Evolving User-Specific Emotion Recognition Model via Incremental Genetic Programming

November, 2016

Rahadian Yusuf

Graduate School of Science and Engineering

Doshisha University

Acknowledgements

This dissertation can be completed because of the support of many people in my doctoral program. I would never be able to perform my research, nor to complete this dissertation without their help. Therefore, I would like to express my deepest gratitude to the people who keeps on helping me in the whole research and writing.

First of all, I would like to devote my greatest gratitude to Professor Katsunori Shimohara. Not only he accepted me into the doctoral program of Doshisha University, he gave me a chance to perform a research of my own selection. In addition, he also patiently nurtured and supported me on every occasion throughout the whole program. As an advisor, he provided helps and suggestions in research and publications, in my effort to earn doctoral degree, as well as academic life in Japan.

I would also like to thank Professor Ivan Tanev. He provided a lot of support, from the XGP to many elegant revisions on my papers and articles, to many discussions about research progresses, as well as whenever I am stuck with difficult questions.

I would like to express my gratitude to my thesis reviewer, Professor Masashi Okubo, who kindly suggested a lot of good and important matters in revising my dissertation to a better and clearer thesis.

Doshisha University, and also Yoneyama Memorial Foundation, both provided scholarship so I can focus more to my research. Without them, I would be forced to take a lot of work in order to support my education financially.

I would like to thank Dipak-san, who has been my research-buddy and conference-pal, as well as co-author partner during my doctoral course. I also would like to thank Kimura-san, Kimoto-san, Takahara-san, and everyone from socioinformatics laboratory, who have been lab-partners and have been a great help in my research.

Last but not least, I would like to thank my father, my mother, my sister, and my partner Bena, for keep on believing and supporting me in this venture.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Affective Computing and Emotion Recognition

1.1.1 Concept of Affective Computing

A computer agent is software to help a user in operating computers. It works by improving the communications between a computer and the user, thus reducing the difficulties in using computers. It is common knowledge that computers and humans are different, and it is also common knowledge that not every human is capable or comfortable in operating computers.

There is an approach in a user interface that is called Affective computing [1]. Affective computing implements emotions into human computer interactions, whether in the form of emotion recognition or in the form of emotion expressions. It is also said that improving the emotion recognition will also improve the agent's accuracy in predicting user's intention.

For example, a user might tell a computer to open the e-mail and open an empty spreadsheet. Based on the emphasis on user's voice or accent, as well as by user's gestures, the computer can determine whether the user wants the e-mail primarily or the spreadsheet primarily, or want to compare both of them. Without emotion recognition capability, without affective computing, the computer would only assume the task as the plain command demanded.

This is just one example of the use of affective computing. Researches can be done to explore fully about interactions between humans and computers, and it might also contributes in another branch of science. Similar situation occurred between psychology and brain science. It has been a long time since emotion is said to be irrelevant to the brain, cognition, and thinking. However, the brain is also the organ that processes emotion, along with thinking and cognition. It is then safe to assume that by learning about emotion, we can also learn about brain.

There are also several applications that can reap benefits from intelligent agent capable of emotion. The elderly can receive helps in operating the computer, and the computer with affective computing will be able to understand the elderly’s intention, as well as expressing proper words to support them in learning to operate the computer.

Another application that can benefit from intelligent agent capable of emotion is a mediator agent. In the case of communications between people from different culture, there are high chances of misunderstanding. The agent can be used as a mediator (since the agent will have known about its user’s unique features) to avoid misunderstandings because of different way in expression or gestures. Both agents might be able to communicate between themselves, or even might provide direct translations between the two people.

Because of these reasons and dreams, we could summarize that the research about intelligent agent capable of emotions is necessary.

1.1.2 Several Emotions Recognition Models

The field of emotion recognition is related to affective computing, for which emotion recognition systems have been developed. Most of these studies have implemented similar methods such as using still-image data, generalization of human expressions, or the non-pervasive sensors such as electrocardiography (ECG) or electroencephalography (EEG) [2]. Meanwhile, there are several gestures that usually relate to human emotion, such as nodding or head movements, and the stillness or the rapid movements of a whole body might relate to anticipation or anxiety. Still images have failed to capture these gestures, since temporal information is not stored in a single image. One of the scientists who use temporal information is Alex Pentland. He stated that the activity level of persons is related to their emotions [3].

Using non-pervasive sensors also might make a user uncomfortable and may influence the user's actual emotion. In addition to this issue, a non-pervasive sensor is often not practical for real-world applications due to the special setup of the environment that might be needed.

A survey paper on affect recognition models [4] covered much recent research on affect and emotion recognition methods. It stated that there are no available benchmark data to compare the various methods fairly and also surveyed many researches that focus on emotion recognition based on photographs of actors or detecting images of a video by focusing on images in each frame. Further, it also stated that, in many of the studies, the available training data often consisted of actors making exaggerated expressions.

Another book proposes the paradigm that there are several matters in the world that cannot be averaged, and that individuals are unique. Thus when we design anything to fit the average data, then it will fit into no one, since no one is average [5].

It is then a common sense that individuals might express emotions quite differently too, and designing an emotion recognition model based on generic data might be not a good idea.

1.1.3 Limitations and Challenges

To sum up, there are several limitations and challenges on emotion recognition model.

Temporal Information

Current researches that analyze facial expressions tend to rely on still images. The researches sometime able to analyze facial expressions of a video but relying on frame per frame analysis, thus the temporal information (such as relative movements and gestures) might have been lost.

Pervasiveness

There are several researches that analyze the temporal information, however they use special sensors that are not pervasive, or using an environment that are not pervasive. Many of the researches are performed on a special setting of environment, or attaching special sensors to the subjects.

General and Not Unique

There have been several issues with *benchmark data* for emotion recognition, even if they are available [6] [4]. First, the data gathered from subjects who specifically requested to express specific emotion, one at a time. Therefore, they tend to be exaggerated, and the subjects might be actors tasked to express emotions. Second, there have not been a good benchmark data that have good temporal information and being natural as well. Third, the benchmark data relies on many people to be averaged, and each has only few data; thus it is usable only to recognize a set of emotion of people, to be tested on a set of emotion of people. From the paradigm mentioned at Rose' book [5], it is possibly not a good idea to use average information, and it is better to make a customized, user-specific approach instead.

Learning / Training

There have been several methods in artificial intelligence to develop a classifier, such as neural networks or support vector machine. However, they are similar to a black box, and analyzing the functions might be relatively difficult, if not impossible.

1.2 Motivations of Research

Based on the limitations and challenges of current research, we would like to approach from another direction and propose a model of a user-specific (not average) emotion recognition that is relying on pervasive sensors, incorporating temporal information as important feature, with classifiers or recognition programs can be analyzed by third party such as the user or psychologist.

1.3 Objectives of Research

We aim to design an emotion recognition model that is: (1) incorporating temporal information as an important feature for emotion analysis; (2) using pervasive and ubiquitous sensor, not a specific or non-pervasive sensors, on a pervasive environment; (3) avoid using many people data but instead focuses on a unique user specific features through repeated interactions, therefore the model should be able to implement new user's data; (4) with the programming code that can be analyzed if necessary, rather than a black-box system.

1.4 Thesis Overview

Emotion, Tree Structure, XGP, and Proposed Model (Chapter 3)

In this thesis, the proposed model of user-specific emotion recognition is presented. The environment for experiments and data acquisition, as well as preliminary results and preparations for simulated evolution are elaborated.

Enhancing the Model: Implementation of Voting (Chapter 4)

This thesis presents an approach to enhance the model, by implementing collaborative filtering in the form of majority voting.

Exploring Evolutionary Computation: Adaptive Voting (Chapter 5)

This thesis aims to explore evolutionary computation to see whether adaptive voting of weighted trust can emerge naturally, by implementing genetic algorithm, in an attempt to facilitate the required ability of the model to accept new data of users.

Exploring Evolutionary Computation: Incremental Genetic Programming (Chapter 6)

This thesis aims to explore evolutionary computation towards another direction, which is by implementing incremental genetic programming. The incremental genetic programming is an effort to enhance the result from adaptive voting, with the same goals.

Miscellaneous Emotion Analysis (Chapter 7)

This thesis aims to analyze several emotions using the model proposed, such as the optimal timing of acquiring the data or which emotion is the easiest to recognize. We also show that the proposed model is not a black-box model, that the programs evolved by XGP can be analyzed and comprehended.

1.5 Thesis Organization

This thesis is organized as follows: In Chapter 2, the background of the research is described. Chapter 3 provides explanations about our initial experiments and environments, as well as our proposed model. Enhancing the model using collaborative filtering is proposed at Chapter 4. Explorations on evolutionary computation are performed and then reported at Chapter 5, about the implementation of adaptive voters, and at Chapter 6, about the use of incremental genetic programming. Miscellaneous analysis on emotions using our proposed model is presented at Chapter 7. Finally, Chapter 8 concludes this thesis.

Chapter 2

Research Framework for Emotion Recognition

This chapter presents the concepts and definitions that lay the foundation for this thesis. First, a brief description on fundamental knowledge for the current research is introduced. Then, the evolutionary computation and the system used on this research are presented.

2.1 Affective Computing

Affective computing is the study and development of systems and devices that can recognize, interpret, process, and simulate human affects. It is an interdisciplinary field spanning computer science, psychology, and cognitive science [7].

An important aspect in affective computing [1] is emotion recognition for a user. One of the implementations of affective computing consists of an affective agent that is not only logical but also compassionate; this agent can help the user better, based on the current emotion.

One of the main areas of affective computing is recognition of user's emotion. There have been many methods with different results, but many of them are not natural enough, thus might not be usable for real life applications.

2.2 Honest Signals

Although many of the research on emotion recognition that uses facial expression relies on still images, or the peak of facial expressions only, there are evidences that some signals are more important at recognizing user's emotion [3], such as the variation of activity or repetitive motions.

Pentland in his book showed that variation of activity correlates to human's behavior. In example, nervous people would either suddenly become still and not moving, or they might also move too much. So it can be said that a change of the variations (either suddenly dropping or increasing) might means that there is a change of emotional state.

2.3 Facial Action Coding System

Facial Action Coding System (FACS) is a system to taxonomize human facial movements by their appearance on the face, based on a system originally developed by a Swedish anatomist Carl-Herman Hjortsjö. It was later adopted by Paul Ekman and Wallace V. Friesen [8].

The system categorized facial muscles movements (contraction or relaxation) into Action Units (AU), and by analyzing the AU, an analysis to guess the emotion expressed can be performed.

One of the face model that implemented FACS is the CANDIDE [9]. A face model of CANDIDE is shown on Fig. 1 below.

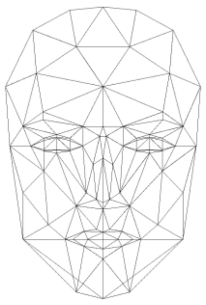


Figure 1. CANDIDE-3 Face Model

2.3 The End of Average: General and Uniqueness

The End of Average [5] proposed a paradigm where the idea of average actually never fits any. An excerpt from the book is shown below:

"In 1950, researchers at Wright Air Force Base in Ohio measured more than 4,000 pilots on 140 dimensions of size, including thumb length, crotch height, and the distance from a pilot's eye to his ear, and then calculated the average for each of these dimensions.

Everyone believed this improved calculation of the average pilot would lead to a better-fitting cockpit and reduce the number of crashes — or almost everyone.

Out of 4,063 pilots, not a single airman fit within the average range on all ten dimensions (height, shoulders, chest, waist, hips, legs, reach, torso, neck, thigh).

One pilot might have a longer-than-average arm length, but a shorter-than-average leg length. Another pilot might have a big chest but small hips. If you picked out just three of the ten dimensions of size--- say, neck circumference, thigh circumference, and wrist circumference---less than 3.5 percent of pilots would be averaged sized on all three dimensions.

There was no such thing as an average pilot."

The paradigm suggested that, for a complex and multi-dimensional system such as human, it is not safe to simply use data of many people to calculate a fit-for-all method.

2.4 Emotion Classifications

Emotions emerged in complicated species to meet the need for high degrees of response flexibility to the often complex and subtle conditions of life that could generate harms and benefits [10]. How a given individual reacts emotionally to an encounter depends on an evaluation of what the encounter implies for that individual [11].

Another major theory of emotions is Component Process Model that regards emotion as the synchronization of many different cognitive and physiological components. Emotions are identified with the overall process where low level cognitive appraisals trigger bodily reactions [12].

Other than behavioral psychology, there is also evolutionary psychology. Evolutionary psychology (EP) is a theoretical approach in the social and natural sciences that examines psychological structure from a modern evolutionary perspective. It seeks to identify which human psychological traits are evolved adaptations – that is, the functional products of natural selection or sexual selection in human evolution. Adaptationist thinking about physiological mechanisms, such as the heart, lungs, and immune system, is common in evolutionary biology. Some evolutionary psychologists apply the same thinking to psychology, arguing that the modularity of mind is similar to that of the body and with different modular adaptations serving different functions. Evolutionary psychologists argue that much of human behavior is the output of psychological adaptations that evolved to solve recurrent problems in human ancestral environment [13].

There are many emotion classifications, and psychologists often propose different classifications. It is a contested issue in emotion research and affective science. Several examples of emotion classifications are described at subsequent subchapters.

2.4.1 Circumplex Model

This model is developed by James Russel, and shows a circular model of two axes: valence and arousal. The valence axis represents the pleasantness of emotion, while the arousal axis represents the arousal level of emotion [14]. Fig. 2 below illustrates this model.

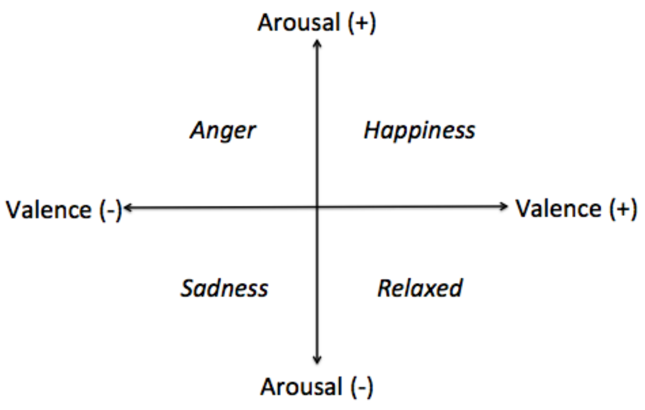


Figure 2. Illustration of Circumplex model

2.4.2 Plutchik's Model

Robert Plutchik arranges emotions in concentric circles where inner circles are more basic and outer circles more complex. Illustration of Plutchik's Model can be found at Fig. 3 below.

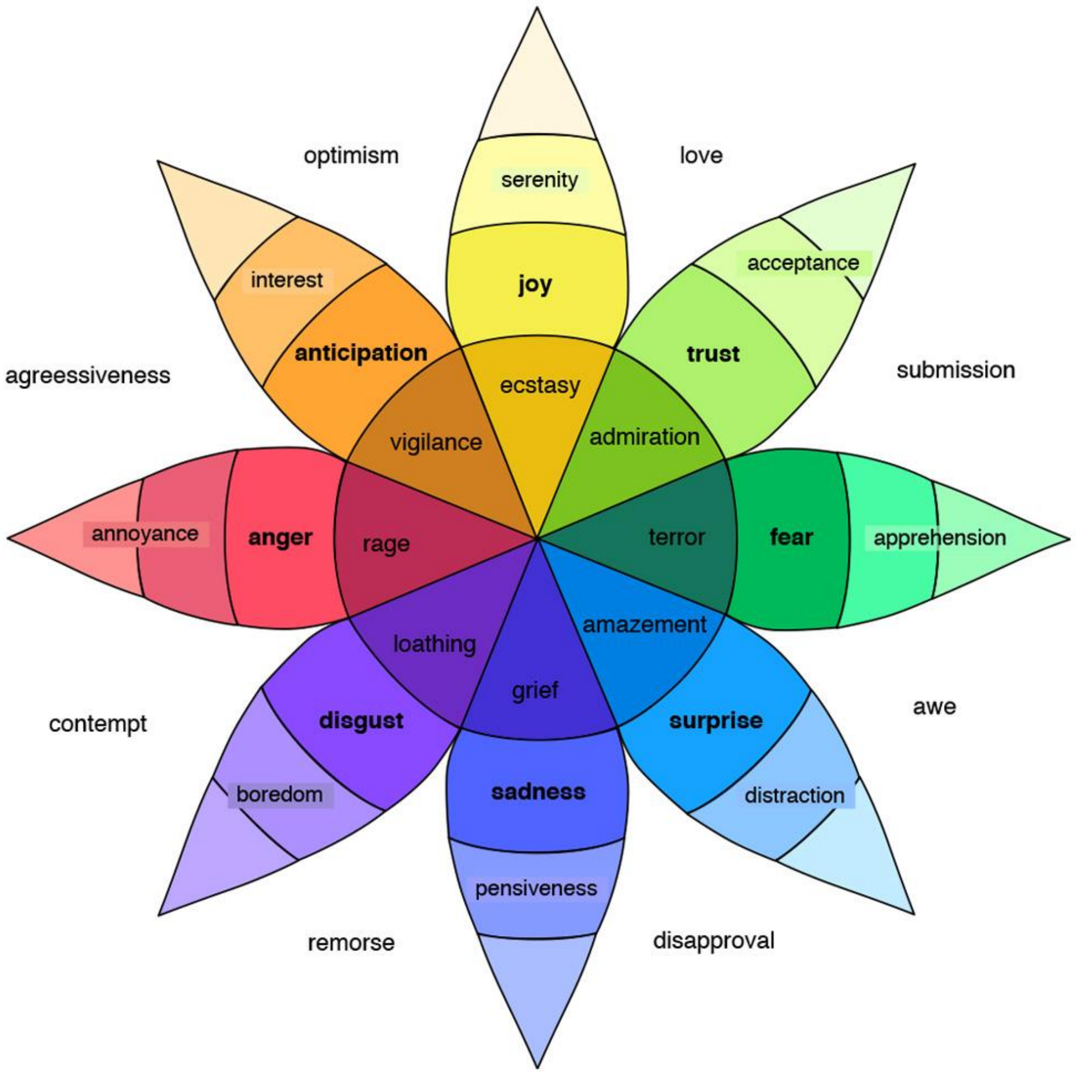


Figure 3. Plutchik Model (or Plutchik's Wheel)

2.4.3 Lövheim's Cube

Lövheim proposed a direct relation between specific combinations of the levels of the signal substances dopamine, noradrenaline and serotonin and eight basic emotions. For example, according to the model, anger is produced by the combination of low serotonin, high dopamine and high noradrenaline. Lövheim wrote that as neither the serotonin nor the dopamine axis is identical to the "pleasantness" (i.e. valence) dimension in earlier theories, the cube seems somewhat rotated when compared to these models [15].

The cube is illustrated at Fig. 4 below.

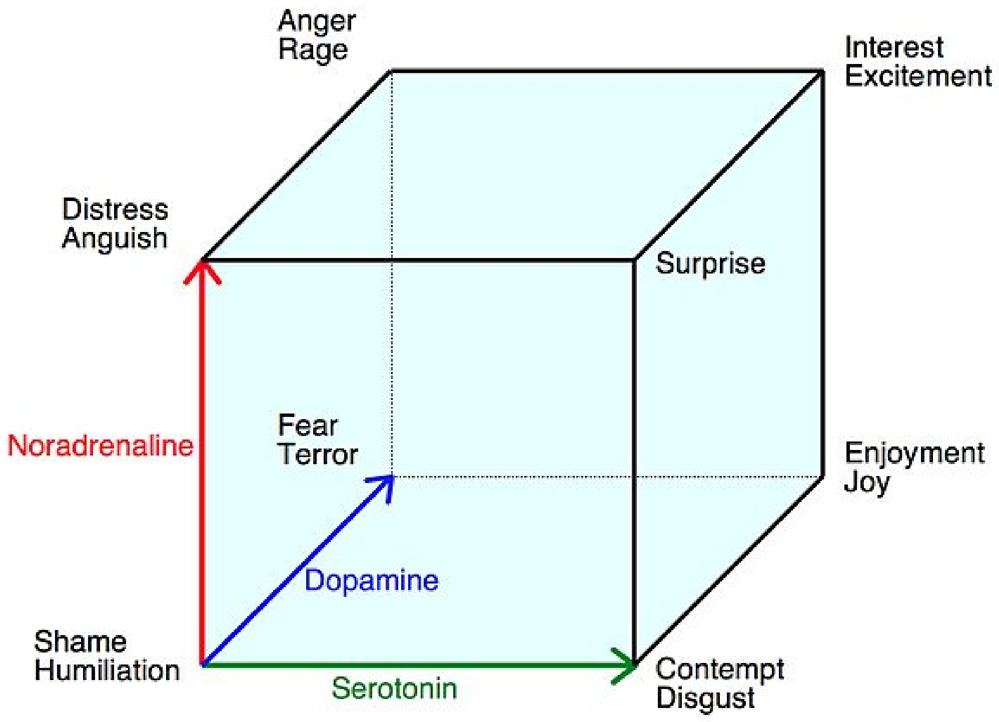


Figure 4. Lovheim's Cube

2.5 Evolutionary Computation

Evolutionary computation is a biologically inspired algorithmic paradigm that uses the principle of evolution in Nature, especially, the survival of fittest. As a nature-inspired algorithm, there are several terminologies borrowed from biological world, such as selection, individual, mutation, crossover and fitness.

