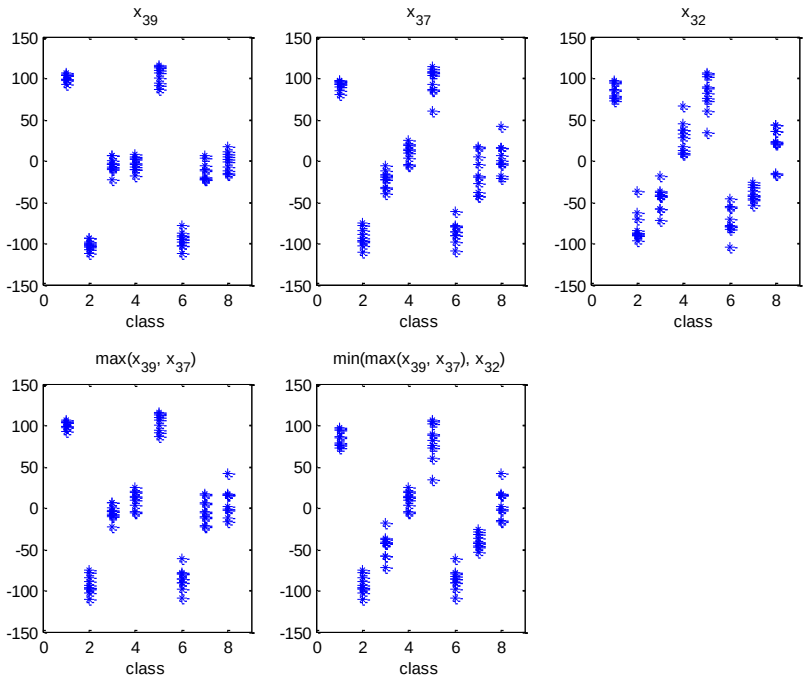


Figure 5-27. Construction of $min(max(x_{39}, x_{37}), x_{32})$

Similarly, values for the features that construct the 10th feature are plotted in Figure 5-28. The summation of two features x_{23} and x_{38} could notably distinguish different gesture shapes, albeit no significant change is made by the max operation. Lastly, the angle feature is also included in the extracted feature set. This is the angle right after the middle of the gestures, and appears to differentiate right-to-left gestures (class 1) from left-to-right gestures (class2). However, right-to-left/left-to-right gestures (class 1 and 2) can be confused with counter clock-wise/clock-wise gestures (class 4 and 3).

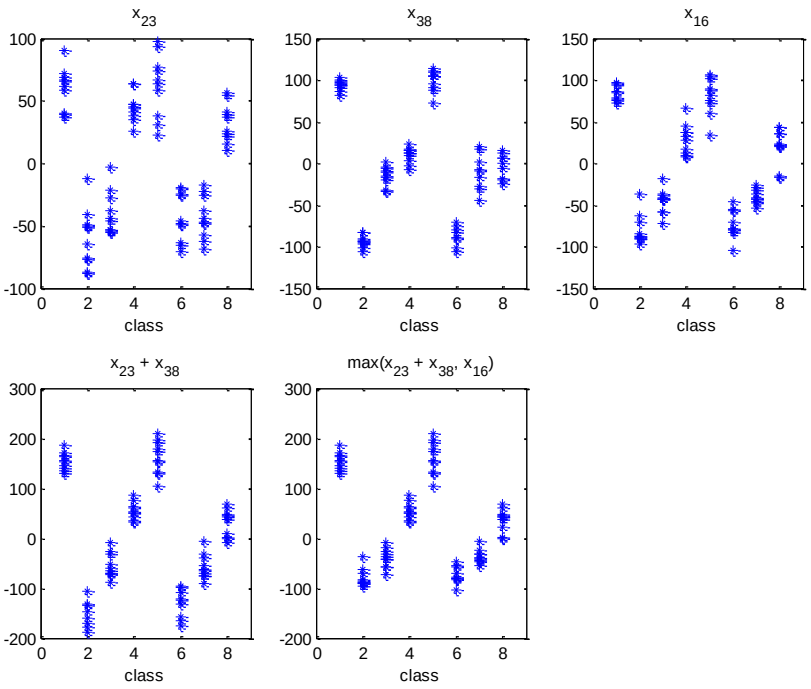


Figure 5-28. Construction of $max(x_{23} + x_{38} , x_{16})$

In some EA runs angular velocity was also extracted. Overall, the EA seems to be able to construct meaningful compound features and provide interesting relationships between the features.

