國 立 交 通 大 學

電控工程研究所

博士論文

正規化最小平方法為基礎的階層式合作共同進化演算法及其於模糊類神經網路設計和影像對準的應用

Regularized Least Squares based Hierarchical Cooperative Coevolutionary Algorithm for Neural Fuzzy Network Design and Image Alignment Applications

研究生：徐啟曜

指導教授：林昇甫 博士

中華民國一〇一年六月

正規化最小平方法為基礎的階層式合作共同進化演算法及其於模糊類神經網路設計和影像對準的應用

Regularized Least Squares based Hierarchical Cooperative Coevolutionary Algorithm for Neural Fuzzy Network Design and Image Alignment Applications

研 究 生：徐啟曜　　Student：Chi-Yao Hsu

指導教授：林昇甫 博士　　Advisor：Dr. Sheng-Fuu Lin

國 立 交 通 大 學

電控工程研究所

博 士 論 文

A Dissertation

Submitted to Institute of Electrical Control Engineering

National Chiao-Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

June 2012

Hsinchu, Taiwan, Republic of China

中華民國一〇一年六月

正規化最小平方法為基礎的階層式合作共同進化演算法及其於模糊類神經網路設計和影像對準的應用

研究生：徐啟曜　　指導教授：林昇甫 博士

國立交通大學 電控工程研究所

摘 要

進化型演算法經常被使用在訓練模糊類神經網路參數方面，主要是因為該方法有並行搜尋的技術。不過目前此類型的方法有著無法拓展到多數量的訓練參數以及低效率的調整模糊法則問題，所以本篇論文提出了正規化最小平方法為基礎的階層式合作共同進化演算法來改善上述問題。使用正規化最小平方法的主要效用為減少訓練參數的數量，而在階層式合作共同進化演算法方面，兩層級進化法被提出能夠有效地進化模糊規則以及使得網路的參數及其架構能夠被分別被區域性及全域性的進化，因此以正規化最小平方法為基礎的階層式合作共同進化演算法有著參數學習及架構學習的優點，並且進化完成的網路可以被應用到現實世界的實例。第一個應用為二維影像對準問題，本論文所提出的演算法則可用來建立一個以合作式模糊類神經網路為基礎的二維影像對準系統，該系統利用多級模糊神經網路來解決單級模糊神經網路在仿射參數的大範圍應用的困難。第二個應用為三維影像對準問題，採用本論文所提的學習演算法可建立以模糊類神經網路為基礎的粗糙到細緻的三維影像對準系統，該系統改善傳統的主成份分析對準法的高粗糙對準誤差的缺點，在細緻對準階段成功改善了遞迴式最近點法的繁重計算的問題。這些證據可以被發現在實驗結果中來表示本論文提出的二維及三維影像對準系統，相較於其他一些典型的影像對準系統，本文的方法有較佳的性能。

關鍵字: 正規化最小平方法，階層式合作進化型演算法，兩層級進化，參數學習，架構學習。

Regularized Least Squares based Hierarchical Cooperative Coevolutionary Algorithm for Neural Fuzzy Network Design and Image Alignment Applications

Student：Chi-Yao Hsu　　Advisor：Dr. Sheng-Fuu Lin

Institute of Electrical Control Engineering

National Chiao Tung University

Abstract

Evolutionary algorithms are very popular in training parameters of neural fuzzy network due to their parallel search techniques. However, current methods have problems of not scaling well to a large number of training parameters and adjusting fuzzy rules inefficiently. In this dissertation, a regularized least squares based hierarchical cooperative coevolutionary algorithm (RGLS-HCCA) is proposed to improve above problems. The major utility of RGLS is to reduce the number of learning parameters. In HCCA, two-level evolution is proposed to evolve fuzzy rules efficiently and make the parameters and structure of a network be evolved locally and globally, respectively. Thus, RGLS-HCCA has advantages of parameter learning and structure learning, and the evolved network can be applied to the real world applications. The first application is a 2D image alignment problem. The proposed RGLS-HCCA is used to construct a cooperative neural fuzzy network (CNFN)-based 2D image alignment system, which utilizes the multi-stage of neural fuzzy networks to solve problems that one-stage of neural network have difficulty in applying a large range of affine parameters. The second application is a 3D image alignment problem. The use of RGLS-HCCA can construct a neural fuzzy network (NFN)-based coarse-to-fine 3D image alignment system, which solve the problem of the high alignment error caused by principle component analysis (PCA) and heavy computational cost caused by iterative closest point (ICP). The evidence can be found in experimental results demonstrate the superior performance of the proposed 2D and 3D surface alignment system over typical systems.