Originally, evolutionary computation consisted of three evolution-based paradigms: evolution strategies [16] that focus on building systems capable of solving real-valued parameter optimization problems, evolutionary programming [17] on which early work centered on automatization of evolving finite state machines capable of responding to environmental stimuli, and genetic algorithm [18] was originally applied to combinatorial optimization.

Other machine learning research like neural network have issues such as it is almost impossible for human to reverse engineer the system. Other than that, the processing power needed to effectively handle large neural networks is too high [19].

The general idea of evolutionary computation is in generating a set of individual then checking their fitness to the environment. The good ones survive to the next generation, and might have new offspring from performing genetic crossover. As the evolutions repeat by itself, the fit individuals will survive, and the better genes of the individuals to fit the environment will survive.

In the terms of algorithm, the individual could represent a program or a function, determined by a set of terminals that represent genes. The environment and the fitness can represent a problem and the solution; the fit individuals to the environment means a better program or function to solve the problem towards the solution.

To simulate the process of evolution and birthing new generations and offspring, a set of operators are applied to surviving individuals. There are three basic operator groups: selection, crossover, and mutation.

Selection

The selection operator chooses which individuals of the population at generation *t* will survive to the next generation. Common selection method is by evaluating their *fitness* to the environment (or problem, or target solution).

Crossover

The crossover operator generates new individual (*offspring*) by combining the *genes* or parameter of two or more parent (surviving individuals). There are many ways to perform this, e.g. by swapping some parts of the genotypes of both parents.

Mutation

The mutation operator is applied directly on an individual (at a gene or parameter), by performing pinpoint modifications (that can be random or regulated according to a rule). Values can be added or subtracted, or bits are flipped.

2.5.1 Genetic Programming (GP) Basic Frameworks

Genetic Programming (GP) [20] [21] is a domain-independent problem-solving algorithmic paradigm inspired by the natural evolution of species based on the survival and reproduction of the fittest. GP and natural inspired computing are successfully applied for delivering a human-competitive solutions of increasingly difficult problems in AI such as analog and digital circuits design, spatial and temporal information identification and prediction, machine learning, cyberterrorism prevention [22], financial [23], etc.

2.5.2 XGP Introduced

XGP [24] is an in-house GP engine that features XML (Extensible Markup Language)-based genotype representations of candidate solutions (genetic programs), XML-schema that determines the allowed syntax of the genotypes, and a UDP channel to communicate between a fitness evaluator and the XGP. XGP manages the population of genetic programs and performs the main genetic operations – selection, crossover, and mutation.

The notoriously long execution time of GP is caused by the fact that typically, a population of many (hundreds or thousands) potential solutions to the problem (individuals) trying to find the near-optimal solution to the problem in an enormous search space, evolve through many (hundreds or thousands) generations based on their simulated ability (fitness) to solve the given task over many (tens or hundreds) sampled environmental situations (fitness cases).

Inspired by flexibility and recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML), XGP uses an approach of representing genetic program as a DOM-parsing tree featuring corresponding flat XML text. XGP's approach implies performing genetic operations on DOM-parsing tree using off-the shelf, platform- and language neutral DOM-parsers, and using XML-text representation as a Web-compliant format, feasible for representation of genetic programs during their migration among the computational nodes in eventual distributed GP.

XGP gives an advantage in the form of a faster development time due the versatility of usages: it only took a short time to change the XML-schema and adapt the desired syntax of genotypes for a program.

Evolving programs using XGP requires the engine as a Microsoft Windows application running in parallel with another application to evaluate the fitness of each individual sent by XGP.

Chapter 3

Proposed Model with XGP

This chapter aims to investigate the classifications of emotions, and how to represent them in the form of tree structure. This tree structure is used by the XGP to evolve the program that will be used as classifiers. This chapter also proposes a user-specific emotion recognition model, using pervasive sensors, as well as casual (not over-specific) environmental setup. [25]

3.1 Emotion Classifications

In our research, we employ the circumplex model [14] that uses both Valence (pleasantness or general mood) and Arousal (which is similar to activity level) to represent emotions. Fig. 5 illustrates the classification of emotions based on Valence and Arousal of the circumplex model.

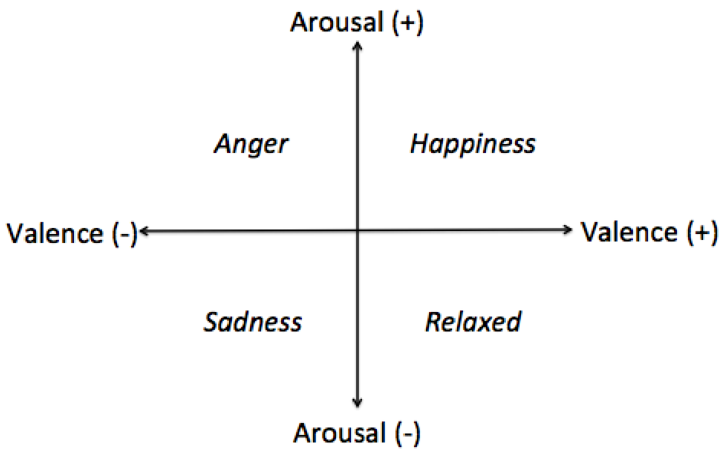


Figure 5. Circumplex mode showing Valence and Arousal

Compared to other models, the circumplex model is very basic and existing in other models, and should be representative enough of most basic emotions. Our research also avoids giving users difficult choices (e.g. choosing between feeling disgusted or annoyed), therefore we opted for an easier to understand emotions to be used. If we opted for another model, there is a high chance that the user would find difficulties in selecting their own current emotions, which might inhibit the natural interactions between the system and the user.

The four emotion categories that we use in our research can be described as follows:

- Happiness (positive valence, positive arousal),
- Relaxed (positive valence, negative arousal) ,
- Sadness (negative valence, negative arousal) , and
- Anger (negative valence, positive arousal)

3.1.1 Data to be used

Due to the lack of benchmark data that we can use in our approach and model [4], our research uses our own experimental data, while emphasizing the variety of such data (e.g., using different seating positions and lighting conditions to simulate real-world conditions, and not setting the environment for gaining specific kinds of data, and different subjects of varied demographics) in order to improve the validity of data and experiments, as well as to investigate the robustness of our system. However, our research is still limited by the minimum and maximum lighting conditions required by the sensor, as well as subject's subjectivity.

3.1.2 User's Features to be used

Our research uses the Facial Action Coding System [26] as representation of facial expression, selecting several action units (AU) from the system. To model the user's facial expression, we use the CANDIDE-3 face model [9], a 3D face model shown at Fig. 6 below.

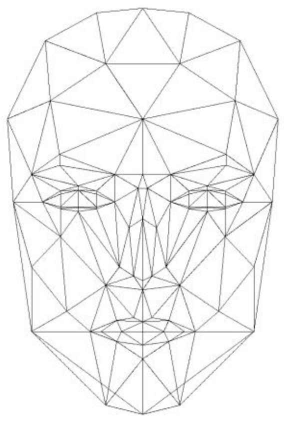


Figure 6. CANDIDE-3 Face Model

We selected several features from a user's facial expressions based on the CANDIDE-3 face model used by Microsoft Kinect. On the basis of this model, we selected nine Action Units (AU) that correspond to the detected facial movements:

1. upper lip raiser,
2. jaw lowerer,
3. lip stretcher,
4. brow lowerer,
5. lip corner depressor,
6. outer brow raiser,
7. head tilt pose: yaw,
8. head tilt pose: pitch, and
9. head tilt pose: roll.

The raw data is obtained from Kinect in the form of stream of video frames, which can be processed immediately to extract these nine features. The stream of raw data is fed into the system with a sampling period of 33ms.

Each AU is also normalized to a scale between -1,000 (no trace) and 1,000 (clear existence).

As the extracted AUs only represent facial expressions, further feature extraction is needed. Other research that recognize emotion using AU in real-time performs by analyzing each individual frame [27], thus removing the temporal information. To extract activity level and ***temporal information***, we need to know the changes of the AUs as well. Therefore, we put the extracted AUs into a buffer, and for every 100 rows (time-series data) of AUs, we extract three features for each AU:

- average value,
- standard deviation, and
- power

Average value and standard deviation represent the general level and activity level of each AU, respectively. An integration or power of the signals is used as well as one of the extracted features. The total number of extracted features is 27 (9x3) for every 100 frames of AU extracted from Microsoft Kinect. These features are coded as v_0 ~ v_26, where v_0 ~ v_8 represent the average of the 9 AUs in order above, v_9 ~ v_17 represent the standard deviation, and v_18 ~ v_26 represent the power.

3.2 XGP and Tree Structure

We use the in-house XGP engine explained at previous chapter to simulate evolutions for the experiments and in evolving classifiers for the emotion recognition. [25].

Since we use the circumplex model to represent emotions used in the research, we represent the emotion in the form of a tree structure, which are Valence and Arousal. This tree structure will be used by XGP to evolve program that will take user’s extracted features into account, along with several other functions that will be explained in the following subsections.

3.2.1 XGP

Here we will focus on evolution of the Emotion Recognition Module using XGP. The XGP is an in-house engine of Genetic Programming that uses XML-based genotypic representations of candidate solutions (genetic programs), XML-schema to determining the allowed syntax of the genotypes, and UDP channel to communicate between fitness evaluator and the XGP manager (which manages the population of genetic programs and performs the genetic operations – selection, crossover and mutation - on them) to perform evolution on the individuals. Fig. 7 below illustrates the configuration of the system that consists of XGP and the fitness evaluator module used for simulated evolutions.

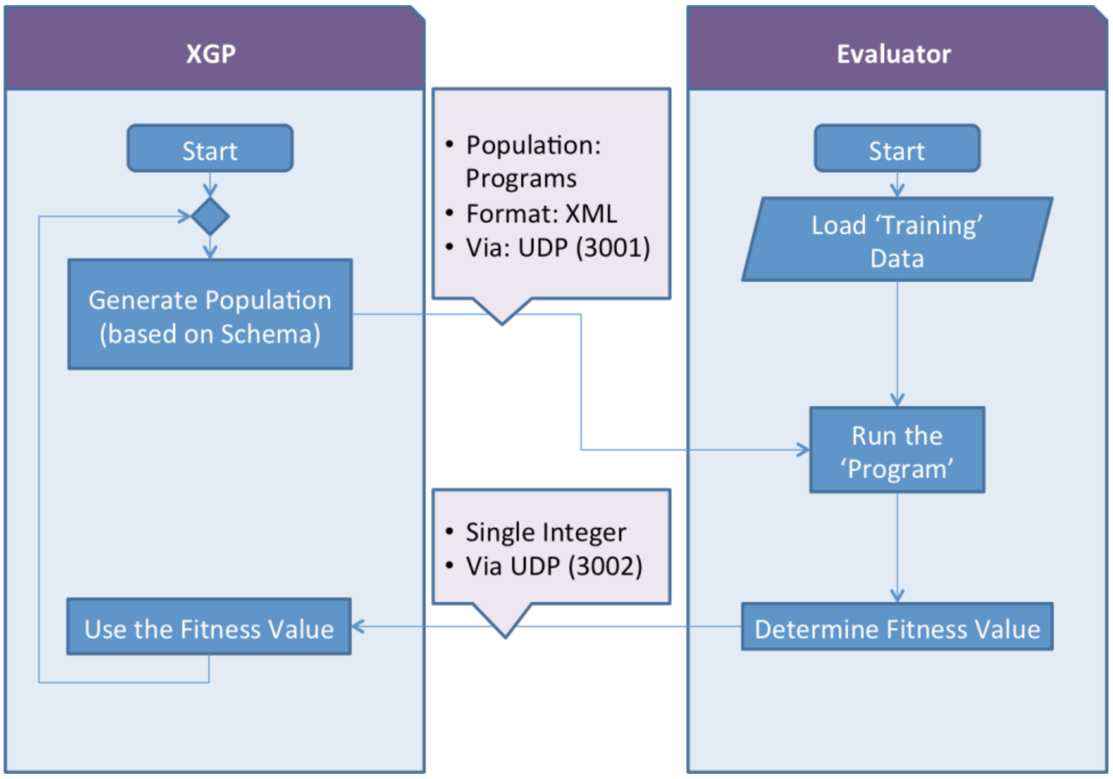


Figure 7. Integration of XGP (left) with the fitness evaluator (right) into a system that performs the simulated evolution of emotion recognition formulas.

3.2.2 Genes

There are 9 basic AU values gathered from Microsoft Kinect sensor, and each of them is normalized into a scale of -1000 to 1000. The system then extracts 3 values (average, standard deviation, power) from each AU for a specific time frame, which results into 27 extracted features. These features, together with random constant within the range (1..10) comprise the set of terminal symbols in XGP. The set of non-terminal (functional) symbols consist of arithmetical operations (+, -, *, /) and threshold evaluator at the root of tree-representation of candidate solution. This evaluator determines the condition of each Valence and Arousal of the inputs.

Table 1. XGP Features

Feature Name	*Values*
27 Extracted Features (terminal)	-1,000 – 1,000
Random Constants (terminal)	1 – 10
Arithmetical Operators (non-terminal)	+, - , * , /
Comparison Operators (non-terminal)	< , >

3.2.3 Fitness Function

In order to evaluate the fitness of an individual, a fitness function is needed. This research uses a variant of accuracy and precision: Matthew's Correlation Coefficient (MCC) [28]. It considers True Positive, True Negative, False Positive, and False Negative to represent the 'truth-ness' of a result.

We opted for this value instead of a simple accuracy calculation, because our system should to consider both precision and recall as a single value, and MCC offers this possibility.

MCC can be expressed as the following equation.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

where:

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

MCC results in a value between -1 (exact opposite) to 1 (perfect match). The value of zero means total randomness. Our system scaled this MCC to fit within the range between 1 (perfect match) and 9,999 (exact opposite). The value of 5,000 indicates a total randomness.

3.2.4 XGP Tree Structure

The XGP would produce a schema in the form of XML, to represent a tree structure of the ‘program’. The tree structure consisted of multiple nodes, each node~~s~~ is a multiple variety of genes. Each gene (or node or branch) can be used by XGP for crossover, mutations, and anything else deemed needed for evolution. The root of the tree is separated into two distinct subtrees. One of them is the Valence subtree; the other is the Arousal subtree. Each subtree branched into a comparison between two of the branches below them. The value of the subtree (also the Valence and Arousal, respective to their subtrees) is set according to the condition of the comparison (to True or False). The branches below them are a combination between arithmetical genes, constant genes, and variable genes.

Our mathematical models (functions) represent the two axis of emotion, which are Valence (represented by the Valence tree) and Arousal (represented by the Arousal tree), as represented in Fig. 5.

Fig. 8 below illustrates the tree structure of our mathematical model.

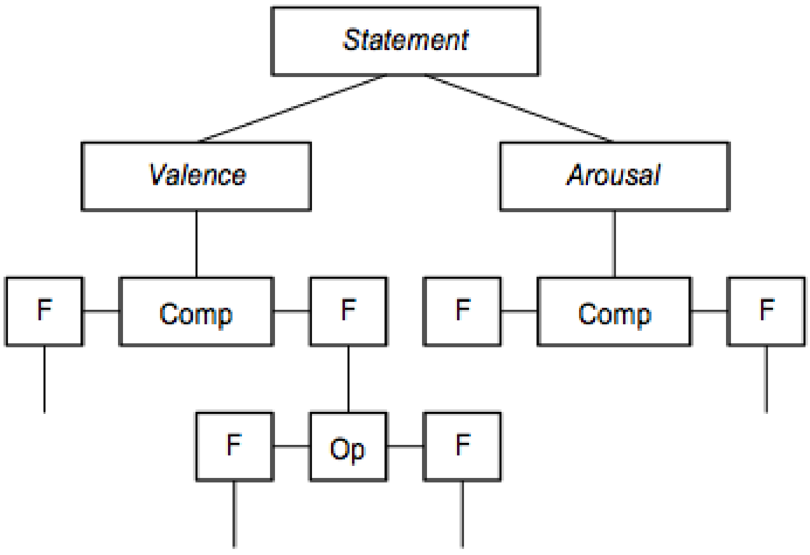


Figure 8. Tree Structure of Mathematical Model

The positive or negative of each Arousal and Valence is determined on the basis of the result of the comparison of the sub-trees using comparison operators. A 'true' result is considered 'positive', while a 'false' result is considered 'negative'.

Each parse tree is represented using the following Backus–Naur Form (BNF), along with the syntaxes, terminal sets, and functions:

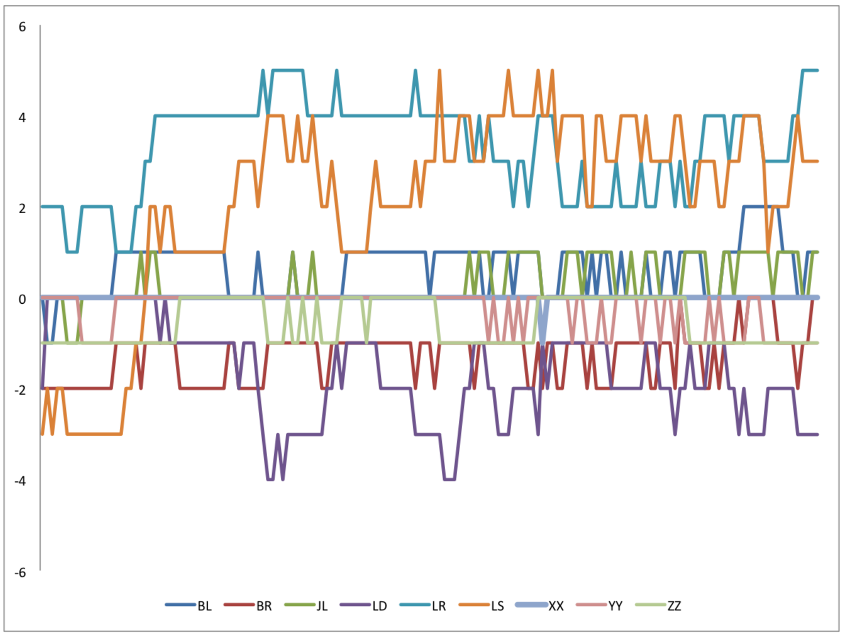
```
IF “F” “Comp” “F” THEN True

Comp::= “<”|“>”

F::= “Const”|“Var”|“F”,”Op”,”Const”|”Var”|”F”

Const::= “1”..”10”

Var::= “v_0”..”v_26”

Op::= “+”|”-“|”*”|”/”
```

3.4 Preliminary Experiments and Results

3.4.1 Users might express differently

Our first experiment is intended to test the Kinect to acquire the AUs of two separate users, each expressing *happy* emotions. The graphical representations of the signals of all of the AUs along the time for both users are shown in Fig. 9 below.

As Fig. 9 illustrates, the facial expressions of two different persons showing same emotions can be different. Using this information as one of the basis of our research, along with the concept written at the book “End of Average” [5], we proposed user-specific emotion recognition model.



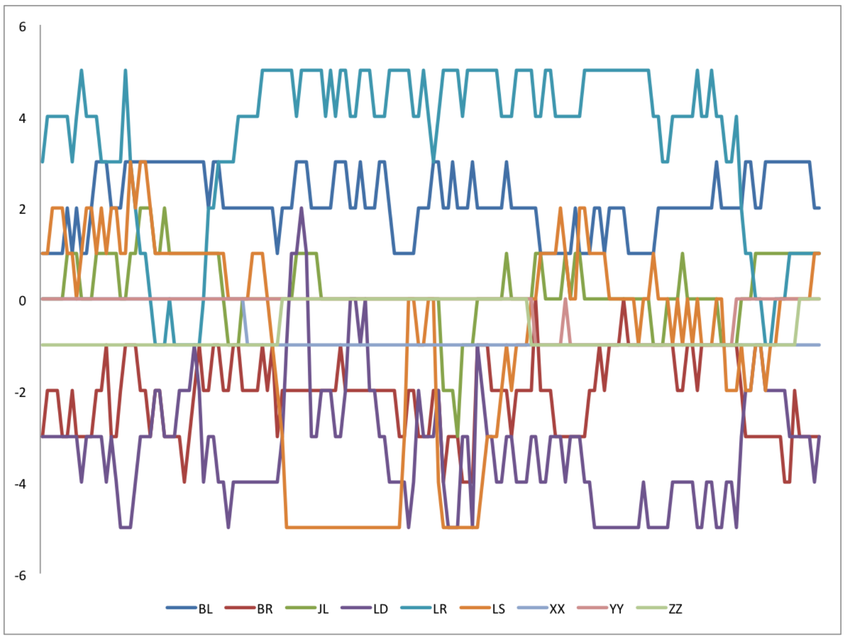


Figure 9. Signals of all AUs from two different persons showing same emotion on a similar time period

3.4.2 Analogy of Approach: Expert versus Close Relative/Family

The model that we propose has a novel approach in shifting the paradigm of emotion recognition. Most researches focus on developing and designing an expert agent, capable of recognizing many users' emotions, by extensive learning and accumulating massive training data. This is quite similar to training a person to be an expert psychologist who learns a lot about everyone.

However, our approach tries to shift emotion recognition towards different paradigm: by interacting with a single unique user through time, accumulating data through time, and evolving through time. This is quite similar to a close relative of ours, who often knows about our emotions, albeit they are not psychologists themselves (and they might be unable to understand strangers' emotions too).