6. Summary

This Chapter proposed an evolutionary-based approach for constructing and selecting a set of compound and simple features. The representation, genetic operators such as mutation and crossover, and fitness evaluation methods are specified for my gesture recognition problem. An individual is represented by a set of simple or compound features; mutation constructs higher level compound features and adds or removes features from an individual; and crossover operator aims to find a good combination of features. The proposed EA could improve the accuracy of a naïve Bayes classifier with a simpler feature set. The performance accuracy was significantly better than using all simple features or a set of simple features that are selected from 1,000 randomly generated feature subsets.

Three mutation schemes: EAFCS-EX, EAFCS-IN, and EAFCS-BN are compared. The EAFCS-EX selects one operand from the current individual and the second operand from the set of simple features, while the EAFCS-IN selects both operands from the current individual. The EAFCS-BN selects the second feature based on the structure of a Bayesian network built using the simple features. EAFCS-BN can perform better than EAFCS-EX and EAFCS-IN when the crossover rate is small by adding useful yet independent features to individual as independent feature *f*s. However, while the performance of EAFCS-EX and EAFCS-BN is similar when the crossover rate is large, both perform better than EAFCS-IN.

I analyzed how an individual's search space is restricted by the mutation and crossover operators. In addition to the mutation rate, an individual's search space is affected by the following factors: (C1) the number of mutation operators, (C2) the number of features in the current individual, i.e., the number of features for the first operand and (C3) the number of features for the second operand. The proposed EA starts with very limited search space in terms of C2 by initializing individuals with a single simple feature, and expands the search space

constrained by (C2) by adding features (*f*s) to individuals through crossover and the inclusion mutation operator. Different mutation schemes have different search space in terms of C3. EAFCS-EX is less restrictive than EAFCS-IN and EAFCS-BN. Finally, I analyzed the semantic meanings of the extracted features, and found that my EA can construct meaningful high level compound features and serve as a knowledge discovery process.

Chapter 6

Conclusion

Gesture recognition is an emerging technology for human computer interaction, but it has not been widely applied to real-world applications to the mobile devices. This dissertation formulates a speed sensitive finger gesture recognition problem to explore real-world applications on personal mobile devices. In particular, I use a novel camera called dynamic vision sensor (DVS) camera, which only responses to pixels with temporal luminance change. As with other machine learning problems, using a good feature set is critical for gesture recognition performance. I study two feature extraction methods, namely local and global feature extractions.

Different from the frame-based conventional camera, DVS camera outputs a long sequence of events for pixels with temporal luminance change detected at the microsecond-level granularity. Although frame-based comparison is not needed to detect finger movement, using the raw data is computationally prohibitive due to the large number of events. Instead, I first extract important features by segmentation, namely local feature extraction. The proposed method first divides the sequence of events into segments. These segments either have the same time interval, and are called the time-based, or the same number of events, and are called the event-based. The method then repeatedly augments the neighboring segments based on the distance between their event distributions. The experimental results have shown that my proposed event-based segmentation can extract the important patterns of the gesture, and outperform the general time-based segmentation. I also found that a Bayesian network classifier can outperform a hidden Markov model when features are well extracted using event-based segmentation.

My second feature extraction technique, a global feature extraction method, aims to construct high level compound features by transforming the locally extracted features and finding

the interactions between the simple features. I used an evolutionary algorithm (EA) to construct and find a good set of compound and simple features and defined representation, fitness evaluation, and genetic operators including mutation and crossover specified to the gesture recognition problem. I proposed three mutation schemes: EAFCS-EX, EAFCS-IN, and EAFCS-BN. EAFCS-EX selects the second operand from the set of simple features; EAFCS-IN selects the second operand from the current individual; and EAFCS-BN uses the relational information about the features presented in the structure of the Bayesian network for selecting the second operand. I analyzed how my three mutation schemes and the crossover operator control the individual's search space and their impact on the performance of the EA. Using a naïve Bayes classifier both for fitness evaluation and testing, I found that the constructed features from the EA can significantly enhance the accuracy of the naïve Bayes classifier using a smaller number of features. More importantly, my EA can find interesting relationships between the simple features, and thus serve as a knowledge discovery process.