Keywords: regularized least squares, hierarchical cooperative coevolutionary algorithm, two-level evolution, parameter learning, structure learning.

誌謝

經過多年的努力，終於取得博士學位，在博士班的研究生涯中，首先要感謝我的指導教授-林昇甫博士，林教授在研究上給予學生正確方向的指導，讓學生覺得獲益良多，並且林教授總是正向地鼓勵學生，讓學生對於學術研究充滿了信心，有了這份信心學生才能夠通過論文投稿以及博士口試這兩大關卡，所以非常感謝林教授的指導。

在博士論文口試方面，感謝口試委員潘晴財教授、鍾鴻源教授、張翔教授及林錫寬教授不辭辛苦參與學生的口試，並給與學生許多寶貴的修正意見，讓學生的論文能夠更加地完整，所以學生由衷地感謝口試委員。

接下來要感謝 806 實驗室的博士班學長弦澤及培家、同學逸章、學弟俊偉及裕筆，碩士班學弟柏宏、植彥、俊良、學妹婷婷及雅君，有了你們的幫忙，我的求學之路才會如此順利，另外特別要感謝實驗室已畢業的博士班學長-徐永吉博士，有了你在論文上的互相討論及協助，我才能完成博士論文中的許多研究。

也感謝中科院各個長官的栽培及支持，讓我能夠有機會攻讀博士，並順利取得博士學位，而我的同事也在我的求學之路上幫忙很多，不論在工作上的協助，或是在研究上的心得分享，對我而言都是相當大的助力。

而我的父母是我的心靈上的寄託，感謝你們對我的支持及鼓勵，我才能有信心接受許多挑戰，並達成我人生重要的夢想-取得博士學位，而我的哥哥對我的照顧，我也是心存感激。另外我要感謝我相識多年的女友，每當我在研究或工作上遇到挫折，妳總是在我身邊陪我渡過許多的難關。

最後我要再次感謝所有幫過我的人，有了你們的幫忙，我才能夠取得博士學位。

徐啟曜

一〇一年六月

Contents

List of Tables

List of Figures

Chapter 1
Introduction

For most interesting real world problems, the environment is more complicated and highly non-linear. For instance, to consider image alignment problems, the prediction of the relationship between input image and output pose is non-linear and it is hard to use the linear mathematical tools to accomplish modeling. Based on this fact, neural fuzzy networks can take its “black box” nature and linguistic information to deal with non-linearity. Thus, the purpose of this dissertation is to develop a methodology to automatically design neural fuzzy networks by using regularized least squares (RGLS) based hierarchical cooperative coevolutionary algorithm (HCCA) to evolve the networks for applying to real world problems.

This chapter is divided into four subsections. In Section 1.1, the motivation of this dissertation is introduced. Section 1.2 describes the related works of the evolutionary algorithm and image alignment applications. Section 1.3 specifies the proposed approach. In Section 1.4 the organization of this dissertation is presented.

1.1 Motivation

In most physical systems, the relationship between input and output is inherent non-linear in nature. Non-linear relationship is difficult to solve and give rise to interesting research topics. To cope with non-linearity, neural networks are algorithms that can be used to perform nonlinear statistical modeling and diverse engineering applications based on this modeling method have been successfully developed. However, their operation is restricted to the numeric domain. In recent years, neural fuzzy networks (NFNs) used for several problems have become a popular research topic [1]-[6], especially for solving nonlinear and complex problems [7]-[10]. The reason is that it combines fuzzy set and fuzzy logic into the neural

network framework to bring the benefits of processing linguistic and numeric information.

Training parameters is the main issue for designing neural fuzzy networks. The most well known algorithm is back-propagation (BP) [3], [6] which is a powerful training technique for tuning the parameters of networks. Since the BP algorithm adopts the steepest decent approach to minimize the error function, they suffer from a major problem: getting in local minima of the error surface. To deal with the drawback, there is a need to face with suboptimal problem. Towards this end, evolutionary algorithms appear to be better candidates than the BP algorithm because of their parallel search techniques and optimization methodology.