3.3 Proposed Model

3.3.1 Overview

User-Specific Model

The aim of our research is to develop a user-specific emotion recognition model that can recognize the emotions of a unique user and can self-evolve. Our study is based on our preliminary findings that show that people may express their emotions differently as shown at Section 3.4.1, and that a generalized method may not be better than a specialized unique recognition method using a specially crafted classifier.

As an analogy, sometimes, someone close to us knows our emotions better than a psychologist, even if they are not psychologists themselves, e.g., our mother or spouse or another family member tend to know us better although they may not have special training in psychology.

This is also inline with the paradigm presented in the book "The End of Average" [5], that individuals might be unique and when designing something to fit the average, then it will fit to none, because there is no one who really fits the 'average'.

Pervasive Model

Acquiring subject’s data outside their natural context will not be applicable for real-life applications. Meanwhile, the uses of non-pervasive sensors or environment would result the subject to be taken out from their natural context, and might influence the data acquired.

Our model uses a pervasive sensor from Microsoft Kinect. Microsoft Kinect acts similarly to a webcam and can be used as a source of perceptions for the system. Fig. 10 illustrates the structure of the proposed system of the intelligent agent [29].

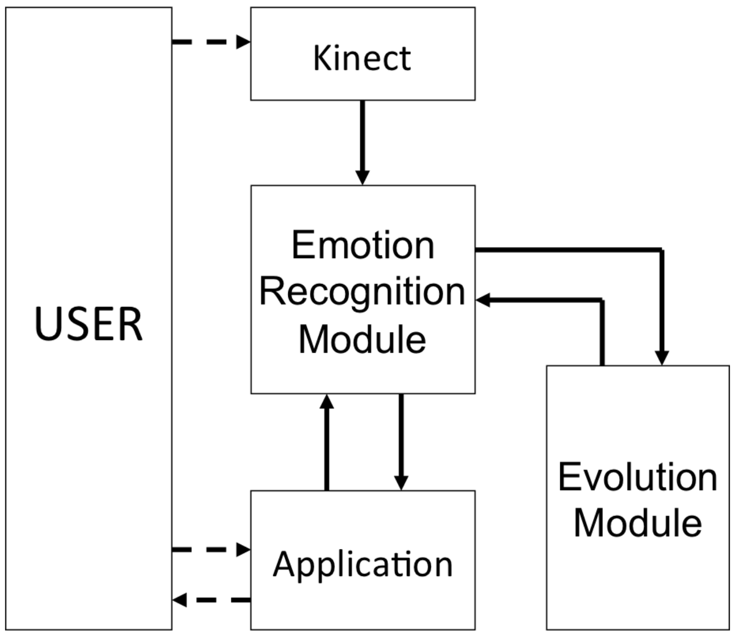


Figure 10. Illustration of Proposed Model

There are several reasons for choosing Microsoft Kinect as a sensor. The first is because it can be categorized as pervasive sensor. A non-pervasive sensor might influence a user's actual emotion, making the user more anxious or uneasy. Thus, using a pervasive sensor such as Microsoft Kinect might reduce the influence of an alien sensor on the user.

Another reason is that Microsoft Kinect is an off-the-shelf commodity, widely available as a practical tool for home personal computers (PC). It is not dedicated to a computer agent but is a general device that can be used as a webcam and a gaming device.

Further, Microsoft Kinect can operate under normal room situations, and do not need special attachments on the users, or special setups on the background. The working range of the sensor has similar condition with a simple webcam, which are normally illuminated room (not too bright or too dark), and that the user is located in the camera’s point of view.

In short, we chose Microsoft Kinect because of the pervasiveness and multi-usability, along the capability of both red-green-blue (RGB) camera and infra-red (IR) camera for better face-tracking capabilities.

3.3.2 Components and Features of Proposed Model

Our model has several important components (as shown on Fig. 10 above), as well as several features that can be explained according to Table 2 and Table 3 below. [30].

Table 2. Components of the Proposed Model

Component	Details
User	A single user who interacts with the system
Kinect	The main sensor of the system, consisting of several cameras and microphones
Emotion Recognition Module	The main module to manage and synchronize flows of data, including the module for machine learning
Evolution Module	Performing simulated evolutions to evolve classifiers
User Interface	To interact with the user

Table 3. Main Features of the Proposed Model

Category	Descriptions
Perceptions	Features extracted
	Collected features from database
Actions	Features to be stored in database
	Features to be deleted in database
	Proper response
Behaviors	Recognition (classifier)
	Self-evolution
	Analyze

3.4 Initial Experiments

The initial experiments are conducted to test the XGP, as well as exploring the evolutionary computation to obtain more efficient values for the XGP settings [25].

3.4.1 Experiment Environment

Many researches concerning facial expressions-based emotion recognition conducted experiments or acquiring training data (and test data) from people (or even actors) who are asked to express a single emotion, despite this method of acquiring data is often criticized [6].

Our proposed model avoid asking subjects to express their emotion; instead we let the subjects to be with their own, watching videos or performing other activities in front of a computer, while selecting appropriate emotions during the data acquisition; one of the options is [standby], which means to let the system not taking any data.

Fig. 11 below shows the common experiment environment of our proposed model.

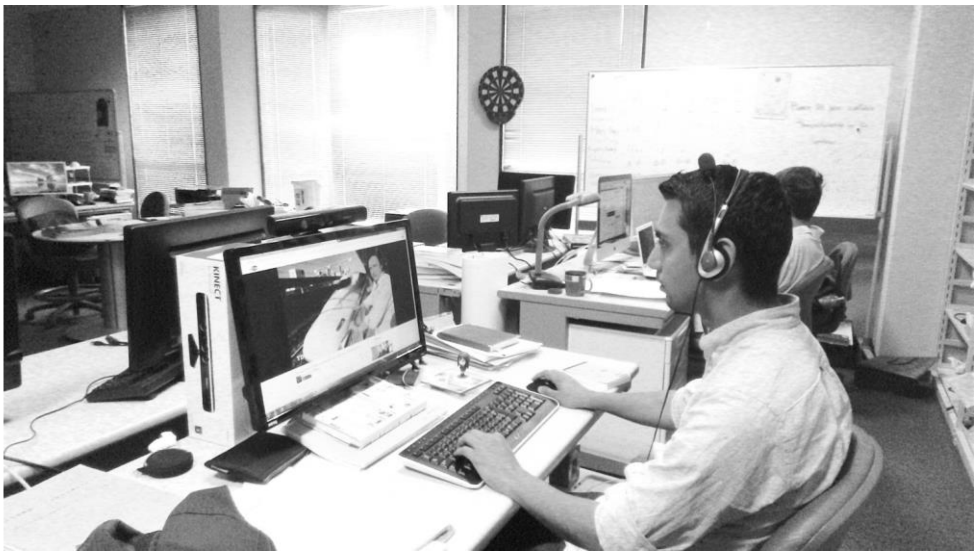


Figure 11. Common environment for experiments and data acquisitions

As it can be seen at Fig. 11 above, the subject uses the PC with a pervasive sensor of Microsoft Kinect (located on top of the monitor), under normal working environment. The subject is free to watch videos or perform other activities, and even to adjust his seating or moves around a bit, as long as he did not go out of Microsoft Kinect view or working condition; the headphone was not mandatory, but the subject shown at the Fig. 11 was watching a video while selecting some emotions related to his current situation (subject to his subjectivity). For the experiments, we also use another setup to test the robustness of the Kinect that we also use in gathering data, as shown at Fig. 12 below.

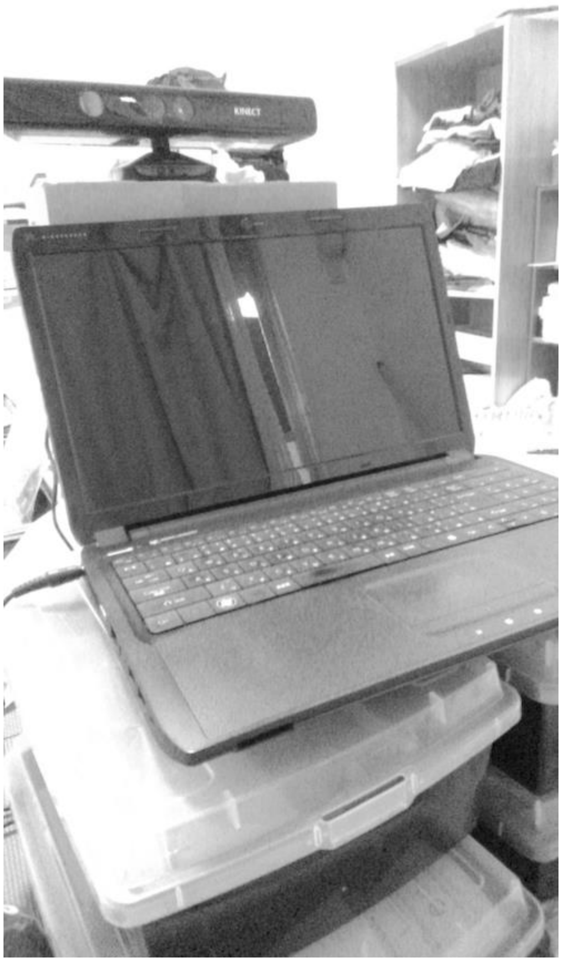


Figure 12. Alternative Experiment Environment

The environment above was set for alternative data acquisition that we use for our research.

The screenshot (cropped) of User Interface of the system we use in our proposed model is shown at Fig. 13 below.

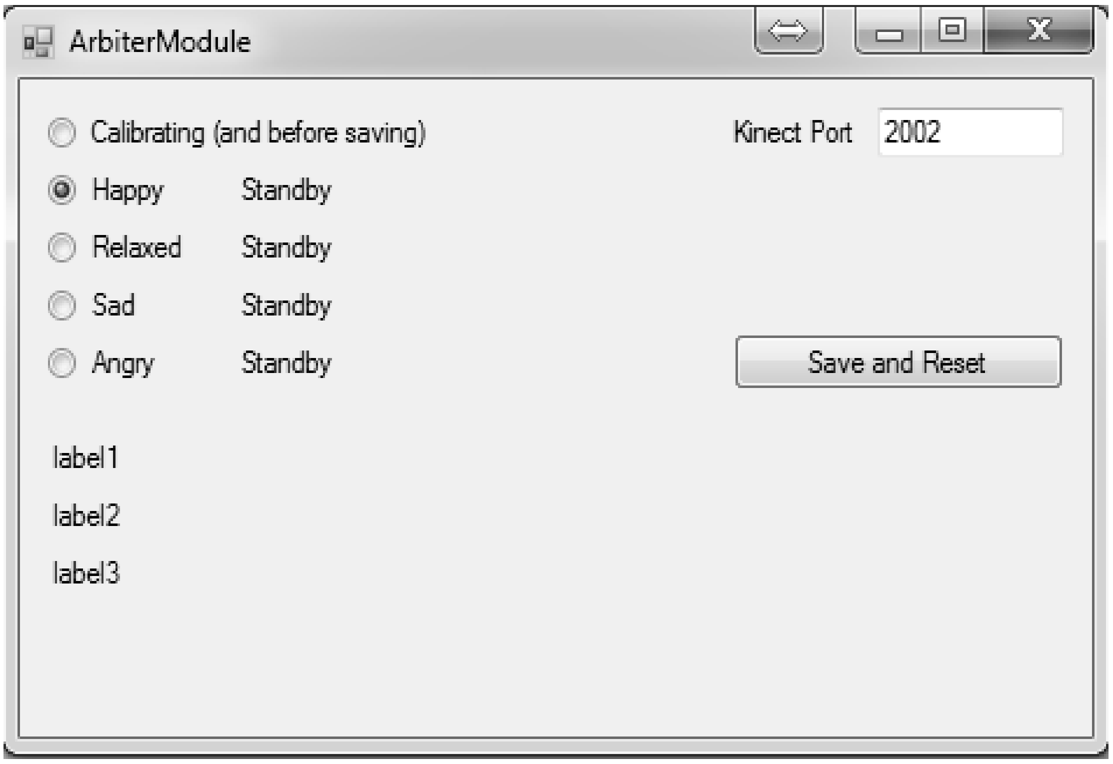


Figure 13. User Interface of the system of the proposed model; it is a small window resizable and can be minimized, so it is still pervasive.

The subjects are free to select and express their emotions, while using PC as usual. Afterwards, the subjects can save the acquired data into a file, to be used as training data for simulated evolution, or for test data for off-line simulated tests.

3.4.2 Feature Extraction Experiment

One of the first experiments conducted is comparing the data of two persons who express the same emotion for around 30 seconds. The persons then express another emotion for around 1 minute.

The extracted features then compared and analyzed. The result was that each person does not have a similar graph. These data then put into an evolution of several runs. The result was by separating the person into a different classifier (customized) would improve the accuracy of the classifier, compared with putting both data on a single evolution.

The graph of extracted raw features between two persons expressing the same emotion can be seen in Fig. 9 at Section 3.4.1. The graph shows that, the lip stretcher (LS) is one of the positive factors for the first person in expressing one emotion, while it is a quite negative factor for the second person in expressing the same emotion. In the data acquisition for emotions, there are cases where a person does not smile a lot while happy (despite popular belief), and not all person frowns when they are sad.

3.4.3 Time Frame

The features extracted also included an Activity, which is dependent on the changes of value in one time frame. An experiment is performed to test which time frame is feasible.

The experiment uses a single person data, expressing two kinds of emotion. The data is separated into training set and test set. A same number of 48 independent XGP session runs were performed for two training events, first for 30 frames data (around 1 second) of time frame and then for 100 frames data (around 3 seconds) of time frame.

The settings for the XGP are shown at Table 4 below:

Table 4. XGP Settings for Experiments on Time Frame

XGP Setting	*Value*
Termination	Generation > 200 | Fitness Value < 2 | Stagnation
# of Individuals	100
Elite Individuals	2
Selection Rate	10%
Mutation Rate	2%

The experiments then yielded result as follows:

Table 5. Time Frame Experiment, comparing 30 frames and 100 frames

Category	*30 frames*	*100 frames*
Average Fitness (training)	85.60%	86.84%
Average Fitness (test)	61.25%	82.10%
Best Fitness (training)	93.18%	92.10%
Best Fitness (test)	84.18%	97.49%
Best Individual (average)	85.23%	86.13%
Average Generations	81	127
Average Nodes	202	405

Average Fitness is the average of all result. Best Fitness is the best result from the whole sessions. Best Individual is the individual with good enough result for both training set and test set (by averaging them). Average Generations is the average generations needed to have the result. Average Nodes is the average number of Nodes of the best individuals from the whole sessions.

From the result shown in Table 5 above, it is safe to conclude that using 100 frames instead of 30 frames would give better result.

3.4.4 Bloat Penalty

As a common sense, a smaller tree would represent a better and faster operation. However, when the system is complex, sometimes it is better to have a bit larger tree, to put into considerations multiple variables/features at once.

An experiment is set to test this hypothesis. The controlled environment is achieved by setting a similar setup for data and XGP settings. The differences are just Bloat Penalty, maximum stagnation, and initial depth of the tree.

The experiment resulted in an increase of accuracy, but also increasing the number of generations needed to achieve the result.

3.4.5 Different Test Data

During different days, people might change their behavior. Environmental conditions during the different day of data acquisition might also changes, in example the position of seating, the lightings of the room, etc.

An experiment was performed to test the effect of environmental changes on the recognition. This experiment uses a data set from one day for training, and two test sets: one of the same day and the other from a totally different day (with different environmental setting to add noise). None of the test data set is used for training.

The training set consisted of 120 rows of data, while each test sets consisted of 20 rows of data.

The settings for the XGP are shown below:

Table 6. XGP Settings for Experiment on Different Test Data

XGP Setting	*Value*
Termination	Generation > 300 | Fitness Value < 2 | Stagnation
# of Individuals	100
Elite Individuals	2
Selection Rate	10%
Mutation Rate	3%

The results are as follows:

Table 7. Comparison between same-day data and different-day data

Category	*Same-day*	*Different-day*
Average Fitness	82.10%	66.24%
Standard Deviation Fitness	1.74%	7.17%
Best Fitness	97.49%	80.02%

It is also worth to notice that the average fitness of the training sets was 86.84% with a standard deviation of 2.98%. The best fitness for the training sets was 92.10%.

From the results shown on Table 7 above we can safely conclude that the change of environment condition significantly lowers the accuracy.

3.4.6 Calculating Time Needed For Evolutions

The time needed for evolution was calculated by putting a time-stamp on each generation. The result was more or less homogenous, despite other changes in XGP setting or the evaluator setting. The number of individuals per generation stays at 100 individuals.

The time for each generation needed using XGP and evaluator is around 36 seconds.

However it should be noted that the time also includes the UDP transfer.

3.4.7 Separating Valence and Arousal Tree

Previous experiments were using a single tree to be evolved using XGP. However, upon short analysis on the tree results, it can be seen that there were cases where the GP would evolve the ‘wrong’ tree.

This is the result of the fitness function that calculates both tree at once, thus the XGP does not receive information about which tree needs improvement. The XGP only receives information regarding the general accuracy of both trees.

An experiment was then prepared to investigate the possibility of splitting trees, and the comparing the computational effectiveness and computational efficiency. Computational effectiveness can be seen by the fitness result, while computational efficiency can be seen from the time and generations needed to achieve result.

In order to perform the experiments, two instances of XGP are prepared, as the tree schema must be changed. One XGP is similar to the XGP used on previous experiments, while the other XGP is using a different schema. The schema is edited by removing one branch altogether, and run separate sessions for Valence tree and Arousal tree.

The training data used for all session runs are the same. The other settings for XGP are using the settings shown on Table 6. The result of the experiments can be seen from the Table 8, below:

Table 8. Raw Comparisons of Valence, Arousal, and Both Tree

Category	*Valence*	*Arousal*	*Both*
Average Fitness	88.51%	87.89%	84.46%
Stdev Fitness	2.72%	2.99%	4.09%
Best Fitness	93.69%	92.18%	89.32%
Worst Fitness	82.31%	81.93%	77.99%
Average Generation	90.75	92.8	150.45
Stdev Generation	32.49	35.79	63.68
Average Node	307.1	334.75	463.09
Stdev Node	263.29	244.76	276.37
Average Total Minute	58.25	62.85	99.27
Stdev Total Minute	20.96	30.44	43.63
Average Second/Generation	38.59	39.8	39.49
Stdev Second/Generation	2.05	3.73	0.90

From Table 8, it can be seen that the computational effectiveness increases when separating the trees and evolve them on their own. Not only the average fitness is increased, the best fitness also increased and the worst fitness value also decreased quite significantly.

Unfortunately, the generations and time needed to evolve separately is higher than to evolve them using a single XGP. Therefore it can be said that separating the tree will reduce the computational efficiency of the system.

Another interesting point is the standard deviation of the result. The variance of fitness value and number of generations from separated tree is much lower than the single tree. Meanwhile, the standard deviation of average time needed to evaluate a single generations is lower when processing a single tree.

3.5 Conclusions

In this chapter, we investigated the emotions classification to be used in our research, and chose a variant of Circumplex model for the simplicity for the user in selecting emotions subjectively. Further, we designed the tree to represent the emotions so XGP can evolve programs. We also selected the model to represent facial expressions and the emotions, which are CANDIDE 3 and Facial Action Coding System. The pervasive sensor Kinect has been chosen as well, and we have stated the reasons in selecting the sensor. We then conducted preliminary experiment to test the sensor and the model, as well as to clarify our novel approach is feasible and reasonable to be implemented, supported with the theory from The End of Average [5]. Then, we proposed the model, represented by a system block diagram, and performed initial experiments to set several crucial values in evolving the programs or classifiers.

We also have shown the pervasiveness of the system of the model we proposed, and that there is no special setting needed for the experiment environment. The user interface also can be minimized in order to maintain the pervasiveness of the system.

Our initial experiment results also suggested that evolving a classifier using data of two persons would result to a classifier performing less than the one evolved using data of single person, because not everyone express their emotions the same way.

Another findings that we found was that separating simulated evolution for Valence and Arousal give better computational performance on test, compared to running the simulated evolution using both trees (of Valence and Arousal) at once, due to possibility of unnecessary crossover between the trees. The computational efforts are reduced due to the longer time needed to evolve and to combine the tree.

Regardless of the results, the end results still have several problems such as the possibility of over-fitting and the lack of generality of genetic programming due to its non-determinism. These issues will be discussed on the next chapter.

Chapter 4

Enhancing the Model: Implementation of Voting

This chapter aims to tackle the problems of genetic programming (GP), namely generality. To overcome with the problem and improve the accuracy of the classifier, we select several classifiers to act as voters. The guesses are based on the majority vote of these several classifiers. We use XGP to evolve the programs at this experiment. [31]

4.1 GP Limitations: Generality and Over-fitting

The lack of generality (non-determinism) has been a problem for GP due to the random factor persisting in most of GP operation such as generating initial population, performing crossover, or applying mutation. Further, as mentioned at previous chapter, several simulated evolutions are needed to select 'best of the best' program evolved by the GP.

However, there is no guarantee that the 'best of the best' program can understand the generality of the problem, due to various reasons, two of which are:

- The training data used to evolve are not diverse enough, thus several situations that might occur on test data are not represented at the training data.
- Over-fitting (the evolution worked how to tackle the problem represented at the training data only).

Since our proposed model does not work on theoretical or ideal data, but works on practical data gathered from user, the first situation above (lack of diversity) has a high chance of appearing (especially with the user selecting their own emotion). Therefore, in order to improve the accuracy, the model should be able to at least handle over-fitting problem.