Future work includes testing my gesture recognition system with different users. The performance of the event-based and the time-based segmentation for local feature extraction could be different depending on a user. If the user moves her/his hand at a constant speed then there might be no significant difference between the two local segmentation methods. Also, there might be common features that are important across different users, and tailored features that are particularly useful for a specific user. To apply EAFCS to learn features tailored to a particular user in real-world applications, the EAFCS should be modified so that useful features can be discovered in real-time. One possibility for finding tailored features in real-time might be performing an offline search first aimed at finding common features using data collected from different users, and next quickly tailoring common features online to reflect the gesture characteristics of the current user's behavior.

This research also needs to be extended to include more gestures such as going up or down, which might be frequently used in the personal mobile devices. Increasing the number of classes and instances will make the computation of EAFCS more expensive. Thus, we will need more efficient way of evaluating fitness of a candidate solution. For example, instead of using the whole data, only representative examples sampled from each class might be used for fitness evaluation.

In addition, increasing the number of classes and instances may require more number of simple features. Since the population of an EAFCS is initialized by selecting a simple feature for an individual, a large population size will be required when the number of features increases. One possibility to tackle this problem might be to use random restart mechanism so that the performance of EA is less sensitive to the randomly selected initial population [84]. In the general random restart mechanism, the population is reinitialized when the EA converges to local optima. The new population keeps the best-so-far solution found in the previous cycle. However, since my EA evaluates an individual at the level of a feature set not at the level of a feature, keeping the best-so-far solution may cause the loss of important features that are not included in the best-so-far solution. A better way to generate a new population might be to learn the usefulness of features while the EA evolves, and use the information for random restart. In this approach, evaluating the usefulness of each feature and generating promising individuals from the set of good features for each class would be key performance improvement considerations.

Finally, I expect that the speed-sensitive finger gesture recognition system can potentially provide useful real-world application to personal mobile devices especially in situations where a user cannot actually touch the screen. Speed-sensitive gestures may also provide more semantics, such as flipping pages, moving quickly, or speeding up music, to the limited number of finger gestures.

Bibliography

[1] T. Delbruck, "Frame-free dynamic digital vision," presented at the International Symposium on Secure-Life Electronics, Tokyo, Japan, 2008.

[2] L. Hyeon-Kyu and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 21, pp. 961-973, 1999.

[3] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor," *Solid-State Circuits, IEEE Journal of,* vol. 43, pp. 566-576, 2008.

[4] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 37, pp. 311-324, 2007.

[5] C. Wah Ng and S. Ranganath, "Real-time gesture recognition system and application," *Image and Vision Computing,* vol. 20, pp. 993-1007, 2002.

[6] C. Qing, N. D. Georganas, and E. M. Petriu, "Real-time Vision-based Hand Gesture Recognition Using Haar-like Features," in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE,* 2007, pp. 1-6.

[7] P. Garg, N. Aggarwal, and S. Sofat, "Vision Based Hand Gesture Recognition," *Engineering and Technology,* vol. 49, pp. 972-977, 2009.

[8] X. Yishen, G. Jihua, T. Zhi, and W. Di, "Bare Hand Gesture Recognition with a Single Color Camera," in *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, 2009, pp. 1-4.

[9] A. Licsar and T. Sziranyi, "Supervised training based hand gesture recognition system," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2002, pp. 999-1002 vol.3.

[10] J.-h. An and K.-S. Hong, "Finger gesture-based mobile user interface using a rear-facing camera," in *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, 2011, pp. 303-304.

[11] K. Tsukada and M. Yasumura, "Ubi-Finger: Gesture input device for mobile use," presented at the Proc. of 5th Asia Pacific Conference on Computer Human Interaction, 2002.

[12] K. Morimoto, C. Miyajima, N. Kitaoka, K. Itou, and K. Takeda, "Statistical segmentation and recognition of fingertip trajectories for a gesture interface," presented at the Proceedings of the 9th international conference on Multimodal interfaces, Nagoya, Aichi, Japan, 2007.

[13] H. Morrison and S. J., "Contact-free recognition of user-defined gestures as a means of computer access for the physically disabled," in *Proc. 1st Cambridge Workshop on Universal Access and Assistive Technology*, 2002.

[14] N. Henze, A. Löcken, S. Boll, T. Hesselmann, and M. Pielot, "Free-hand gestures for music playback: deriving gestures with a user-centred process," presented at the Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia, Limassol, Cyprus, 2010.

[15] T. Shanableh, K. Assaleh, and M. Al-Rousan, "Spatio-Temporal Feature-Extraction Techniques for Isolated Gesture Recognition in Arabic Sign Language," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, pp. 641-650, 2007.

[16] N. Liu, B. C. Lovell, P. J. Kootsookos, and R. I. A. Davis, "Model structure selection & training algorithms for an HMM gesture recognition system," presented at the Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on, 2004.