Recently, several evolutionary algorithms, including genetic algorithm (GA) [11], hierarchical genetic algorithm (HGA) [12], symbiotic adaptive neruoevolution (SANE) [13], enforced sub-population (ESP) algorithm [14], and multi-groups cooperation based symbiotic evolution (MGCSE) [15] have been proposed to train neural networks or fuzzy systems. Although these algorithms can obtain better performance than the BP algorithm, they still have difficulty in scaling to more complex tasks or high input dimension of networks. Moreover, they also conduct the problem of the random group selection of fuzzy rules and the lost of potential fuzzy rules combinations. Therefore, these problems are the main issues this dissertation intends to address.

Furthermore, to transfer the problem from simulation to the real world applications, two image alignments tasks are utilized. The first one is a 2D image alignment problem which is widely applied to many industrial applications, such as automatic visual inspection, factory automation, and robotic machine vision. The second one is a 3D image alignment problem which is an extended version of 2D image alignment. Thus, this dissertation aims to propose an evolutionary algorithm to train neural fuzzy networks to apply these two real world problems.

1.2 Related Works

Neural fuzzy networks are gaining research interest and they have been widely used in fields of pattern recognition, control problems, image processing, and diagnosis. The major benefit of neural fuzzy network is the integration of computation power from neural networks and human-like reasoning from fuzzy systems. Since neural fuzzy networks can bring such benefit, how to train neural fuzzy networks has become a critical issue.

The back-propagation (BP) algorithm [3] is a typical method for training neural fuzzy networks. Although the use of steepest descent technique in BP learning can reach the local minimal much quickly, the global minimal may be never found. Thus, evolutionary algorithms are better ones than BP due to their parallel search techniques. Recently, evolutionary fuzzy models have become a popular research field [16]-[24]. The evolutionary fuzzy model is a learning process using evolutionary learning procedures to generate a fuzzy system automatically. Among these evolutionary fuzzy models, the well-known algorithms are the genetic fuzzy models, which are augmented by incorporating genetic algorithms (GAs). There are several genetic fuzzy models have been proposed [16]-[18]. In [16], Karr adopted GAs to adjust membership functions for designing a fuzzy controller where its fuzzy rule set must be predetermined. Lin and Jou [17] applied GAs to fuzzy reinforcement learning to control a magnetic bearing system. In [18], Juang *et al.* proposed symbiotic evolution based genetic reinforcement learning for designing fuzzy controllers. In their work, the symbiotic-evolution-based fuzzy controller required fewer trail and less CPU time than the traditional GA-based fuzzy controller.

Although the genetic fuzzy models can be used to search for the optimal solution, they may have some limitations, such as the same lengths of chromosomes, predefined parameters, and so on. Thus, there are several improved evolutionary algorithms [19]-[22] to take into account these limitations. In [19], Carse *et al.* used the fusion of genetic algorithms and fuzzy

logic to evolve variable length fuzzy rule-sets. In [20], Bandyopadhyay *et al*. proposed variable-length genetic algorithm (VGA) to encode different length of chromosomes in the same population. Tang [21] proposed a hierarchical genetic algorithm to enable the optimization of designing a fuzzy system for particular applications. Juang [22] proposed a combination of online clustering and Q-value based GA learning for fuzzy system design (CQGAF) to generate fuzzy rules automatically and free parameters in a fuzzy system. In addition, Gomez and Schmidhuber [14] proposed enforced subpopulations (ESP) to provide several subpopulations to evaluate each partial solution. The subpopulations that are used to evaluate the solution locally can obtain better performance than those methods that only use one population for evaluating the solution. In [15], Hsu and Lin proposed a multi-groups cooperation based symbiotic evolution (MGCSE) to train a TSK-type neuro-fuzzy network (TNFN). They develop a novel symbiotic evolution to let each sub population can cooperate to generate better offspring.

In spite of the above evolutionary learning algorithms improving genetic fuzzy models, these algorithms may conduct one or more of the following problems: (1) the random group selection of fuzzy rules, (2) low convergence rate as the problem becomes complex, and (3) potential fuzzy rules combinations are lost.