Fig. 14 below shows the sample of a fitness convergence of a simulated evolutions consisted of several independent XGP runs.

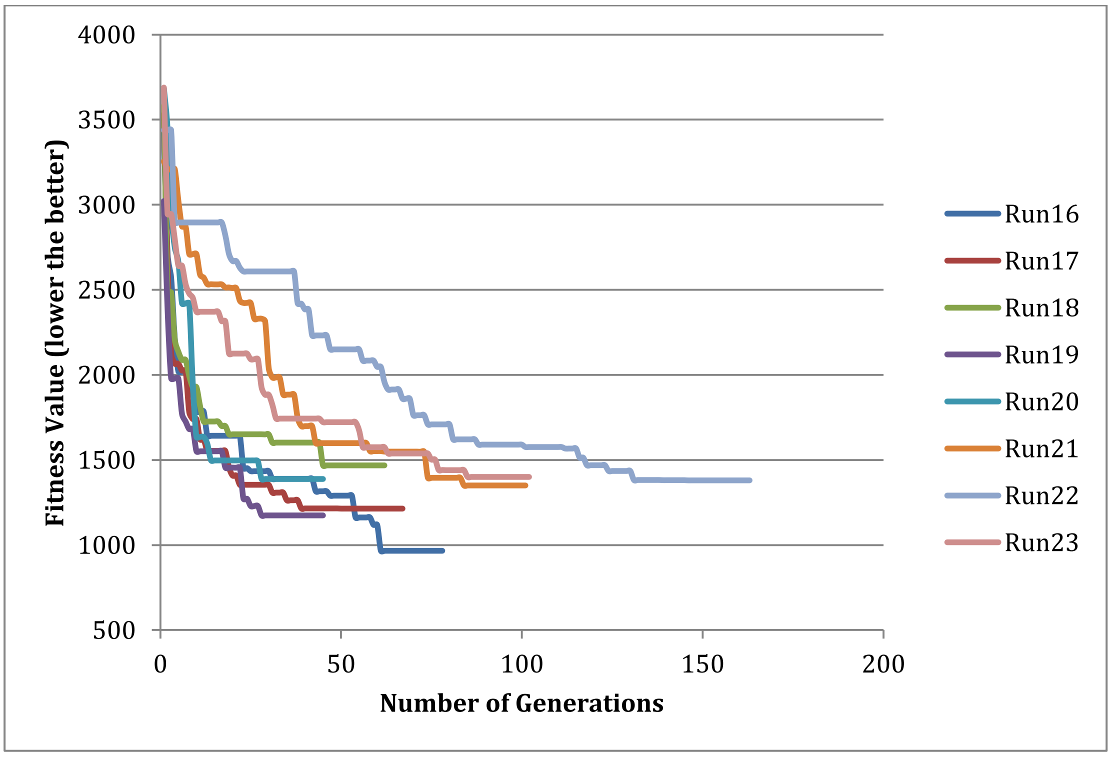


Figure 14. Fitness convergence sample of several independent XGP runs (y-axis=fitness value, x-axis=number of generation)

4.2 Collaborative Filtering: Voting by Majority

One method to overcome this problem is by having several programs evolved to act as voters instead of a single voter. The intention of this is to reduce the chance of selecting an over-fitted program and having more chance of selecting good enough programs.

The voting by majority can be done by selecting 5 best-evolved programs (based on fitness value) to act as voters. The final guess is selected by taking what the majority of the programs guess.

4.3 Experiments and Results

The *objective* of this experiment is to investigate the effectiveness of Genetic Programming (GP) to evolve emotion recognition module, and to verify the feasibility of using a voting among several evolved agents in order to address the well-documented drawback of the non-determinism of GP.

The evolutions were simulated using XGP, with the same fitness function explained at previous chapter.

4.3.1 General setup

For the experiments we have prepared a setup standard for data acquisition and evolution of mathematical function for emotion recognition via XGP. In our research we used two mutually exclusive sets of features - training set and test set, respectively.

We experimented with the recognition of emotions of two subjects, labeled as R1 and D1, respectively. For each of these two subjects, we used both training- and test sets of data, as follows:

- R1 training set,

- R1 test set,
- D1 training set, and
- D1 test set

The results of our previous experiment indicated that the system performed relatively poorly when the test set is sampled in an environment that is totally different from the environment used in the training set (e.g., different ambient lighting schema). In order to improve the generality of the evolved emotion recognition, in the current experiments, we added data, sampled in a poor lighting, into the training set of R1.

The main parameters of XGP are shown in Table 9 below.

Table 9. Main Parameters of XGP

XGP setting	**Value**
Termination criteria	Generations = 300, or Fitness < 2, or Stagnation for 32 generations
Population Size	100
Elite	2 individuals
Selection rate	10%
Mutation rate	3%

4.3.2. XGP independent sessions

Each of the training sets (D1 train sets, R1 train sets) are used for two batches of XGP run, with each batch consisted of 20independent sessions.

On previous chapter we have described the overall results and convergence of each independent run, and current experiments also showed similar results. As we focus on effectiveness (accuracy) of our current system, it is sufficient to state that the time needed for current experiments are very similar to our previous reports.

Current experiments also use two methods on evolving classifiers. First, by evolving valence classifier and arousal classifier as a single program, thus the branches from valence can crossover to branches at arousal, and vice versa. Second, by evolving the valence as a separate tree, and arousal as separate tree, then combines them as a single tree, thus avoiding crossover between valence and arousal.

4.3.3. Evolution results

To evaluate the effectiveness of the classifiers, we use average value and standard deviations, as well as minimum and maximum value, of a single batch (consisted of best individuals from each sessions) of tree.

Average value represents the general performance of the classifier, while standard deviations represent the general stability of the classifier. A minimum and maximum value can represent the best and worst performance of the classifier.

The values are the fitness values, scaling from 1 (perfect match) to 9999 (exact opposite). The value of 5000 means total randomness.

The evolution results (fitness value) of R1 from the 1st batch and 2nd batch are shown in Table 10 and Table 11, respectively:

Table 10. Training result for 1st batch of R1

Category	Average	Std dev	Min	Max
Both trees:				
- Fitness	1403	253	1004	2092
- #Generations	146	34	93	200
Arousal only:				
- Fitness	1242	402	746	1877
- #Generations	74	29	40	139
Valence only:				
- Fitness	1421	303	983	2161
- #Generations	94	39	40	174

Table 11. Training results for 2nd batch of R1

Category	Average	Std dev	Min	Max
Both tree:				
- Fitness	1350	269	956	1830
- #Generations	152	58	93	286
Arousal only:				
- Fitness	1249	450	507	2092
- #Generations	90	43	41	211
Valence only:				
- Fitness	1250	270	810	1971
- #Generations	91	35	39	188

The evolution results (fitness value) of D1 from the 1st batch and 2nd batch are shown in the Table 12 and Table 13, respectively:

Table 12. Training results for 1st batch of D1

Category	Average	Std dev	Min	Max
Both tree:				
- Fitness	900	285	471	1320
- #Generations	87	25	50	141
Arousal only:				
- Fitness	628	397	1	1525
- #Generations	67	22	41	128
Valence only:				
- Fitness	752	406	191	1710
- #Generations	66	20	40	106

Table 13. Training results for 2nd batch of D1

Category	Average	Stdev	Min	Max
Both tree:				
- Fitness	787	322	275	1534
- #Generations	94	27	55	143
Arousal only:				
- Fitness	664	416	1	1790
- #Generations	68	16	34	100
Valence only:				
- Fitness	856	443	191	1702
- #Generations	67	22	33	118

4.3.4 Evaluation on accuracy and Voting

To perform an evaluation on the programs created by XGP, the experiments use a different test set, namely D1 test sets and R1 test sets.

In the experiments we chose the evolved 5 best-of-run recognition functions and test them, comparing between without Voting and with Voting, as well as by testing them using a test sets from different subject. Voting means using majority decisions to give predictions.

Experimental results of the evaluations are shown in Table 14 below. *Train* denotes the accuracy of emotion category guessed by the classifier on the training sets, while *Test* refers to the appropriate Test sets. *Cross* indicates the test sets of another subjects.

Table 14. Evaluation on best-of-run test results

Evolved best-of-run test results	R1 bests, %			D1 bests, %		
	Train	*Test*	*Cross*	*Train*	*Test*	*Cross*
Combine1	80	70	20	96	79	25
Combine2	80	55	35	94	76	20
Combine3	79	50	45	94	72	50
Combine4	79	55	57	92	76	40
Combine5	79	70	27	94	88	35
Separate1	87	70	47	98	76	35
Separate2	84	60	44	98	82	40
Separate3	83	55	35	96	85	35
Separate4	83	75	17	96	86	55
Separate5	84	70	35	94	80	35
Avg Combi	*79.4*	*60*	*36.8*	*94*	*78.2*	*32*
Avg separate	*84.2*	*66*	*35.6*	*96.4*	*81.8*	*40*
Vote combi	84	60	38	98	85	35
Vote separate	87	65	39	100	85	45

Table 14 above suggests that Voting will almost always giving a better result than the average, unless on one occasion where the vote result is not better than the average.

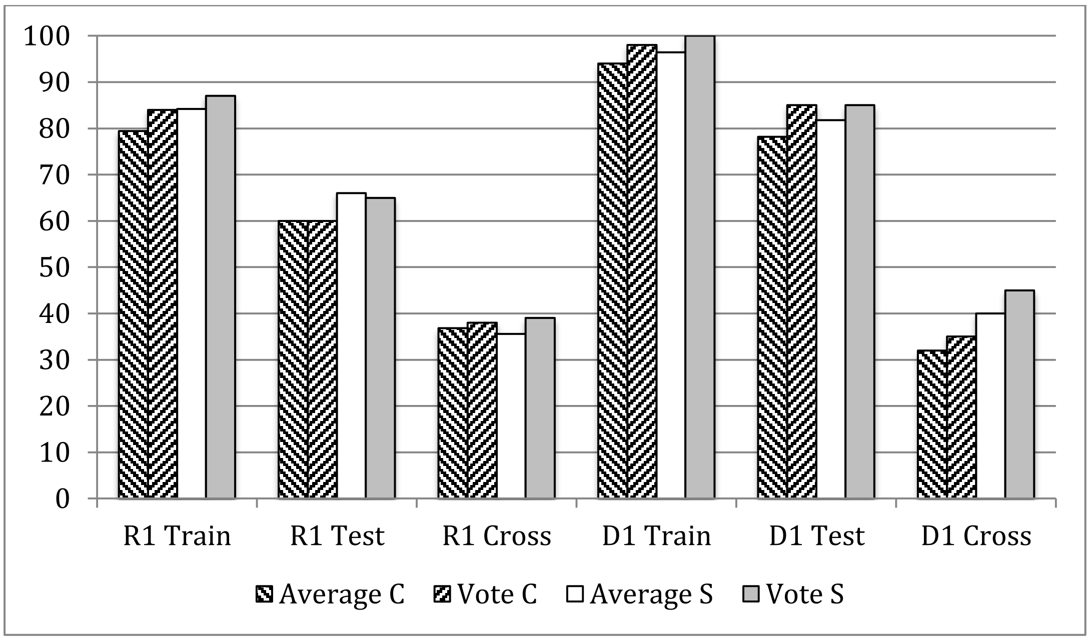


Figure 15. The graphic showing accuracy on test result (%) using several methods: Average C (averaging, combined tree), Vote C (voting, combined tree), Average S (averaging, separated tree), Vote S (voting, separated tree); and also several situations: R1 and D1 (subjects), Train (on training data set), Test (on test data set), Cross (on other subject's data set)

We can also observe the uniqueness of each subject in expressing emotions by the low accuracy of the cross-testing (using other's data for testing) results, despite the achieved good accuracy of the normal (using own's test data for testing) test results; this includes already in adding several new data to the subject for different seating arrangements and lighting conditions.

4.5 Conclusions

The results from the experiments suggest that people tend to express emotions differently, and by focusing on a single (user-specific) subject a better classifier can be evolved.

Genetic programming is also proven to be able to evolve classifiers with good enough results. In order to reduce the effect of GP's non-determinism (and lack of generality), we implemented collaborative filtering in the form of voting by majority to further improve the accuracy of our model, and the results show that the voting by majority improved the accuracy.

However, further improvements and enhancements can be performed.

For improvements of the classifiers, the next chapters will discuss about our effort in exploration of evolutionary computing, by conducting a second evolution (basically by giving weights or credibility to the voters) and evolve them using a separate training sets; and then about exploring the use of incremental genetic programming (incremental GP).

Chapter 5

Exploring Evolutionary Computation: Adaptive Voting

Previous chapter explained about the implementation of collaborative filtering in the form of voting. This chapter aims at improving the performance of the emotion recognition model, by exploring evolutionary computation namely the genetic algorithm (GA). [32]

5.1 Concept of Weighted Trust

On previous Chapter 4, our experiments showed that implementing collaborative filtering in the form of majority voting improves the result on test subjects. They also showed that the 'best-of-the-best' evolved program sometimes did not perform well on test subjects, due to chance of over-fitting and GP's non-determinism. Therefore, by selecting an elite group, we reduced the effect of over-fitting. The *voting by majority* also performed better than the average of individual performers.

However, we would like to explore the evolutionary computation a bit further, by evolving the *weighted trust*.

The concept of *weighted trust* is an adaptation of common *voting by majority*. In *voting by majority*, each voter has equal value in decision-making. However, there are several conditions when the voters might be not that good at decision-making; therefore we would like to explore the use of evolutionary computation in making adaptive system.

5.2 Problem Statement

The adaptive system of *weighted trust* is necessary to tackle the problem our model faces: 'how to adapt to a new data without forfeiting the old data'. Our model requires the system developed to be able to adapt to a new user data, since the idea is to learn the user through interactions. This means that there will always be newer data, while the method previously explained (even with the collaborative filtering of majority voting) will have a lot of difficulties in using new data for training.

There are several methods, first by simply piling up the whole data to the old one, and perform the evolution again from zero. However, this would costs too much time in the evolution as the number of data would keep on piling up.

If we only use a specific data (e.g. the latest one), then the classifier might not be prepared well for data similar to the old one. We would like to have a model that can adapt itself to new data (or new user), and therefore we conducted an exploration experiment in adaptive weighted trust.

This chapter then will report the experimental results in our attempt in developing *weighted trust*.

5.3 Implementing Genetic Algorithm for Weighted Trust

Genetic algorithm can evolve the weighted trust, as the weighted trust is an array of integers only. Unlike genetic programming with possible program (function) evolution, genetic algorithm can evolve faster but at a more limited scope.

Due to this simplicity, we explored the genetic algorithm to evolve our weighted trust model, and conduct an experiment to compare the result between *voting by majority* and *weighted trust*. [33]

5.4 Experiments

The setup for XGP to evolve classifiers are similar with previous works, which are as follow:

- Termination criteria, if # of generations reach 300, or fitness value reaches below 2, or stagnation for 32 generations,
- Population size is 100 individuals,
- Elite individuals are 2 individuals,
- Selection rate is 10%, and
- Mutation rate is 3%

5.4.1. Experiment setup

The experiment is set into several configurations, and each configurations use several batches. Each batch of experiment consists of several XGP sessions, where each sessions will have an XGP runs (selections, crossovers, mutations) for several generations until termination criteria is reached.

There are two basic configurations, the genetic programming and voting configuration, and the genetic programming and credibility evolution.

For voting configuration, the genetic programming will evolve 10 (ten) classifiers, then the 5 (five) best of these 10 (ten) classifiers will be used for voting. Voting will be performed by using the rule of majority: if 3 (three) or more of the classifiers vote for true, then the value is true.

Fig. 16 describes the voting configuration, while Fig. 17 represents credibility evolution.

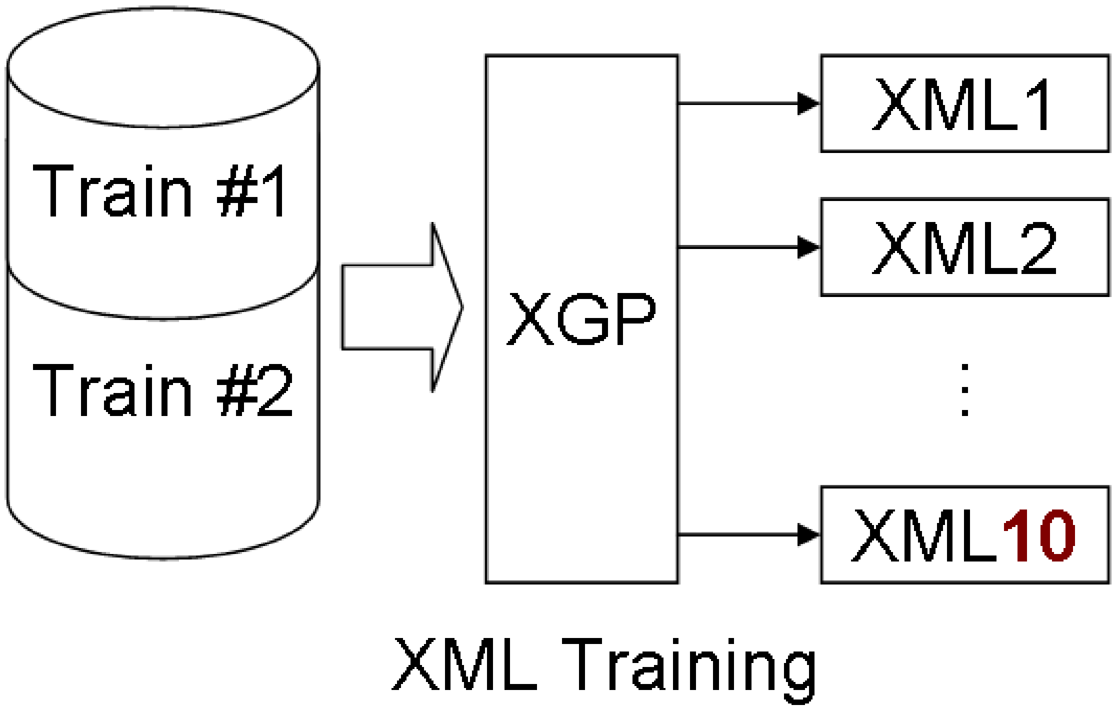


Figure 16. Configuration for simulated evolution to evolve classifiers for normal voting

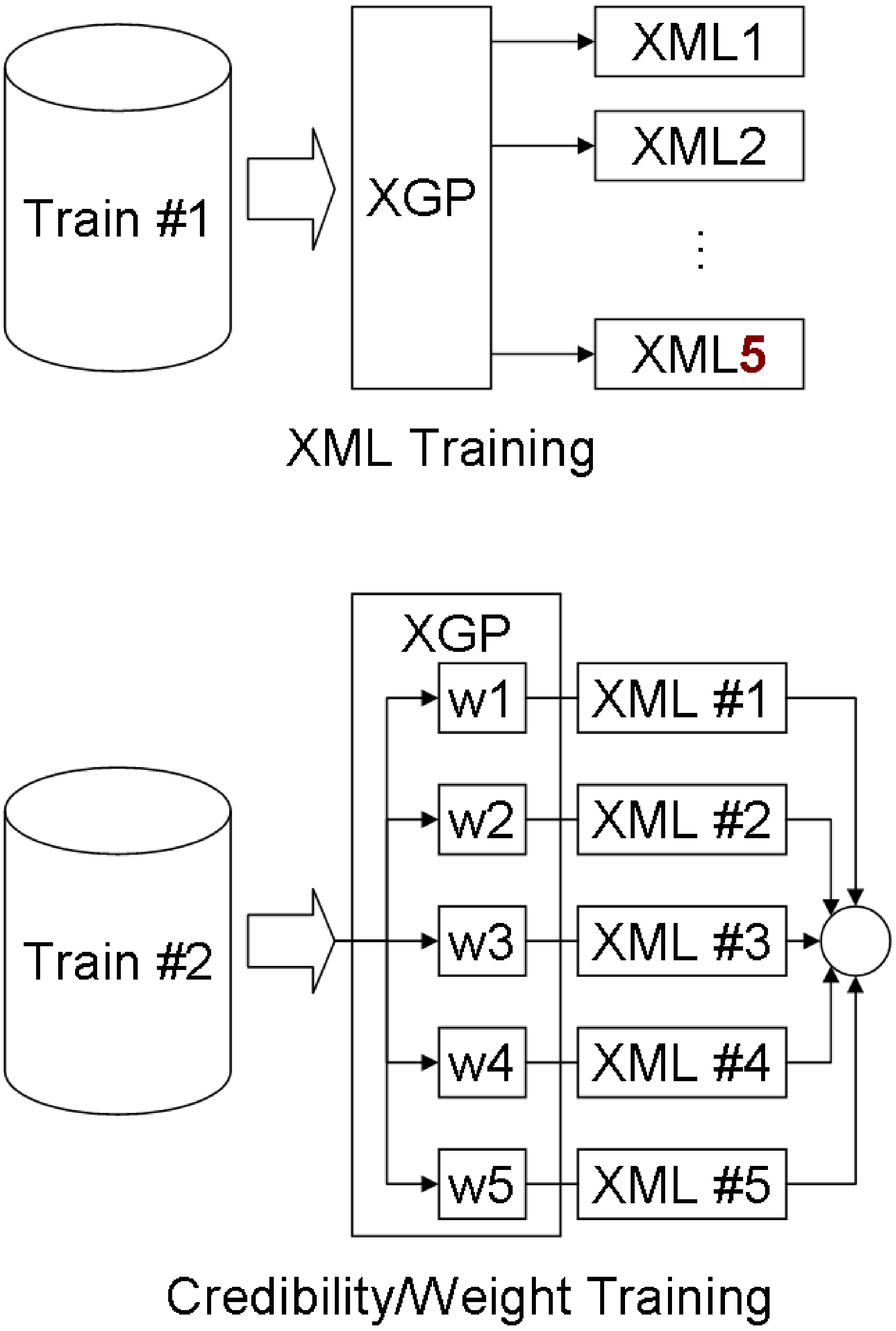


Figure 17. Configuration for simulated evolution to evolve classifiers for weight voting

For configuration of weighted credibility evolution, the genetic programming will only evolve 5 (five) classifiers, which each will be multiplied by a weight evolution using genetic algorithm. This method uses autonomous learning, since no human is needed to select the better classifiers to be used.