[17] B.-H. Oh, J.-H. An, and K.-S. Hon, "Mobile User Interface Using a Robust Fingertip Detection Algorithm for Complex Lighting and Background Conditions " presented at the 2012 International Conference on Information and Computer Networks (ICICN 2012), Singapore, 2012.

[18] H. Kang, C. W. Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recogn. Lett.*, vol. 25, pp. 1701-1714, 2004.

[19] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257-286.

[20] L. Welch, "Hidden Markov Models and the Baum-Welch Algorithm," *IEEE Information Theory Society Newsletter*, vol. 53, 2003.

[21] E. Sánchez-nielsen, L. Antón-canalís, and M. Hernández-tejera, "Hand Gesture Recognition for Human-Machine Interaction," *WSCG*, vol. 12, 2004.

[22] J. Alon, V. Athitsos, Y. Quan, and S. Sclaroff, "A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1685-1699, 2009.

[23] F. Morchen and A. Ultsch, "Optimizing time series discretization for knowledge discovery," presented at the Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, Chicago, Illinois, USA, 2005.

[24] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," in *Data Mining In Time Series Databases*. vol. 57, M. Last, A. Kandel, and H. Bunke, Eds., ed: World Scientific Publishing Company, 2004, pp. 1-22.

[25] H. Liu and H. Motoda, *Computational Methods of Feature Selection*: Chapman & Hall/CRC, 2008.

[26] Y. Zhang and P. I. Rockett, "Domain-independent feature extraction for multi-classification using multi-objective genetic programming," *Pattern Anal. Appl.*, vol. 13, pp. 273-288, 2010.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*: Springer-Verlog, 2009.

[28] J. Shlens, *A tutorial on Principal Component Analysis*, 2005.

[29] I. William F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. D. Hovland, and R. J. Enbody, "Further Research on Feature Selection and Classification Using Genetic Algorithms," presented at the Proceedings of the 5th International Conference on Genetic Algorithms, 1993.

[30] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, pp. 273-324, 1997.

[31] D. V. Sridhar, E. B. Bartlett, and R. C. Seagrave, "Information theoretic subset selection for neural network models," *Computers & Chemical Engineering*, vol. 22, pp. 613-626, 1998.

[32] M. A. Hall, "Correlation-based Feature Selection for Machine Learning," PhD, Waikato University, 1998.

[33] Q. Yang, E. Salehi, and R. Gras, "Using feature selection approaches to find the dependent features," presented at the Proceedings of the 10th international conference on Artificial intelligence and soft computing: Part I, Zakopane, Poland, 2010.

[34] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data : A Fast Correlation-Based Filter Solution," presented at the the Twentieth International Conference on Machine Learning, Washington D.C., 2003.

[35] Ö. Uncu and I. B. Türkşen, "A novel feature selection approach: Combining feature wrappers and filters," *Information Sciences*, vol. 177, pp. 449-466, 2007.

[36] O. Ritthoff, R. Klinkenberg, S. Fischer, and I. Mierswa, "A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm," in *the 2002 U.K. Workshop on Computational Intelligence*, 2002, pp. 147-154.

[37] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, "Hybrid feature selection by combining filters and wrappers," *Expert Syst. Appl.*, vol. 38, pp. 8144-8150, 2011.

[38] R. K. Agrawal and R. Bala, "A Hybrid Approach for Selection of Relevant Features for Microarray Datasets," in *World Academy of Science, Engineering and Technology*, 2007.

[39] B. Boashash, *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*: Oxford: Elsevier Science, 2003.

[40] Y. Jiang, B. Cukic, and T. Menzies, "Can data transformation help in the detection of fault-prone modules?," presented at the Proceedings of the 2008 workshop on Defects in large software systems, Seattle, Washington, 2008.

[41] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Classifier design with feature selection and feature extraction using layered genetic programming," *Expert Syst. Appl.*, vol. 34, pp. 1384-1393, 2008.

[42] O. Oechsle and A. F. Clark, "Feature extraction and classification by genetic programming," presented at the Proceedings of the 6th international conference on Computer vision systems, Santorini, Greece, 2008.

[43] Y. Zhang and P. I. Rockett, "A generic multi-dimensional feature extraction method using multiobjective genetic programming," *Evol. Comput.*, vol. 17, pp. 89-115, 2009.