5.4.2. Preparations for data acquisitions

There will be several data sets for this experiment, and this experiment makes sure that the data for training (*Train #1*, *Train #2*) and the data for testing (*Test*) are mutually exclusive, which means that no data will be deliberately copied to another data set, nor used more than once. Further, it also makes sure that data for *Train #1* and data for *Train #2* mutually exclusive usage as well, in order to obtain a better objective results.

5.4.3. Data acquisitions

There are three different data acquired, and all are from a single subject. Each of the data is taken from a different time period (morning, noon, late afternoon), with a different conditions regarding seating positions and Microsoft Kinect's position. There is no special setup on the positions, only the required sufficient lighting condition for the Microsoft Kinect, and the general direction of the device, to operate properly. This is meant to imply the randomness of human movement in front of the camera while operating computer, and also to make sure that these three sets of data are not from the same series.

5.4.5. Data classifications

The data is classified as three kinds of data: *Train #1*, *Train #2,* and *Test set.*

This experiment uses two method in preparing these three kinds of data. The first one is prepared by sampling randomly and equally from each time (morning data, noon data, late afternoon data). *Train #1, Train #2,* and *Test* data sets have mutually exclusive uses. This first method is used in order to test the general ideal condition of training, by putting more combinations of data into each set.

The second method is by using morning data, noon data, and late afternoon data as *Train #1, Train #2,* and *Test* data, respectively. This second method is used in order to test a practical applications where it is highly inconvenient to get ideal data which represents most situation.

Fig. 18 below shows the representation of the data classifications used in the experiment.

1st Data Capture	Morning (A)	Morning (B)	Morning (C)
2nd Data Capture	Noon (A)	Noon (B)	Noon (C)
3rd Data Capture	Afternoon (A)	Afternoon (B)	Afternoon (C)

Figure 18. Data acquired 3 times, and captured data classified into several chunks; the *ideal* data set will take (A), (B), and (C), while the *practical* data set will take 1st, 2nd, and 3rd, for Train#1, Train#2, and Test, respectively

5.5 Results and Discussions

There are 2 (two) configurations: *voting* and *credibility*. There are also 3 (three) batches of experiment for each configurations, where each batch also consisted of tree separation and combined tree evolution.

5.5.1. Performance (time and generations)

Evolving a single classifier using genetic programming (XGP) needs much more time and many generations compared to evolving a single weight combinations. Evolving the weight only takes a maximum of 15 minutes (40 generations), while to evolve a single classifier the minimum time required is more than 50 generations.

Using these figures only, it is safe to state that evolving credibility is far better than evolving more classifiers, in terms of simulated evolution time needed to prepare the classifiers.

5.5.2. Fitness convergence

The experiment results on fitness convergence, from all batches (2 configurations, 3 batches each, which has 2 tree structure types), are shown at Table 15 and Table 16, for ideal data set and practical data set, respectively. Average, standard deviation, as well as minimum and maximum of both # of generations and fitness (in the form of accuracy) is used to give a statistical performance of the method based on each batches.

Table 15. Fitness convergence results using *Ideal* data set for training

Voting Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness%	# Gen	Fitness%
Worst	224	81.91	296	83.07
Best	51	95.59	84	97.79
Average	127.83	89.94	164.10	91.15
Stdev	40.64	3.77	32.74	3.64
Credibility Evolution Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness%	# Gen	Fitness%
Worst	218	87.71	265	82.64
Best	70	99.97	75	99.14
Average	115.93	93.01	153.13	93.09
Stdev	41.46	3.44	35.08	4.97

Table 16. Fitness convergence results using *Practical* data set for training

Voting Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness%	# Gen	Fitness%
Worst	302	85.00	226	82.42
Best	55	95.91	73	99.14
Average	135.27	91.02	128.97	92.63
Stdev	43.73	2.86	20.81	4.17
Credibility Evolution Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness%	# Gen	Fitness%
Worst	143	84.55	234	82.16
Best	54	97.43	70	99.14
Average	92.00	91.27	128.40	92.40
Stdev	21.89	5.18	23.27	5.18

5.5.3. Accuracy

Results regarding accuracy is best described in Table 17 below. The data shown is based on the statistical result (worst, best, average, and standard deviation) from each batches.

Meanwhile, Fig. 19 below shows the graphical representative of averages and standard deviations of the results.

Table 17. Accuracy Results on Test Data Set

Ideal Test Data Set				
Category (%)	*Normal Voting*		*Credibility Evolution*	
	Combi	Separate	Combi	Separate
Worst	77.27	79.55	68.18	77.27
Best	90.91	81.82	79.55	81.82
Average	82.58	80.30	75.76	80.30
Stdev	7.31	1.31	6.56	2.62
Practical Data Set				
Category (%)	*Normal Voting*		*Credibility Evolution*	
	Combi	Separate	Combi	Separate
Worst	65.96	68.09	63.83	53.19
Best	76.60	74.47	68.09	74.47
Average	70.21	71.63	65.96	63.83
Stdev	5.63	3.25	2.13	10.64

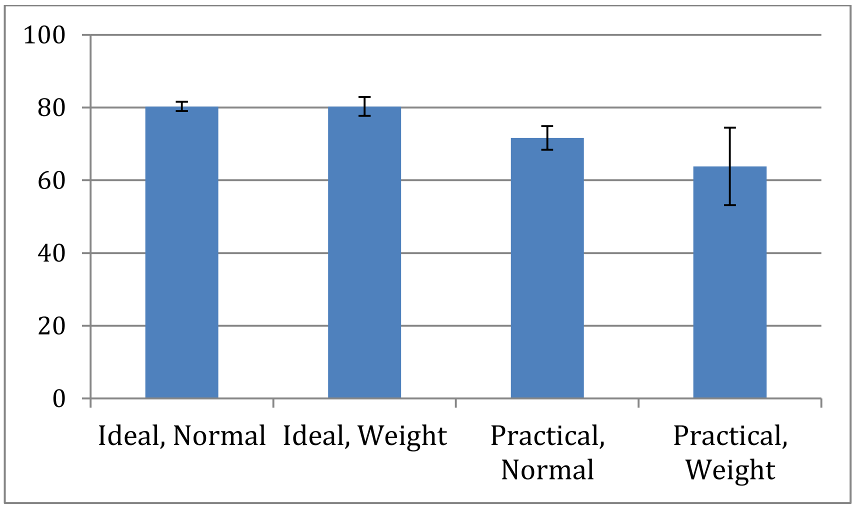


Figure 19. Stem height (mean ± s) of the results

5.6 Conclusions

Using genetic algorithm to evolve the credibility of classifiers automatically gives a clear advantage in the terms of computational performance, in the form of less time needed for evolution. Further, this method is capable of automated evolution, as the classifiers are not selected by human. This method offers possibility in completing a fully autonomous evolving intelligent agent.

The results might be not satisfactory enough, but it might be related to the insufficient number of data for training. Increasing training data might improve the result, as well as increasing the number of classifiers for voting and credibility evolutions. However, another approach can also be used to enhance the evolutionary computation, and will be discussed on the next chapter.

Chapter 6

Exploring Evolutionary Computation: Incremental GP

This chapter aims at tackling the difficulty of real-life situations where new data from the user might be gathered, or user has some changes in their behavior, or simply from new interactions between the user and the agent, thus adaptation is needed. This chapter aims to explore the evolutionary computation of incremental genetic programming, and also to analyze and evaluate their robustness.

6.1 Incremental Evolution / Incremental Genetic Programming

On previous chapter, an exploration on adaptive evolution using genetic algorithm to evolve *weighted trust of voters* was presented. The results suggested that the method was a trade-off between computational performances (accuracy on test subject) and computational effort (time for evolution).

Further, there is still a problem in handling situation when new data is introduced. Should the new data be used on the genetic algorithm, the base classifiers might have been not properly trained. Meanwhile if it is used to train a new set of classifier, then the results might be unfitting to the old data.

To deal with this situation, we explored another approach, which is to use incremental evolution in the form of incremental genetic programming (incremental GP).

Incremental GP is different with evolving weighted trusts of voters, since the evolution only uses genetic programming and not genetic algorithm. This approach uses another process of evolution using XGP, but by selecting several of previously evolved programs by the XGP as the seed instead of randomizing the first generation of each independent XGP run.

The selected individuals or programs were evolved using a set of training data, thus already adapted to the user's data. Then, with user's new data, the program is evolved again, with the aim of adapting to the new data. Using this approach the classifier can adapt to new data but still have some ability to recognize older data.

6.2 Experiments

We evolved the emotion recognition module by using Genetic Programming (GP) and explored several optimizations. We investigated and compared the evolution of a unique classifier (evolved from data from a single specific subject only), the evolution of a general classifier (evolved from data from multiple subjects), and the evolution of an adaptive classifier by implementing incremental GP (evolved incrementally, first from multiple subjects and then from a single specific subject). We conducted the experiment by using the same budget in terms of evolution sessions to obtain the best programs for a fair [34].

6.2.1 Experiment Setup

These experiments used XGP to perform the evolution, and also similar environmental settings with the experiments of the previous chapter. Further, there are several terminologies that these experiments still use:

- Genotype: a single gene that forms the program (function).
- Individual: a single program evolved by XGP, in the form of XML, with information of tree structure that represents the program.
- Generation: One generation of XGP runs. In the experiments, we use 100 individuals per generation.
- Sessions: Independent XGP runs until the termination criterion is reached. The termination criterion is one of the following: number of generations (300 is the maximum), fitness value (20, which implies that it is a near-perfect match), or stagnation period (32 generations of stagnation or no improvements in fitness value).
- Batches: Each batch consists of 20 XGP sessions, and the best are selected.
- Experiment: One experiment consists of several batches (each with a different setting) to be compared in order to investigate the strengths and weaknesses of each method. In the current experiment, the different setting corresponds to the data set used for training and testing.

6.2.2 Data Gathering and Preparations

In our study, we performed experiments using several subjects (n is the number of subjects). Data were taken from all the subjects, and were labeled as Data(i), where i = 1 to n.

From each Data (i), 75% of the data were labeled as Training (i) and the remaining 25% were labeled as Test (i). From each Training (i), 50% of the data were further labeled as General (i) and the remaining 50% were labeled as Adaptive (i).

The focus of the experiments was to compare the possibilities of self-evolving using incremental GP and to investigate the possibilities of introducing new data for learning into the existing system (thus, the system could evolve better and faster as the time spent with a unique user increased). For the comparison, we performed 3 types of experiments.

In the first batch of experiment, the task was to evolve genetic programs using Training (n) (training set from a single unique subject) as the training set. This task evolved programs specifically designed to recognize only a single unique subject. This experiment is called "1st Unique" experiment.

In the second batch experiment, the task was to evolve genetic programs using Training (i) (training set from all the subjects) as the training set. This task evolved general programs that are not built specifically for a particular subject. This experiment is called "2nd General" experiment.

In the third batch of experiment, the task was to evolve genetic programs using General (i) as the training set and then implement incremental GP to further evolve the genetic programs using Adaptive (i). This task first evolved general classifiers, and then, it adapted them to a particular subject. This experiment is called "3rd Adaptive" experiment.

The third batch of experiment (or the "3rd Adaptive" experiment) could also be used to test whether the method can adapt itself to the introduction of new or recent data of the same user, thus revealing the possibility of adaptability to the latest data of a single user.

6.2.3 Experiments Performed

We performed three separate experiments to investigate the method. The first experiment used 6 subjects to investigate the general feasibility of the method. The second experiment investigated stagnation effect using 3 subjects. The third experiment used 2 subjects but performed repeated simulated evolutions to clarify the robustness of the method [35].

6.3 Results and Discussions

6.2.3 Experiments on 6 Subjects

The experiment used 6 subjects from various demographics and nationalities; male and female subjects were included. Each subject required approximately 15 minutes for data acquisition.

From the data acquired, the application immediately extracted the features used; therefore, the stored data were not in the form of videos but in the form of features.

In all the experiments, the evolutions were performed by using the same budget of 30 sessions. The "1st Unique" experiment required 30 sessions of simulation of XGP (for each subject), and we selected the five best sessions for evaluation. The "2nd General" experiment also required 30 sessions, and we selected the five best programs for evaluation. The "3rd Adaptive" experiment consisted of two-phase experiments: 15 sessions to evolve the general classifiers, and 15 sessions to adapt the classifier evolved from the first phase to a specific user.

For evaluation, we tested the accuracy of the evolved program on the test data of the respective subjects. We used the voting system (majority), selecting the 5 best programs as voters to classify the emotions. The evaluation results differed from one subject to another, as shown in Table 18 below. The numbers are represented in percentage (%), rounded down.

Table 18. Accuracy results on test data

	1st Unique (%)	2nd General (%)	3rd Adaptive (%)
Subject1	88	34	**73**
Subject2	34	20	**27**
Subject3	57	42	**92**
Subject4	40	20	**69**
Subject5	25	42	**53**
Subject6	76	21	**56**
Average	53	29	***62***

From Table 18 we could conclude that, in terms of the average value, the evolution from a general classifier or the “3rd Adapt” experiment yields better results than the other two experiments. The usage of incremental GP also improves the accuracy with test data; the 1st unique averaged 53.3 (s=24.8), the 2nd General averaged 29.8 (s=10.8.), and the 3rd Adaptive averaged 61.7 (s=21.9) (one-way ANOVA, $p< 0.05$).

However, in several scenarios, the system could not yield a satisfactory result, especially with Subject2. After retracing the environment of the experiment, we found that during the data acquisition of Subject2, the movement of the subject was very erratic, and therefore, data acquisition was repeated. We investigated the scenario and found that the Kinect required some time to track the user when the face was outside the screen capture area; further, during the capture of the entire data for Subject2, the Kinect was located in front of the monitor. We concluded that the distance at which the Kinect is placed is important.

On an average, the total number of generations (and thus, the time) required to evolve the program with "3rd Evolve" experiments is less than the total number of generations required with the "2nd General" experiments (thus implying that the 3rd method is faster, in general). The average value of the total number of generations required to evolve the classifier is shown in Table 19 below.

Table 19. Total number of generations (average)

Experiments	# of Generations (Average of Total)
1st Unique	2978.59
2nd General	3768
3rd Adaptive	3676.47

The data in Table 19 are obtained by the following methods:

• For the "1st Unique" experiment, we calculated the average of all the total number of generations (from 30 independent sessions of each subject), to represent the total number of generations needed using the entire budget, average of all 6 subjects.

• For the "2nd General" experiment, we added the number of generations from 30 independent sessions to represent the total number of generations needed using the entire budget.

• For the "3rd Adaptive" experiment, we calculated the average of the sum of the number of generations for the 15 independent sessions of the first phase and the number of generations for the 15 independent sessions of the second phase.

The number of generations required for evolving a unique classifier using "1st Unique" is significantly less than the corresponding number for other methods owing to several factors. The first factor is the smaller amount of data whose similarity should be higher; the data are for a single person only (unlike the other two experiments in which some parts of the training used data from several people who express emotions differently). Thus, in this method, it should be faster to reach termination.

The second factor is that the stagnation period for all the experiments was the same; this period is 32 generations, and it might influence an unfair comparison with incremental GP. Incremental GP could have been faster if the termination criteria of the phases of “3rd Adapt” used a lower stagnation number (the dual phase of “3rd Adapt” could have resulted in stagnation being reached twice, or in double the number of generations on stagnation). As a theoretical prediction, by removing the unnecessary 32 stagnated generations (total of 32 x 15 = 480 generations), the values could have been approximately 2978 (1st Unique) and 3197 (3rd Adapt), which are considerably close.

6.2.3 Investigating the Stagnation

As shown at previous subsection 6.2.3, we predicted that the long stagnation period is hindering the computational effort of the method, without giving additional computational performance. Therefore, we conducted an experiment to investigate the effect of changing stagnation period of the method.

In order to investigate stagnation, we performed another experiment with XGP batches to compare 32 stagnated generations with 16 stagnated generations.

For this experiment, we used 3 subjects; the aim was to compare a long (32 stagnated generations) evolution with a short (16 stagnated generations) evolution. The data of the subjects are labeled as *General*, *Adaptive*, and *Test*. *Test* will be used only for the test data set. *General* and *Adaptive*, as in the experiment explained in section 5.2, will be used for training (either combined or incrementally).

We created the following batches:

- 30 sessions of Long, trained with *General* and *Adaptive*. This batch is labeled as S(x)-All.
- 20 sessions of Short—trained with *General*— followed (incremental) by training using *Adaptive* (used 5 best programs as seed), and vice-versa, in order to determine whether the method works irrespective of the order of incremental GP. These batches are labeled as S(x)-Inc and S(x)-Inc2.
- 30 sessions of Long, trained with all *General* and all *Adaptive*. This batch is labeled as All-[base].
- 20 sessions of Short, trained with all *General*, followed (incremental) by training using *Adaptive* (used 5 best programs as seed). These batches are labeled as All+S(x).

The results—in terms of the evolution time and the precision (accuracy in percentage) for the *Test* data set—are shown in Fig. 20 and Fig. 21 below.

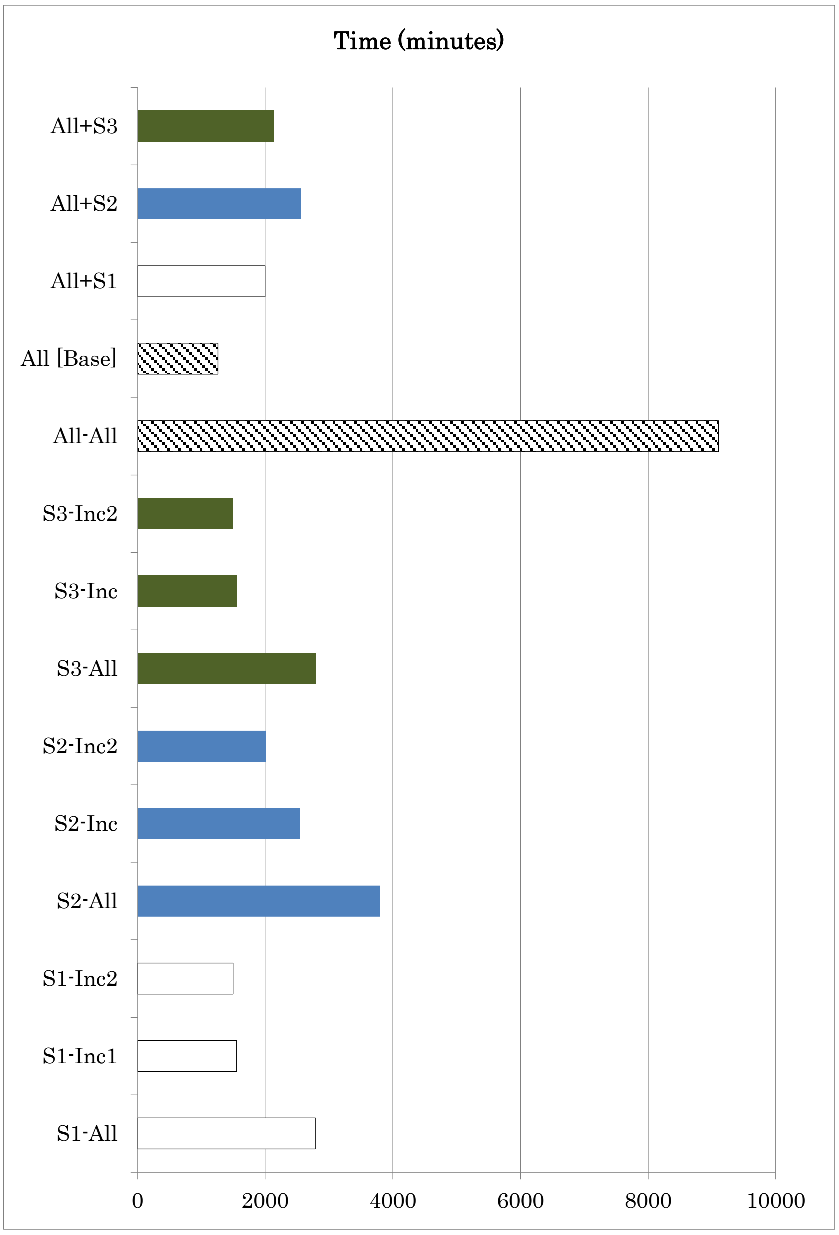


Figure 20. Comparison on time needed to simulate evolution

From Fig. 20 we observe that the total time required for a generic long evolution is much greater than the time required for any other evolution. Further, incremental short evolutions are always faster than the others.

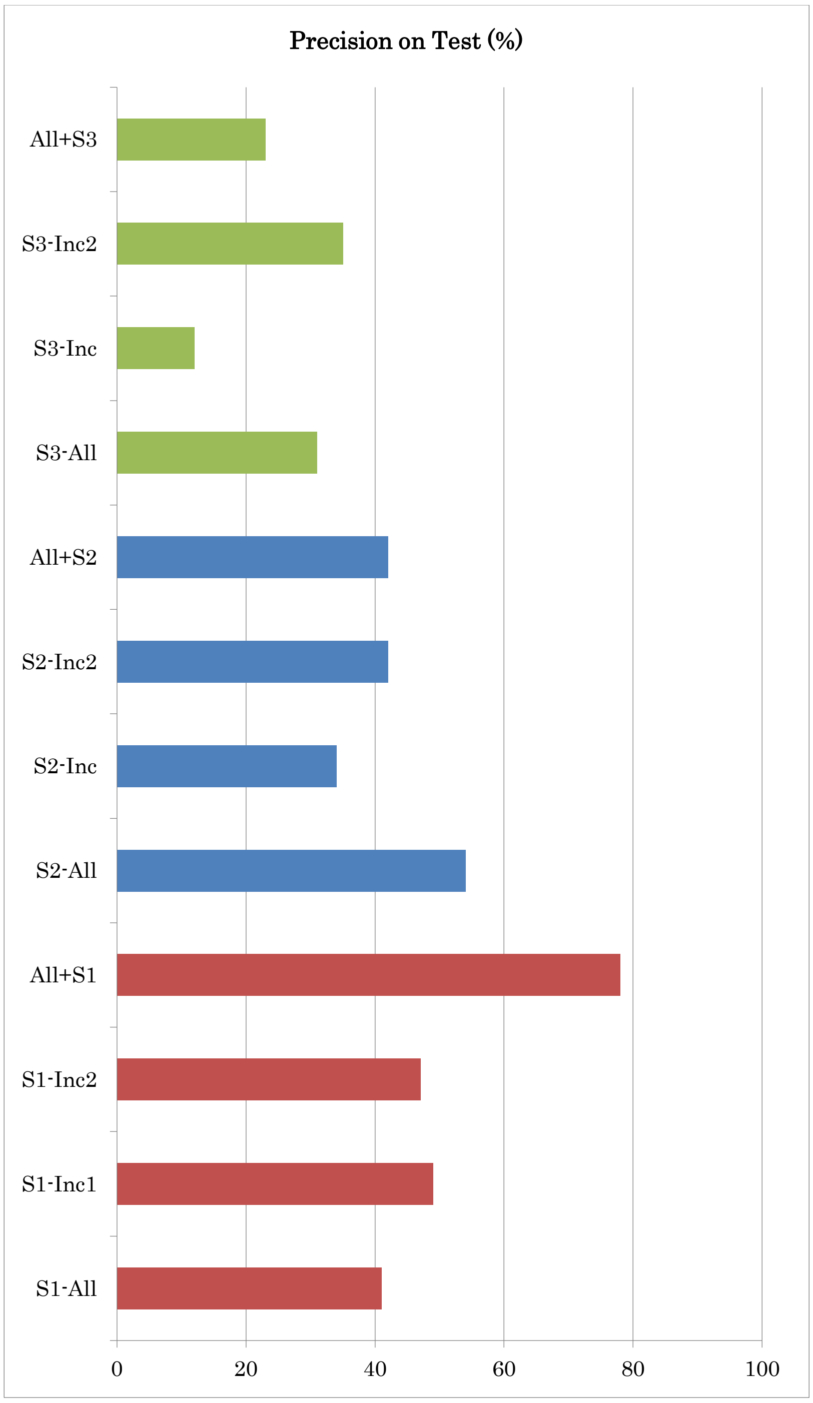


Figure 21. Accuracy on test data

Fig. 21 above shows that the accuracy of short-short evolution, when compared with the accuracy of long stagnation, depends on the subject; the results obtained by calculating an average of the results are shown in Table 20 below.

Table 20. Mean of accuracy

	Average (%)
Sx-All	42.00
Sx-Inc1	31.67
Sx-Inc2	41.33
All+Sx	47.67

We used one-way analysis of variance (ANOVA) [36]for the evolution time for each method (null hypothesis is they are similar, a=0.05), and we obtained the p-value of 0.02, thus rejecting the null hypothesis. However, similar analysis on accuracy suggested that there is no significant difference between short-short or long evolution.

6.2.3 Investigating the Robustness

We performed additional repetition of experiments to investigate the decrease the stagnation period in order to reduce the possibility of over-fitting, and performing the experiments multiple times on 2 subjects to investigate the robustness of the method.

Using the data from the 2 subjects, we run several XGP sessions each, using the 3rd Adapt approach explained in the previous section 5.2; we set the stagnation period to 16. In addition to analyzing the average accuracy, we investigated the accuracy of each emotion.

We run 4 batches (4x20 sessions) of XGP using Training (i), thus yielding 4 different sets of genetic programs. Then, we proceed to evolve each set 2 times using Incremental GP, thus obtaining 8 different sets of genetic programs for each subject.

From each set of genetic programs, we selected the 5 best genetic programs for evaluation on the test data set.

The results of the experiment (accuracy on test data, in percentage, with means and standard deviation at the bottom of the table) are shown in Table 21 and Table 22. Fig. 22 and Fig. 23 show the graphical representations of the average and standard deviations for the test results of each subject.

Table 21. Test results for Subject1

	Happy%	Relaxed%	Angry%	Sad%	Total%
1a	84	45	8	25	41
1b	84	63	8	25	45
2a	92	54	0	25	43
2b	84	72	0	25	45
3a	100	54	33	33	56
3b	92	54	0	25	43
4a	61	54	8	16	35
4b	84	81	33	16	54
mean	*85.29*	*56.57*	*8.14*	*24.86*	*44.00*
s	*11.36*	*11.72*	*13.93*	*5.52*	*6.82*

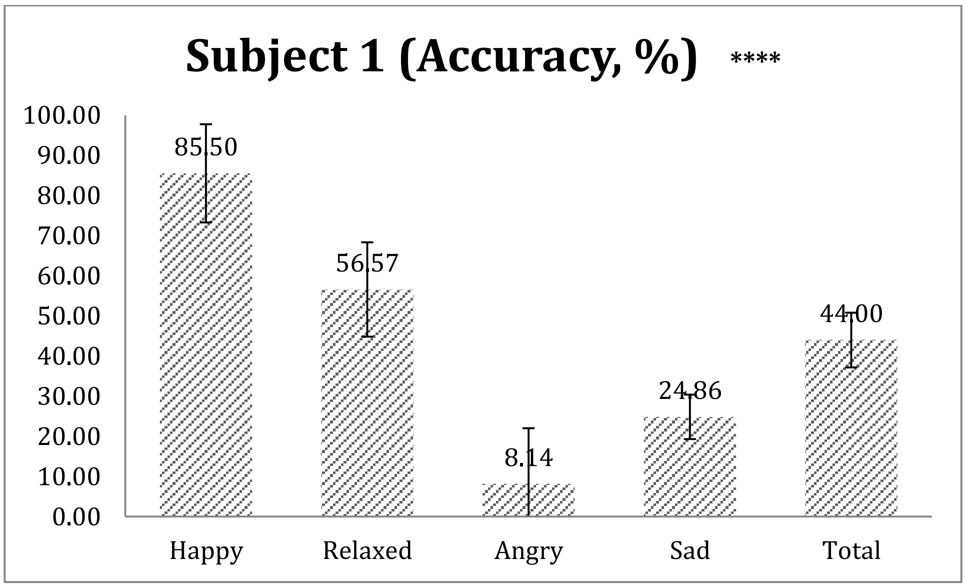


Figure 22. Stem height (mean ± s) of Subject1, p<0.001

Table 22. Test results on Subject2

	Happy%	Relaxed%	Angry%	Sad%	Total%
1a	95	26	100	80	75
1b	95	20	100	80	73
2a	95	0	80	73	63
2b	100	0	80	73	65
3a	90	20	80	73	67
3b	90	6	70	66	60
4a	95	0	70	73	62
4b	95	0	60	73	60
mean	*94.29*	*10.29*	*82.86*	*74.00*	*66.43*
s	*3.20*	*11.11*	*14.14*	*4.49*	*5.71*

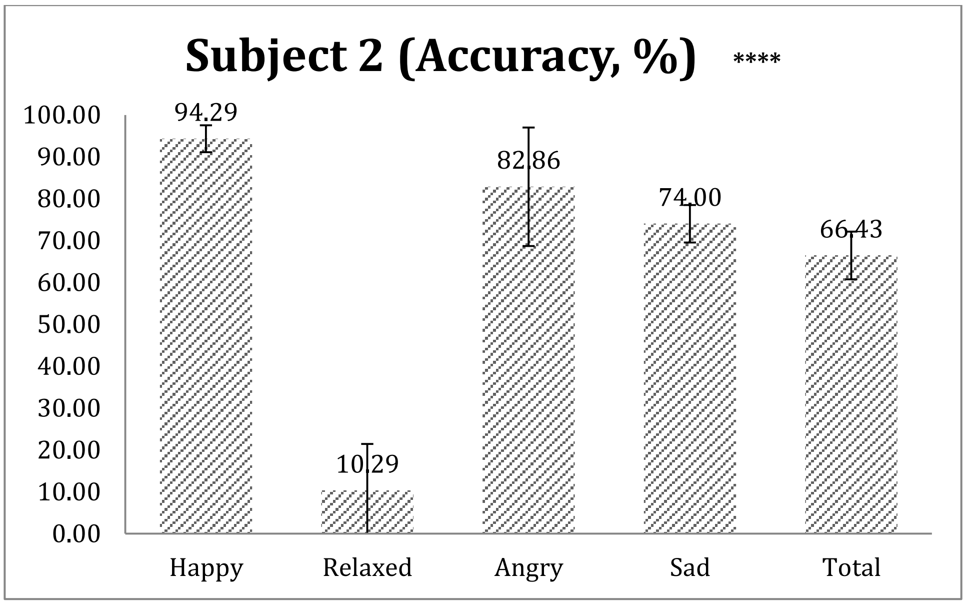


Figure 23. Stem height (mean ± s) of Subject2, p<0.001

From those tables and figures, we observe that there is relatively low variance in the test results. Subject 1 and Subject 2 show standard deviations of approximately 6.82 and 5.71, respectively. However, we observe that although the *happy* emotion can be detected well for both subjects, the methods generate varied results for the detection of the other emotions. The evolution succeeded in evolving a program to recognize *happy* and *relaxed* for Subject 1; however, the evolution did not perform well in detecting the *relaxed* emotion for Subject 2.

The reasons for this result could be the following: the data used to train the classifier may not have been adequately representative, the number of interactions was low (only 2), and the subject may not have selected an appropriate emotion.

We used ANOVA to analyze how significant the difference between emotions is, and the p-value for Subject1 and Subject2 are $1.94e^{-13}$ and $1.5e^{-16}$, respectively.

We also used ANOVA to further check the consistency of experiment repetition, and found that the p-value for Subject1 is 0.97, p-value for Subject2 is 0.98, and p-value for all experiments is 0.76; thus we can conclude that the experiments have similar or consistent result. This suggests that the method is robust.

Another point of interest is that the data used for Subject 2 were obtained from a completely different setup (using the setups shown in Fig. 11 and Fig. 12 from Section 3.4.1), whereas Subject 1 used a single setup (Fig. 11 from Section 3.4.1). This finding indicates that the setup of the Kinect did not significantly affect the results. The results may be affected by another important factor: the subjects can independently select the emotion they think they are experiencing; therefore, the information provided for each emotion may have differed significantly (e.g., a subject may have indicated 4 minutes of the *happy* emotion but only 1 minute of the *angry* emotion because many subjects indicated their reluctance to express anger during the previous experiment). This tendency could have influenced the evolution because the training data for the *angry* emotion were scarce when compared with the other emotions, whereas the training data for the *happy* emotion may have been abundant.

6.5 Conclusions

We investigated the implementation of incremental genetic programming (GP) in the evolution of a user-specific emotion recognition model. We performed and compared 3 experiments: (1) evolving the unique classifier for a specific user, (2) evolving the general classifier, and (3) evolving incremental GP for a general classifier, and then adapting it for a specific user.

The experimental results showed that on an average, incremental GP not only yields better accuracy for test subjects, but also achieves faster evolution when compared with the general classifier.

From the results, we discovered a new area for further exploration—the comparison and analysis of different paradigms in investigating the time required to evolve a user-specific classifier from the perspective of the developer and from the perspective of the end-user.

We also investigated the robustness of the 3rd Adaptive method by repeating the experiments several times to obtain statistical data; we demonstrate that the method yields similar results with relatively low variance, especially for the *happy* emotion.

Chapter 7

Miscellaneous Emotion Analysis

Previous chapters focused on improving the model for a better computational performance and/or computational effort, especially by exploring the evolutionary computation. This chapter aims for performing miscellaneous emotion analysis using the developed emotion recognition model. Several comparisons performed to make small investigations on effect of different time of the day in acquiring user's data, and on emotions that tend to be easier to recognize.

7.1 Comparing data acquisition time

The first small experiment we performed was to investigate whether time of acquisition can affect the data acquired [37]. In order to investigate this matter, we acquired user's data divided into 3 categories based on time of acquisition: *morning*, *afternoon*, and *late* afternoon. Using these 3 data we evolved different programs using XGP with similar methods from previous chapters, to and evaluate the performances on each other data.

We used several experiment terminologies on every experiments performed.

- *Genotype*: a single gene that formed the program (function).
- *Individual*: a single program resulting from XGP runs, in the form of XML, consisting of information regarding the tree structure that represents the program.

- *Generation*: One generation of genetic programming runs. In our experiments, we use 100 individuals per generation.
- *Sessions*: One set of XGP runs until termination criterion is reached. The termination criterion is one of the following: number of generations (100 is the maximum), fitness value (2, or near perfect match), or stagnation period (30 generations of stagnation).
- *Batch*: One batch of XGP is a number of sessions with the same purposes, goals, and settings, in order to achieve statistical data that can show the computational performance and robustness of the system.
- *Experiment*: One experiment consists of several batches (each with different setting) to be compared to investigate the strengths and weaknesses of each method. In the current experiment, the different setting is the data set used for training and testing.

The experiments performed three acquisitions of data from a single subject. The subject was required to imagine a situation that would make him show a specific emotion, and then the Kinect captured a video (and extracted the data) for around 1 minute per emotion.

The first, second, and third acquisitions were conducted in the morning (around 9 AM), afternoon (around 1 PM), and late afternoon (around 5 PM). The data acquired are labeled as Morning, Afternoon, and Late, respectively.

We ran four batches for each data set (12 batches in total). Each batch consisted of 15 sessions. From each batch, the five best individuals (in terms of Fitness Function) were selected. They were then used as classifiers with the implementation of voting (majority selection). [38].

7.2 Results and Discussions on Data Acquisition Time

The basic results are shown at the following Table 4, Table 5, and Table 6, showing accuracy on data set of Morning, Afternoon, and Late, respectively. Shaded parts of the table shows where the Data set used for testing is the same with the data set used for training (or evolving classifiers).

Table 23. Test accuracy result on *morning* data set

Trained	Happy%	Relaxed%	Angry%	Sad%
Mor1	100.00	100.00	100.00	93.00
Mor2	100.00	92.00	100.00	93.00
Mor3	100.00	92.00	100.00	93.00
Mor4	93.00	100.00	100.00	100.00
>> Avg	98.25	96.00	100.00	94.75
>> Std	3.50	4.62	0.00	3.50
Aft1	60.00	85.00	16.00	75.00
Aft2	86.00	50.00	75.00	93.00
Aft3	93.00	85.00	75.00	56.00
Aft4	93.00	85.00	91.00	75.00
>> Avg	83.00	76.25	64.25	74.75
>> Std	15.68	17.50	33.04	15.11
Lat1	93.00	35.00	91.00	56.00
Lat2	86.00	14.00	83.00	75.00
Lat3	93.00	28.00	100.00	75.00
Lat4	80.00	35.00	91.00	68.00
>> Avg	88.00	28.00	91.25	68.50
>> Std	6.27	9.90	6.95	8.96

Table 24. Test accuracy result on *afternoon* data set

Trained	Happy%	Relaxed%	Angry%	Sad%
Mor1	92.00	78.00	63.00	92.00
Mor2	85.00	64.00	63.00	100.00
Mor3	85.00	64.00	100.00	85.00
Mor4	85.00	71.00	81.00	100.00
>> Avg	86.75	69.25	76.75	94.25
>> Std	3.50	6.70	17.67	7.23
Aft1	93.00	100.00	100.00	100.00
Aft2	85.00	100.00	100.00	100.00
Aft3	92.00	100.00	100.00	100.00
Aft4	92.00	100.00	100.00	100.00
>> Avg	90.50	100.00	100.00	100.00
>> Std	3.70	0.00	0.00	0.00
Lat1	92.00	57.00	90.00	64.00
Lat2	85.00	64.00	90.00	85.00
Lat3	92.00	64.00	81.00	78.00
Lat4	85.00	78.00	100.00	92.00
>> Avg	88.50	65.75	90.25	79.75
>> Std	4.04	8.81	7.76	11.95

Table 25. Test accuracy result on *late* data set

Trained	Happy%	Relaxed%	Angry%	Sad%
Mor1	90.00	92.00	75.00	41.00
Mor2	100.00	53.00	83.00	25.00
Mor3	90.00	69.00	91.00	58.00
Mor4	100.00	61.00	91.00	58.00
>> Avg	95.00	68.75	85.00	45.50
>> Std	5.77	16.82	7.66	15.84
Aft1	100.00	84.00	33.00	41.00
Aft2	100.00	76.00	91.00	33.00
Aft3	100.00	76.00	75.00	33.00
Aft4	100.00	69.00	91.00	41.00
>> Avg	100.00	76.25	72.50	37.00
>> Std	0.00	6.13	27.39	4.62
Lat1	100.00	100.00	91.00	83.00
Lat2	100.00	100.00	91.00	75.00
Lat3	100.00	92.00	100.00	83.00
Lat4	100.00	100.00	100.00	93.00
>> Avg	100.00	98.00	95.50	83.50
>> Std	0.00	4.00	5.20	7.37

To refine the tables above, we calculated the average (mean) of each batch. Tables 26, 27, and 28 below show the accuracy in data sets of Morning, Afternoon, and Late, respectively. Shaded columns show where the data sets used for testing and training (or evolving classifiers) were the same.

Table 26. Test accuracy result on *morning* data set

Test Data	Happy%	Relaxed%	Angry%	Sad%	All%
Morning	98.25	96.00	100.00	94.75	97.00
Afternoon	86.75	69.25	76.75	94.25	82.25
Late	95.00	68.75	85.00	45.50	72.25
mean	*93.33*	*78.00*	*87.25*	*78.17*	***83.83***
s	*5.93*	*15.59*	*11.79*	*28.29*	*12.45*

Table 27. Test accuracy result on *afternoon* data set

Test Data	Happy%	Relaxed%	Angry%	Sad%	All%
Morning	83.00	76.25	64.25	74.75	76.00
Afternoon	90.50	100.00	100.00	100.00	97.50
Late	100.00	76.25	72.50	37.00	70.25
mean	*91.17*	*84.17*	*78.92*	*70.58*	***81.25***
s	*8.52*	*13.71*	*18.72*	*31.71*	*14.36*

Table 28. Test accuracy result on *late* data set

Test Data	Happy%	Relaxed%	Angry%	Sad%	All%
Morning	88.00	28.00	91.25	68.50	68.25
Afternoon	88.50	65.75	90.25	79.75	80.75
Late	100.00	98.00	95.50	83.50	93.00
mean	*92.17*	*63.92*	*92.33*	*77.25*	***80.67***
s	*6.79*	*35.04*	*2.79*	*7.81*	*12.38*

From the tables, we can see that the average results for all emotions do not significantly differ; averages for the *Morning*, *Afternoon*, and *Late* data sets are 83.83%, 81.25%, and 80.67%, respectively. However, there are several major inaccuracies, especially when *Late* data was used for training and Morning data for testing, especially for recognizing when the subject was relaxed.

The results suggested that there is no significant differences in general performances between taking the data in the morning, in the afternoon, or in late afternoon; however, there are significant differences in that some emotions are not recognized well if we acquire data of a specific emotion at a specific time.

There might be several possibilities regarding this result. First, the data might have been incomplete. In some cases, the data might also have been insufficient, thus the evolved programs actually may not have learned enough.

Another possibility is that the stress levels (valence) and the arousal levels are dynamic and might have influenced the subject of the experiment to express his emotions differently; his expressions may have been influenced by his ‘real’ feelings (such as tiredness in the late afternoon, or sleepiness in the morning) at the moment of data acquisition. This can be examined by acquiring more data on different days, which can be done in separate experiments.

To illustrate the differences between the results, Fig. 24, Fig. 25, and Fig. 26 show graphs of accuracy using the training data of *Morning*, *Afternoon*, and *Late*, respectively.