[44] H. F. Gray, R. J. Maxwell, I. Martìnex-Pérez, and a. S. C. C. Arús, "Genetic programming for classification and feature selection: analysis of 1H nuclear magnetic resonance spectra from human brain tumor biopsies," *NMR in Biomedicine*, vol. 11, pp. 217-224, 1998.

[45] M. Smith and L. Bull, "Genetic Programming with a Genetic Algorithm for Feature Construction and Selection," *Genetic Programming and Evolvable Machines*, vol. 6, pp. 265-281, 2005.

[46] K. Krawiec, "Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks," *Genetic Programming and Evolvable Machines*, vol. 3, pp. 329-343, 2002.

[47] K. A. DeJong, *Evolutionary Computation: a unified approach*: MIT Press, 2006.

[48] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK: IOP Publishing Ltd., 1997.

[49] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 4, pp. 164-171, 2000.

[50] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," *Intelligent Systems and their Applications, IEEE*, vol. 13, pp. 44-49, 1998.

[51] H. Vafaie and K. De Jong, "Genetic algorithms as a tool for feature selection in machine learning," in *Tools with Artificial Intelligence, 1992. TAI '92, Proceedings., Fourth International Conference on*, 1992, pp. 200-203.

[52] K. Neshatian and M. Zhang, "Pareto front feature selection: using genetic programming to explore feature space," presented at the Proceedings of the 11th Annual conference on Genetic and evolutionary computation, Montreal, Québec, Canada, 2009.

[53] U. Kamath, K. A. D. Jong, and A. Shehu, "Selecting predictive features for recognition of hypersensitive sites of regulatory genomic sequences with an evolutionary algorithm," presented at the Proceedings of the 12th annual conference on Genetic and evolutionary computation, Portland, Oregon, USA, 2010.

[54] C.-F. Tsai, "Feature selection in bankruptcy prediction," *Know.-Based Syst.*, vol. 22, pp. 120-127, 2009.

[55] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, pp. 335-347, 1989.

[56] Z. Huang, M. Pei, E. Goodman, Y. Huang, and G. Li, "Genetic algorithm optimized feature transformation: a comparison with different classifiers," presented at the Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII, Chicago, IL, USA, 2003.

[57] Y. Zhang and P. I. Rockett, "Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection," presented at the Proceedings of the 2005 conference on Genetic and evolutionary computation, Washington DC, USA, 2005.

[58] P.-F. Guo, P. Bhattacharya, and N. Kharma, "An efficient image pattern recognition system using an evolutionary search strategy," presented at the

Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 2009.

[59] J. R. Koza and R. Poli, "Genetic Programming," in *Search Methodologies:Introductory Tutorials in Optimization and Decision Support Techniques*, Springer, Ed., ed, 2005, pp. 127-164.

[60] R. Poli, "Genetic programming for image analysis," presented at the Proceedings of the First Annual Conference on Genetic Programming, Stanford, California, 1996.

[61] T. Mitchell, *Machine Learning*: Mc Graw-Hill, 1997.

[62] D. Robilliard and C. Fonlupt, "Backwarding: An Overfitting Control for Genetic Programming in a Remote Sensing Application." vol. 2310, P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, Eds., ed: Springer Berlin / Heidelberg, 2002, pp. 57-74.

[63] G. Paris, D. Robilliard, and C. Fonlupt, "Exploring Overfitting in Genetic Programming," in *Evolution Artificielle 6th International Conference*, 2004, pp. 267-277.

[64] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's Razor," *Information Processing Letters*, vol. 24, pp. 377-380, 1987.

[65] S. Silva and L. Vanneschi, "Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction," presented at the Proceedings of the 11th Annual conference on Genetic and evolutionary computation, Montreal, Quebec, Canada, 2009.

[66] K. Strimmer, "Statistical Thinking: Introduction to Probabilistic Data Analysis," 2010.

[67] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Mach. Learn.*, vol. 29, pp. 131-163, 1997.

[68] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, pp. 677-695, 1997.

[69] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10-18, 2009.

[70] M. Gillies. (2011). *HMMWeka Package*. Available: http://www.doc.gold.ac.uk/~mas02mg/software/hmmweka/index.html

[71] E. Ogasawara, L. C. Martinez, D. de Oliveira, Zimbra, x, G. o, G. L. Pappa, and M. Mattoso, "Adaptive Normalization: A novel data normalization approach for non-stationary time series," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 2010, pp. 1-8.

[72] K. L. Olli Viikki, "Noise Robust HMM-Based Speech Recognition Using Segmental Cepstral Feature Vector Normalization," presented at the Robust Speech Recognition for Unknown Communication Channels, 1997.