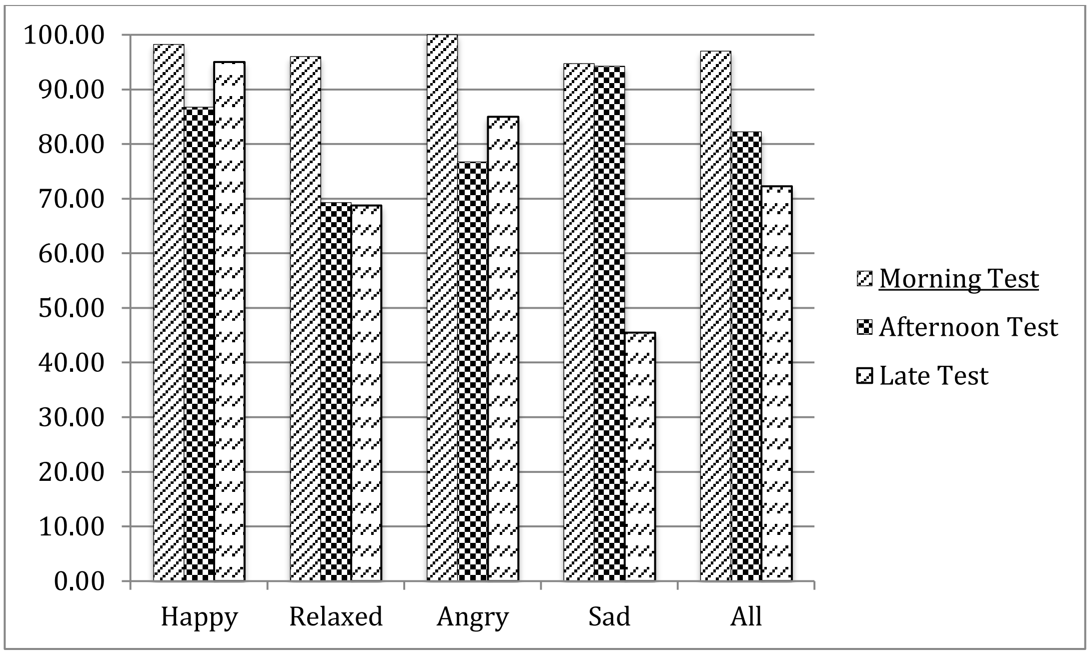


Figure 24. Test accuracy result trained by morning data

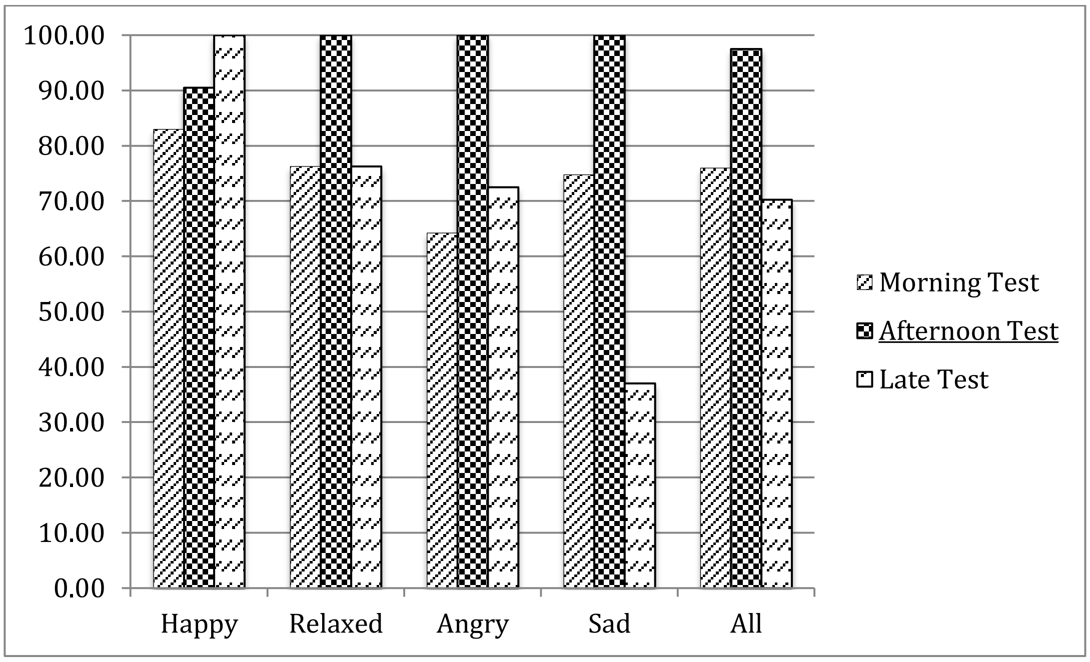


Figure 25. Test accuracy result trained by afternoon data

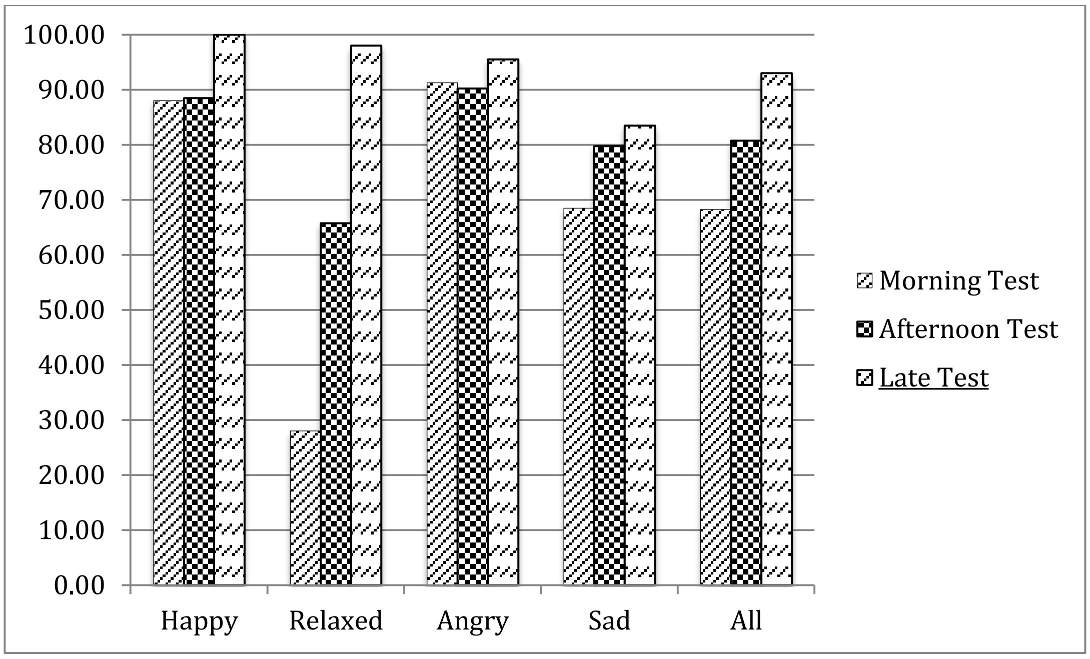


Figure 26. Test accuracy result trained by late data

7.3 Comparing Emotions Captured

In comparing the emotions captured, we used the information gained from the experiments on Chapter 7.1 and Chapter 6.2. First, we will analyze the results from the experiments conducted at Chapter 7.1.

7.3.1 Analyzing Experimental Results from Chapter 7.1

By re-arranging the table, we can obtain a better representation of the results, so we can easily see which emotions are easier to recognize. Table 29 shows the averages and standard deviations of all tests from the experiment.

Table 29. Average (mean) and standard deviation (s) of each emotion

Emotion	*Mean (%)*	*Standard Deviation*
Happy	92.22	6.27
Relaxed	75.36	22.26
Angry	86.17	12.60
Sad	75.33	21.90

The experimental results shown in Fig. 27 suggest that happiness is the easiest emotion to recognize; it has not only a better average accuracy in tests but also robust and stable results (a low standard deviation). Relaxation and sadness are not only difficult to recognize but also have very volatile results (high standard deviations).

These results also suggested that high arousal emotions (Happy and Angry) might be easier to detect, while low arousal emotions (Sad and Relaxed) might be more difficult to detect.

The reason behind this might be related to the tendency of people to try to hide their sadness (as sadness can be viewed as weakness), and that people might express anger or happiness with the intention of manipulating others (such as showing aggressiveness). These behaviors are quite similar to several phenomena in the natural world such as [Handicap principle].

Another possibility that might be related to the results is that sadness and depression are difficult to recognize, since people are becoming more adept at hiding and not showing signs of them to the world.

However, a more thorough analysis from the point of view of psychology might be needed.

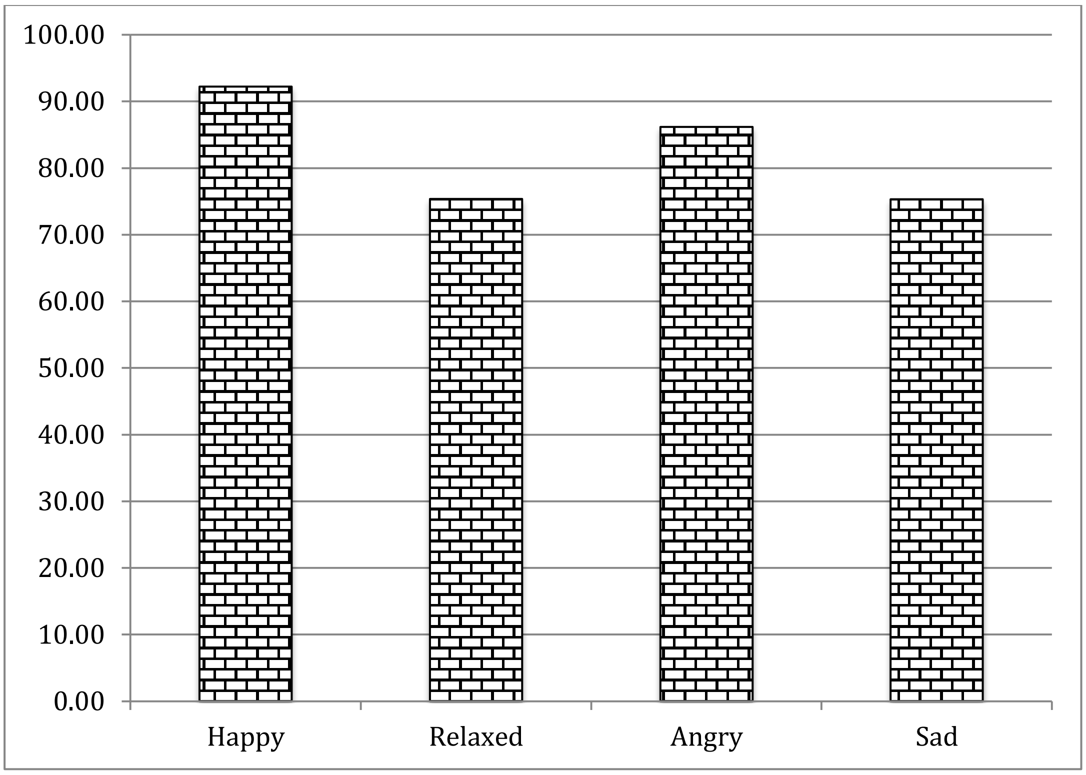


Figure 27. Graph of average (mean) of test accuracy results

7.3.2 Analyzing Results from Chapter 6.2

On this part we will analyze the results presented on Chapter 6.2, regarding the emotion analysis of two people. For clarity purposes, we would like to put the figure representing the results from experiment explained on section 6.2 below.

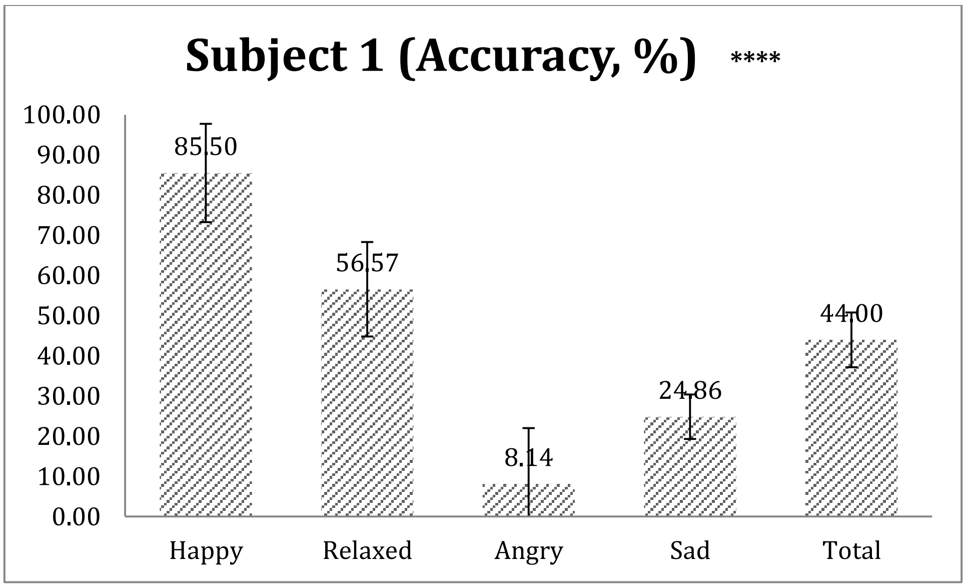


Figure 28. Stem height (mean ± s) of Subject1, p<0.001 (ch 6.2)

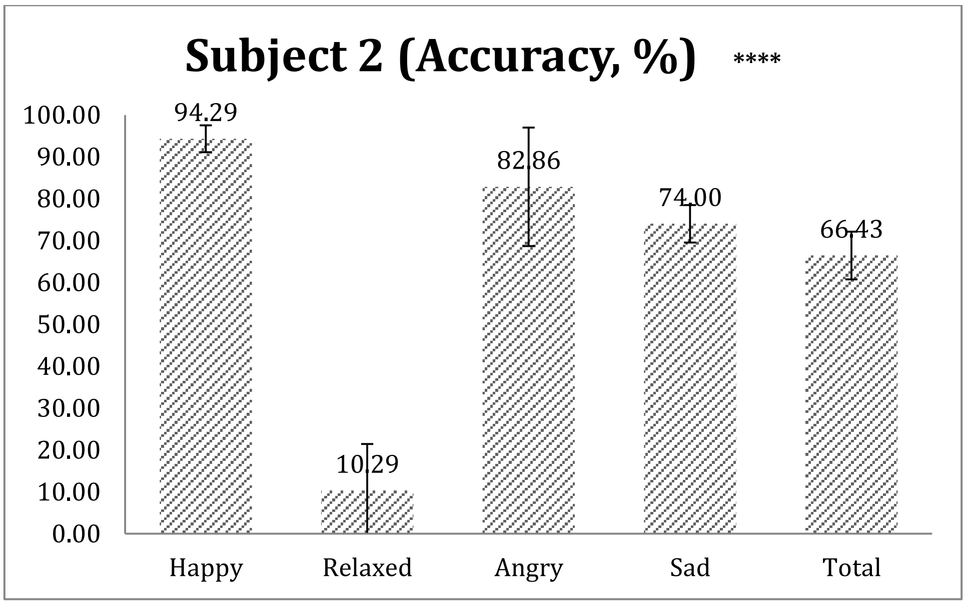


Figure 29. Stem height (mean ± s) of Subject2, p<0.001 (ch 6.2)

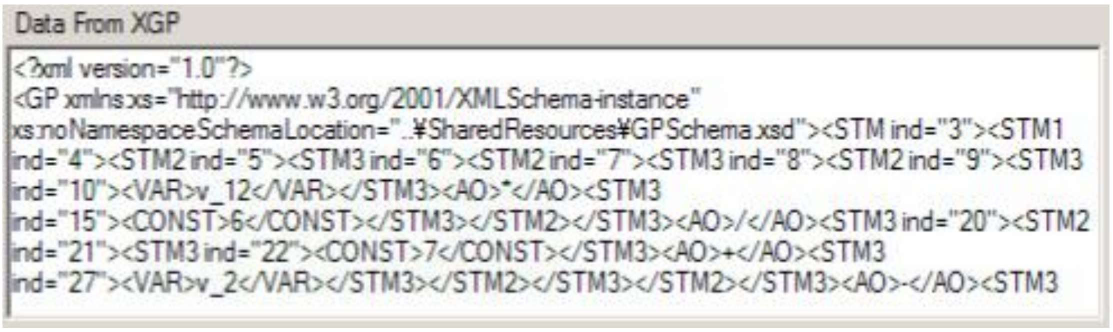
From section 6.2.3, we analyzed that although the *happy* emotion can be detected well for both subjects, the methods generate varied results for the detection of the other emotions. The evolution succeeded in evolving a program to recognize *happy* and *relaxed* for Subject 1; however, the evolution did not perform well in detecting the *relaxed* emotion for Subject 2.

As stated previously, this could possibly be caused by the lack of variety in the training data to represent the emotions other than happiness, which might be also caused by the difficulty of the subjects to express anger. A quick verbal survey was conducted, asking each subject about ‘what emotion was more difficult to express and which took the least time?’, all of the subjects replied ‘anger’ as the most difficult expressed, with one subject claimed to have expressed anger exaggeratedly due to unsure about invoking anger.

7.7 XML Representation of Programs Evolved by XGP

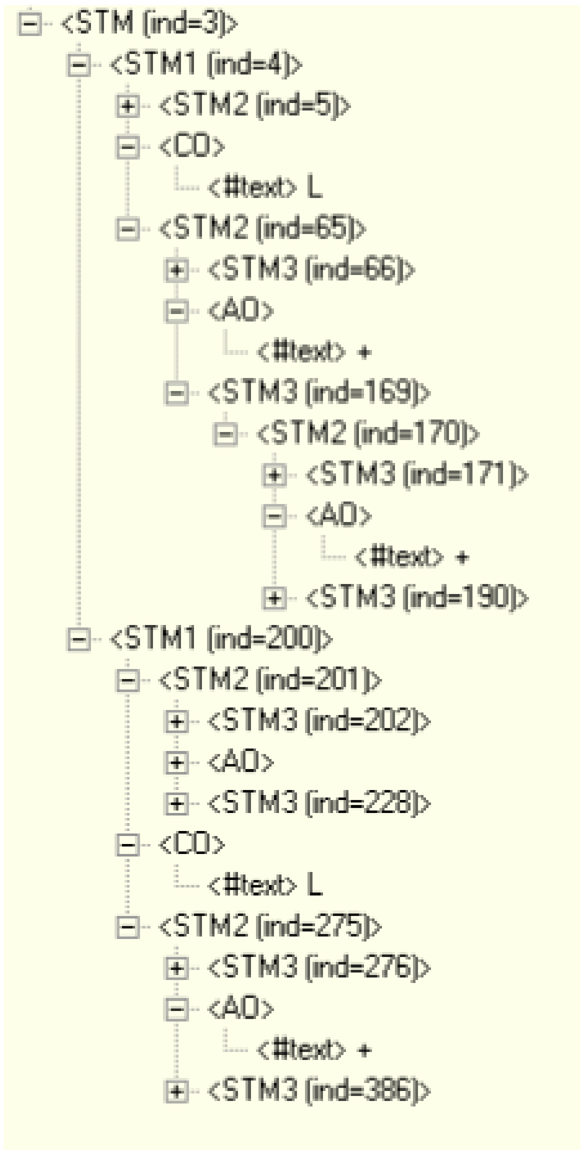
One of the reasons why we chose to use genetic programming to conduct simulated evolution is because the XML representation of the program can be analyzed. The program evolved by the XGP is a function that incorporates several possible terminals, such as arithmetic operations and data of extracted features.

Fig. 30 and Fig. 31 below shows an example of XML Program evolved by XGP in txt format and tree representation, respectively.



```
<?xml version="1.0"?>
<GP xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="..¥SharedResources¥GPSchema.xsd"><STM ind="3"><STM1
ind="4"><STM2 ind="5"><STM3 ind="6"><STM2 ind="7"><STM3 ind="8"><STM2 ind="9"><STM3
ind="10"><VAR>v_12</VAR></STM3><AO>*</AO><STM3
ind="15"><CONST>6</CONST></STM3></STM2></STM3><AO>/</AO><STM3 ind="20"><STM2
ind="21"><STM3 ind="22"><CONST>7</CONST></STM3><AO>+</AO><STM3
ind="27"><VAR>v_2</VAR></STM3></STM2></STM3></STM2></STM3><AO>-</AO><STM3
```

Figure 30. Sample of Genetic Program evolved by XGP, in the form of XML file, that can be analyzed to see which features are important and which are not from a user, in case further emotional analysis is needed.