[73] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems)*: Morgan Kaufmann, 2011.

[74] H. Jin and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 17, pp. 299-310, 2005.

[75] D. J. Hand and R. J. Till, "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems," *Mach. Learn.,* vol. 45, pp. 171-186, 2001.

[76] W. Lam and F. Bacchus, "Learning Bayesian belief networks: An approach based on the MDL principle," *Computational Intelligence,* vol. 10, pp. 269-293, 1994.

[77] I. Tsamardinos and C. Aliferis, "Towards Principled Feature Selection: Relevancy, Filters and Wrappers," in *in Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[78] Z. Yifeng, L. Jian, and L. Shuyuan, "Classification using Markov blanket for feature selection," in *Granular Computing, 2009, GRC '09. IEEE International Conference on*, 2009, pp. 743-747.

[79] S. Yaramakala and D. Margaritis, "Speculative Markov blanket discovery for optimal feature selection," in *Data Mining, Fifth IEEE International Conference on*, 2005, p. 4 pp.

[80] G. Bontempi and P. E. Meyer, "Causal filter selection in microarray data," in *the 27th International Conference on Machine Learning*, Haifa, Israel, 2010, pp. 95-102.

[81] R. Feldt and P. Nordin, "Using Factorial Experiments to Evaluate the Effect of Genetic Programming Parameters," presented at the Proceedings of the European Conference on Genetic Programming, 2000.

[82] S. Luke, G. C. Balan, and L. Panait, "Population implosion in genetic programming," presented at the Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII, Chicago, IL, USA, 2003.

[83] P. Reed, B. Minsker, and D. E. Goldberg, "Designing a competent simple genetic algorithm for search and optimization," *Water Resour. Res.*, vol. 36, pp. 3757-3761, 2000.

[84] G. T. Pulido and C. A. C. Coello, "The micro genetic algorithm 2: towards online adaptation in evolutionary multiobjective optimization," presented at the Proceedings of the 2nd international conference on Evolutionary multi-criterion optimization, Faro, Portugal, 2003.

VITA

Eun Yeong Ahn

EDUCATION:

- Doctor of Philosophy in Information Sciences and Technology (Aug 2012)
  The Pennsylvania State University, University Park, PA
  Minor in Computational Science
  Advisor: Dr. John Yen
- M.S. in Computer Science and Engineering in Pusan National University, Republic of Korea (Feb 2008)
- Bachelor of Engineering in Computer Science and Engineering in Pusan National University, Republic of Korea (Feb 2006)

PROFESSIONAL EXPERIENCE:

- Software Engineering Intern, Google Korea, Seoul, South Korea (Summer 2011)
- Research Intern, Samsung Advanced Institute of Technology (SAIT), South Korea (Summer 2010)
- Reviewer for the IEEE Congress on Evolutionary Computation (2012) and AMEC (2010).

TEACHING EXPERIENCE:

Teaching Assistants for

- Network Management and Security, IST 511 (Spring 2012)
- Complex Software Systems, IST 412 (Fall 2011)
- IST Integration, IST 440W (Spring 2011)
- Language, Logic, and Discrete Mathematics (Spring, Fall 2010)
- Data Mining, IST557/STAT557 (Fall 2009)

PUBLICATIONS:

- E.Y. Ahn, J. Yen, and T. Mullen, Gesture-Inspired Motion Segmentation, *submitted to IEEE Transactions on Systems, Man and Cybernetics (SMC)*.
- E.Y. Ahn, T. Mullen, and J. Yen, A Two-Population Evolutionary Algorithm for Feature Extraction: Combining Filter and Wrapper, *IEEE Congress on Evolutionary Computation*, New Orleans, USA, Jun 2011, pp. 736–743.
- E.Y. Ahn, T. Mullen, and J. Yen, Evolutionary Based Feature Extraction with Dynamic Mutation, *IEEE Congress on Evolutionary Computation*, New Orleans, USA, Jun 2011, pp. 409–416.
- E.Y. Ahn, J. H. Lee, T. Mullen, and J. Yen, Dynamic Vision Sensor Camera Based Bare Hand Gesture Recognition, *in Proc. of IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing*, 2011, pp.52–59.
- E.Y. Ahn, T. Mullen, and J. Yen, Finding Feature Transformations Functions using Genetic Algorithm, *in Proc. of the 12th annual conference companion on Genetic evolutionary computation (GECCO '10)*, pp. 2061–2062.