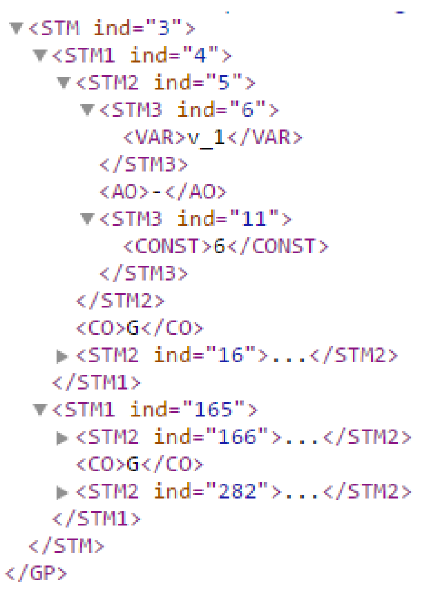


```
<STM [ind=3]>
  <STM1 [ind=4]>
    <STM2 [ind=5]>
    <CO>
      <#text> L
    <STM2 [ind=65]>
      <STM3 [ind=66]>
      <AO>
        <#text> +
      <STM3 [ind=169]>
        <STM2 [ind=170]>
          <STM3 [ind=171]>
          <AO>
            <#text> +
          <STM3 [ind=190]>
  <STM1 [ind=200]>
    <STM2 [ind=201]>
      <STM3 [ind=202]>
      <AO>
      <STM3 [ind=228]>
    <CO>
      <#text> L
    <STM2 [ind=275]>
      <STM3 [ind=276]>
      <AO>
        <#text> +
      <STM3 [ind=386]>
```

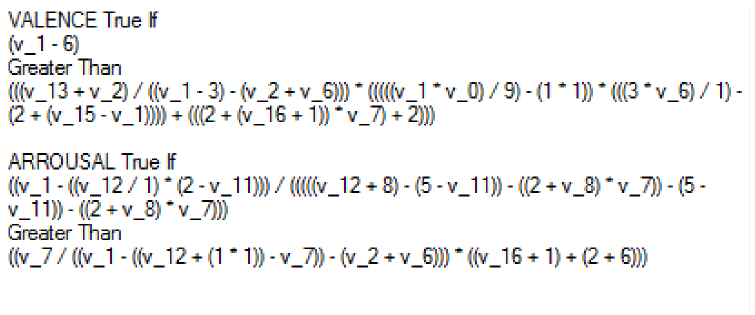
Figure 31. The tree representation of the same XML file from Fig 30 above.

By observing the XML file, analysis on which features are important for a specific user can be performed; in example by spreading the trees and mark the repeatedly occurring feature data (represented with v_[number] in the XML coding as explained at previous chapter 3: the first 9 (i.e. v_0 ~ v_8) represent average, the second 9 (i.e. v_9 ~ v_17) represent standard deviation, the last 9 (i.e. v_18 ~ v_26) represent power, of the nine AUs each), or whether some feature data are amplified.

Using the XML represented at Fig.30 and Fig. 31, combining with a simple analyzer we can see the tree structure of the program, as well as the mathematical function of each subtree as shown below.



```
▼<STM ind="3">
  ▼<STM1 ind="4">
    ▼<STM2 ind="5">
      ▼<STM3 ind="6">
          <VAR>v_1</VAR>
        </STM3>
        <AO>-</AO>
      ▼<STM3 ind="11">
          <CONST>6</CONST>
        </STM3>
      </STM2>
      <CO>G</CO>
    ▶<STM2 ind="16">...</STM2>
    </STM1>
  ▼<STM1 ind="165">
    ▶<STM2 ind="166">...</STM2>
      <CO>G</CO>
    ▶<STM2 ind="282">...</STM2>
    </STM1>
  </STM>
</GP>
```



```
VALENCE True If
(v_1 - 6)
Greater Than
(((v_13 + v_2) / ((v_1 - 3) - (v_2 + v_6))) * (((((v_1 * v_0) / 9) - (1 * 1)) * (((3 * v_6) / 1) -
(2 + (v_15 - v_1)))) + (((2 + (v_16 + 1)) * v_7) + 2)))

ARROUSAL True If
((v_1 - ((v_12 / 1) * (2 - v_11))) / (((((v_12 + 8) - (5 - v_11)) - ((2 + v_8) * v_7)) - (5 -
v_11)) - ((2 + v_8) * v_7)))
Greater Than
((v_7 / ((v_1 - ((v_12 + (1 * 1)) - v_7)) - (v_2 + v_6))) * ((v_16 + 1) + (2 + 6)))
```

$$v_1 - 6$$

$$\frac{v_1 - v_{12}\left(-v_{11} + 2\right)}{2v_{11} + v_{12} - 2v_7\left(v_8 + 2\right) - 2}$$

$$\frac{v_7\left(v_{16} + 9\right)}{v_1 - v_{12} - v_2 - v_6 + v_7 - 1}$$

$$\frac{1}{v_1 - v_2 - v_6 - 3}\left(v_{13} + v_2\right)\left(v_7\left(v_{16} + 3\right) + \left(\frac{v_0 v_1}{9} - 1\right)\left(v_1 - v_{15} + 3v_6 - 2\right) + 2\right)$$

Figure 32. Flow of analysis (part 1): Top is the XML representation; Middle is the logical explanation of tree structure; Last four are the functions from the relevant subtree of Arousal and Valence (true and false)

From Fig. 32 above, an easier-to-understand representation of the functions and the tree structure of the model is shown at Fig. 33 below.

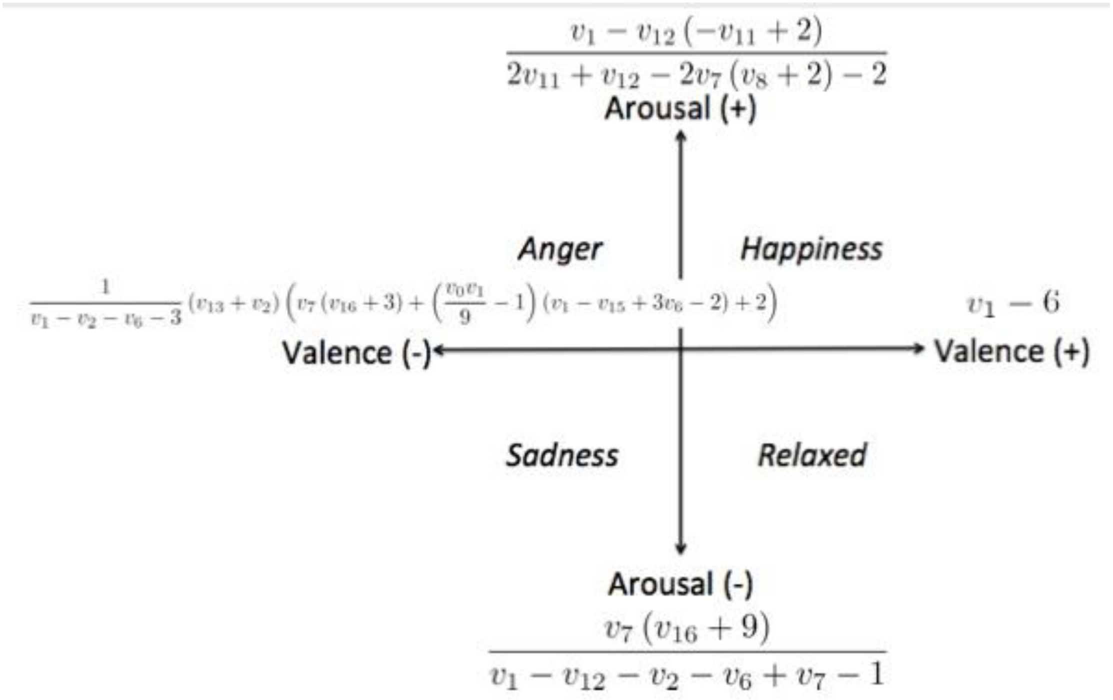

$\frac{v_1 - v_{12}(-v_{11}+2)}{2v_{11} + v_{12} - 2v_7(v_8+2) - 2}$
Arousal (+)
Anger
Happiness
$\frac{1}{v_1 - v_2 - v_6 - 3}(v_{13}+v_2)\left(v_7(v_{16}+3) + \left(\frac{v_0 v_1}{9} - 1\right)(v_1 - v_{15} + 3v_6 - 2) + 2\right)$
$v_1 - 6$
Valence (-)
Valence (+)
Sadness
Relaxed
Arousal (-)
$\frac{v_7(v_{16}+9)}{v_1 - v_{12} - v_2 - v_6 + v_7 - 1}$

Figure 33. Flow of Analysis (Part 2): The four mathematical formulas from Fig.32 are mapped on the Tree. Arousal (+) and Arousal (-) is compared to determine the Arousal logic value, and Valence for the Valence logic value. Emotion is determined by the value of both

After mapping the formula to the tree, the model determines the emotion based on Arousal and Valence logical value: Happy (positive arousal, positive valence), Sad (negative arousal, negative valence), Relaxed (negative arousal, positive valence), Angry (positive arousal, negative valence).

The fact that by reading tree representation of the XML we can perform analysis, has shown that genetic programming is not producing a black-box program, where analysis is very difficult, if not impossible, to perform.

7.6 Conclusions

Our research focuses on the evolution of an agent able to recognize emotions of a single user. Several approaches such as separating a tree structure for evolution, implementing a voting system, and evolving the voting system, showed improvements in accuracies and faster training time (which is required for a self-evolving agent).

During the development of an emotion recognition model, a question was raised regarding the best time of the day for recognizing emotion. Using available data, we performed an experiment to evolve several classifiers using different training data and cross-tested the classifiers using each set of data. The controlled variable is that each data set is taken at different times.

The results from the experiment suggested that although the time of data acquisition might not make average accuracy for all emotions significantly different (around 80% to 83%), the accuracy for each emotion might vary.

Further, our experimental results also suggested that high arousal emotions (Happy) are easier to recognize. This might be related to the psychological reasons behind those emotions; showing anger or happiness might be a means for people to manipulate other people, such as to threaten a foe, prepare to fight, or inform allies of a current situation. However, most subjects felt uneasy in expressing unpleasant emotions, and this might have influenced the results.

To express pleasant emotions such as happiness was easier for all subjects, and this might influence on how should the agent interact with the user during data acquisition period; in example, the agent could focus on detecting happiness, and clarify the user when the agent could not detect happiness [37] [34].

Regarding the difference in results of taking data for training, with the systems capable of evolving using new user’s data, the classifier can be updated as the user see fit, through interaction with the system.

Chapter 8

Conclusions

The present thesis focused on tackling several challenges in designing an emotion recognition model that can be used for real-life applications: temporal information, pervasiveness, unique-general user, and the black-box of many artificial intelligent approaches.

Chapter 3 presented the rough model proposed, and showed that the model can work at real-life applications by incorporating temporal information, and by using pervasive (and off-the-self and multi-purpose) Microsoft Kinect as a sensor. This chapter also showed several literary and experimental evidences that: taking a user-specific approach is better. Also experiments conducted suggested the genetic programming (GP) can be used to perform simulated evolution of programs that can act as classifiers.

Chapter 4 focused on enhancing the model by tackling GP's non-determinism problem, by applying collaborative filtering in the form of majority voting. The experimental results showed that implementing majority voting improves the accuracy and reduced the effect of non-determinism of GP, as well as possibly reducing over-fitting problems.

Chapter 5 focused on exploring the evolutionary computation to adapt new user's data. We conducted experiments on implementing *weighted trust* for voters instead of static voters, and the result was a trade-off between computational performances and computational efforts.

Chapter 6 explored the evolutionary computation on different direction, which is incremental genetic programming. The results showed that performing incremental GP has better results at computational performances and computational efforts, as well as easier incorporating new user's data to the model.

Chapter 7 analyzed the emotion analysis using the model, and also to show that the programs created by the simulated evolution of our model is not a black-box and can be analyzed.

To sum up, the proposed model is: (1) incorporating temporal information into analysis, in the form of mean, variance, and the power (square) of each AU signals in a timeframe, (2) using an off-the-shelf and multi-purpose Microsoft Kinect as a pervasive sensor, and operating in a pervasive environment, (3) capable to adapt to new user's data, and focusing on a single user instead of relying on generalization, (4) shown to be not a black-box, as the component of each AU can be analyzed if necessary.

Bibliography

[1] Rosalind W Picard, "Affective Computing," *MIT Media Laboratory Perceptual Computing Section Technical Report*, no. 321, 1995.

[2] K. Takahashi, "Remarks on Computational Emotion Recognition and Expression," in *6th International Symposium on Image and Signal Processing and Analysis*, 2009.

[3] Alex Pentland, *Honest Signals*.: MIT Press, 2008.

[4] et al. Z. Zeng, "A Surfey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions," in *ICMI 07*, Nagoya, Japan, 2006, pp. 126-133.

[5] Todd Rose, *The End of Average*.: Harper Collins, 2016.

[6] J.F. Cohn, "Foundations of Human Computing: Facial Expression and Emotion," in *International Conference on Multimodal Interfaces*, 2006, pp. 233-238.

[8] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, USA: Consulting Psychologists Press, 1978.

[7] Jianhua Tao and Tieniu Tan, "Affective Computing: A Review," *Affective Computing and Intelligent Interaction. LNCS 3784*, pp. 981–995, 2005.

[9] J. Ahlberg, "CANDIDE 3 - An Updated Parameterized Face," *Report no. LiTH-ISY-R-2326*, 2001.

[10] C.A., Lazarus, R.S. Smith, *Emotion and Adaptation*.: Guildford Press, 1990.

[11] K.R. Scherer, "Appraisal Theory," in *Handbook of Cognition and Emotion*.: Wiley, 1999, pp. 637-663.

[12] K. R. Scherer, "Emotion as a multicomponent process: A model and some cross-cultural data," *Review of Personality and Social Psychology*, vol. 5, pp. 37-63, 1984.

[13] L., Barkow, J., Tooby, J. Cosmides, *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*.: Oxford University Press, 1992.

[14] J.A. Russel, "A Circumplex Model of Affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161-1178, 1980.

[15] H Lövheim, "A new three-dimensional model for emotions and monoamine neurotransmitters," *Med Hypotheses*, vol. 78, pp. 341-348, 2011.

[16] T., Hoffmeister, F., Schwefel, H. Bäck, "A survey of evolution strategies," in *Proceedings of the Fourth International Conference on Genetic Algorithm*, 1991, pp. 2-9.

[18] Holland. J.H., *Adaptation in Natural and Artificial Systems*.: University of Michigan Press, 1975.

[17] L. J., Owens, A.J., Walsh, M.J. Fogel, *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley, 1966.

[19] M. Minsky, *Perceptrons: An Introduction to Computational Geometry*.: MIT Press, 1969.

[20] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[21] J.R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.

[22] Paul Benjamin Lowry, Rayman Meservy, Dan McDonald James Hansen, "Genetic programming for prevention of cyberterrorism through dynamic

and evolving intrusion detection," *Decision Support System*, vol. 43, no. 4, pp. 1362-1374, 2007.

[23] Michael O'Neill Anthony Brabazon, *Biologically inspired algorithms for financial modelling*. Berlin: Springer-Verlag, 2006.

[24] K. Shimohara I. Tanev, "XML-based Genetic Programming Framework: Design Philosophy, Implementation, and Applications," *Artificial-life and robotics*, vol. 15, no. 4, pp. 376-380, 2010.

[25] Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Evolving Emotion Recognition Module for Intelligent Agents," in *18th Asia Pasific Symposium on Intelligent and Evolutionary Systems*, Singapore, 2014, pp. 215-226.

[26] W.V. Friesen Paul Ekman, *Facial Action Coding System*. Washington DC, USA: Consulting Psychologist Press, 1977.

[28] B.W. Matthews, "Comparison of the Predicted and Observed Secondary Structure of T4 Phage Iysozyme," *Biochimica et Biophysica Acta - Protein Structure*, vol. 405, no. 2, pp. 442-451, 1975.

[27] Q. et al Mao, "Using Kinect for real-time emotion recognition via facial expressions," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 4, pp. 272-282, 2015.

[29] Shuyao Wang, Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Designing Evolving Computer Agent Capable of Emotion Recognition and Expression," in *AAAI Spring Symposium, Technical Report SS-140-1*, San Fransisco, 2014, pp. 96-97.

[30] Shuyao Wang, Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Evolving

Computer Agent Capable of Emotion Recognition and Expressions," in *SICE 2014*, Tsukuba, 2014.

[31] Dipak G Sharma, Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Evolving Emotion Recognition Module of Intelligent Agent based on Facial Expression and Gestures," in *20th International Symposium on Artificial Life and Robotics*, Oita, 2015, pp. p227-232.

[32] Dipak G Sharma, Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Evolving an Emotion Recognition Module for an Intelligent Agent using Genetic Programming and a Genetic Algorithm," *Artificial Life and Robotics*, vol. 21, no. 1, pp. 85-90, 2016.

[33] Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Application of Genetic Programming and Genetic Algorithm in Evolving Emotion Recognition Module," in *IEEE Congress on Evolutionary Computation*, Sendai, 2015, pp. 1444-1449.

[34] Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Implementation of Incremental GP on Evolving an Independent Emotion Recognition Module," in *Proceedings of the SICE Annual Conference 2016*, Tsukuba, 2016, pp. p537-542.

[35] Dipak Gaire Sharma, Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Implementation of Incremental Genetic Programming for the Evolution of a Standalone Emotion Recognition Model," *SICE Journal of Control, Measurement, and System Integration (JCMSI)*, in review.

[36] Ronald Fisher, *Statistical Methods for Research Workers*., 1925.

[38] Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Emotion Analysis Using

Emotion Recognition Module Evolved by Genetic Programming," *Harris Science Report*, vol. 57, no. 2, pp. p43-50, July 2016.

[37] Ivan Tanev, Katsunori Shimohara Rahadian Yusuf, "Using Emotion Recognition Module Evolved by Genetic Programming for Emotion Analysis," in *Proceedings of International Conference on Electronics and Software Science*, Takamatsu, 2015, pp. p50-59.

Publications

Journal Papers

Dipak Gaire Sharma, Rahadian Yusuf, Ivan Tanev, Katsunori Shimohara
Human Recognition based on Gait Features and Genetic Programming
Journal of Robotics, Networks and Artificial Life, Vol.1, No.3, pp.194-197, Dec. 2014.

Rahadian Yusuf, Dipak Sharma, Ivan Tanev and Katsunori Shimohara.
Evolving an emotion recognition module for an intelligent agent using genetic programming and a genetic algorithm
Artificial Life and Robotics, Vol.21, No.1, pp.85-90, Feb., 26, 2016.

Dipak Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara.
Human gait analysis based on biological motion and evolutionary computing
Artificial Life and Robotics, Vol.21, No.2, pp.188-194, May 22, 2016.

Rahadian Yusuf, Ivan Tanev and Katsunori Shimohara
Emotion Analysis Using Emotion Recognition Module Evolved by Genetic Programming
The Harris Science Review of Doshisha University, Vol.57, No.2, pp.43-50, July 2016

Dipak Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara
Evaluation of Genetic Programs in Multiple Cases for Gait Classification and Recognition
IEEJ Transactions on Electronics, Information and Systems, Vol.136, No.9, pp.1400-1410, September 2016

Rahadian Yusuf, Dipak Gaire Sharma, Ivan Tanev, Katsunori Shimohara
Implementation of Incremental Genetic Programming for the Evolution of a Standalone Emotion Recognition Model
SICE Journal of Control, Measurement, and System Integration (submitted on Oct 2016)

Rahadian Yusuf, Dipak Gaire Sharma, Ivan Tanev, Katsunori Shimohara
Effects of Cruising Speed on Steering Oscillations of Car Induced by Modeled Cognitively Impaired Human Driver
SICE Journal of Control, Measurement, and System Integration (submitted on Oct 2016)

Domestic Conferences

Rahadian Yusuf, Shuyao Wang, Ivan Tanev, Katsunori Shimohara
Evolving Computer Agent Capable of Emotion Recognition and Expression
計測自動制御学会 第 41 回知能システムシンポジウム，A23-2, 2014 年 3 月

International Conferences

Rahadian Yusuf, Shuyao Wang, Ivan Tanev, Katsunori Shimohara
Designing Evolving Computer Agent Capable of Emotion Recognition and Expression
2014 AAAI Spring Symposium, Technical Report SS-14-01, pp.96-97, March 2014.

Rahadian Yusuf, Ivan Tanev, Katsunori Shimohara
Evolving Emotion Recognition Module for Intelligent Agent
Proc. of The 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, vol.2, pp.215-226, Nov. 2014.

Rahadian Yusuf , Dipak Gaire Sharma , Ivan Tanev, and Katsunori Shimohara,
Evolving Emotion Recognition Module of Intelligent Agent based on Facial Expression and Gestures
The Twentieth International Symposium on Artificial Life and Robotics 2015 (AROB 20th 2015), pp.227-232, Jan. 21-23, 2015.

Dipak Gaire Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara,
Human Gait Analysis Based on Biological Motion and Evolutionary Computing
The Twentieth International Symposium on Artificial Life and Robotics 2015 (AROB 20th 2015), pp. 36-39, Jan. 21-23, 2015.

Rahadian Yusuf, Ivan Tanev and Katsunori Shimohara
Application of genetic programming and genetic algorithm in evolving emotion recognition module
IEEE Congress on Evolutionary Computation (CEC2015), pp.1444-1449, 25-28 May, 2015.

Dipak Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara
Analysis of Genetic Programming in Gait Recognition
IEEE Congress on Evolutionary Computation (CEC2015), pp.1418-1423, 25-28 May, 2015.

Rahadian Yusuf, Ivan Tanev and Katsunori Shimohara
Using Emotion Recognition Module Evolved by Genetic Programming for Emotion Analysis
The International Conference on Electronics and Software Science (ICESS-2015), pp.49-59, July 20-22, 2015.

Dipak Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara
Evaluation of Genetic Programs in Multiple Cases Evolved for Gait Classification and Recognition
The International Conference on Electronics and Software Science (ICESS-2015), pp.87-97, July 20-22, 2015.

Dipak Sharma, Rahadian Yusuf, Ivan Tanev, and Katsunori Shimohara
Steering Oscillations as an Effect of Cognitive Delay in Human Drivers
Proc. of the SICE Annual Conf. 2016, pp.229-236, September 2016

Rahadian Yusuf, Ivan Tanev and Katsunori Shimohara
Implementation of Incremental GP on Evolving an Independent Emotion Recognition Module
Proc. of the SICE Annual Conf. 2016, pp.537-542, September 2016