Recently, hierarchical enforced sub-populations (HESP) [23] provided a hierarchical evolutionary for preserving the potential neuron combinations. In their work, in spite of keeping useful networks, HESP still suffer from: the lengths of chromosomes must be the same and the number of neurons has to be assigned in advance. To this end, this study attempts to propose an evolutionary learning algorithm, which incorporates concepts of data-mining [25-29], regularized least square, and hierarchical evolution, for improving the problems that were mentioned above and achieve the following goals: (1) adapt the trained network to more complex tasks, (2) select groups of fuzzy rules systematically, (3) preserve good combinations of fuzzy rules, (4) allow variable length of chromosome, and (5) adjust

the number of fuzzy rules automatically.

In addition, to consider 2D image alignment application, the problem of precise image alignment has been well-studied in several fields. In [30], Liu et al. point out that image alignment techniques are broadly classified as feature-based [31] and [32] and area-based matching approaches [33-35]. Amintoosi et al. pointed out that area-based methods produce better results than results with low signal-to-noise ratio (SNR) from feature-based methods. Moreover, Zitova and Flusser indicated [39] that area-based methods are preferably applied to less detailed images. In this study, we assume that our proposed image alignment system is developed for industrial inspection tasks such that the captured images usually have less detail. Thus area-based methods that adopt global descriptors are recommended in this paper.

In recent years, the neural network-based image alignment utilizing global features have been a relatively new research subject [40-44]. In [40-43], the alignment scheme is to estimate the affine parameters by a feedforward neural network (FNN). Although FNN is helpful to improve the alignment efficiency, such methods must take a large number of iterations to minimize the error function and several training attempts are needed to provide the robust FNN. In addition to FNN-based methods, Sarnel et al. [44] used a radial basis function neural network (RBFNN) to align images. According to their results, the training time of a RBFNN has been reduced, and the alignment accuracy and robustness against noise are better than those of FNN-based methods. However, a major drawback of the existing neural network-based methods is the difficulty in applying to align images on a large range of affine transformation. The reason is that a large range of affine parameters would lead to a large amount of training data such that the mapping surface becomes more complex and applying one-stage neural network to estimate a large range of affine parameters accurately is almost impossible. In this dissertation, a scheme of multi-stage neural network is proposed to overcome the problem produced by the one-stage neural network. The notion of this approach is to divide a large size of the network into several small networks, aiming to gradually reduce

the image alignment error and finally obtain the desired accuracy. Such phenomenon can be considered a coarse-to-fine alignment of the sensed image with the reference image.

Regarding the 3D image alignment application, the problem of 3D image alignment has been implemented by several methods [45-50]. Among them, a coarse-to-fine technique is a useful way for performing 3D image alignment [45] and [46]. Coarse alignment provides an approximate transformation for aligning two images. Such alignment must be efficient and accurate. Fine alignment uses the initial gauss of a transformation given by a coarse alignment as a starting point to iteratively minimize the distance between the input and the destination images. Specifically, in consideration of coarse image alignment, common methods [45] and [46] utilized principal component analysis (PCA) [51] for coarsely aligning two images due to its high-speed performance. However, PCA cannot ensure that the laser scanned point clouds have the same orientation of principal axes as the reference model. This phenomenon would cause a high alignment error in the coarse alignment phase. In consideration of fine alignment method, iterative closest point (ICP) [52] is a typical method to iteratively calculate the rigid-body transformation to minimize the cost function. Although ICP can provide highly accurate 3D image alignment, its heavy computational cost in searching corresponding points has been criticized by many researchers [45, 46, 53-55]. To this end, this dissertation intends to propose a coarse-to-fine 3D image alignment scheme to improve the drawback generated by PCA and ICP.

1.3 Approach

In this dissertation, three major approaches are proposed. The first one is an evolutionary algorithm called RGLS-HCCA which is used to design neural fuzzy networks. The second one is cooperative neural fuzzy network (CNFN)-based 2D image alignment method. The third one is TNFN-based coarse-to-fine 3D image alignment method. Among these methods, the second and third ones are the applications of RGLS-HCCA designed neural fuzzy

networks.

Regarding the RGLS-HCCA method, the RGLS method is utilized to control HCCA to converge toward optimal solution quickly. In HCCA, two-level evolutions are proposed: parameter level evolution (PLE) and structure level evolution (SLE). In PLE, a data-mining selection method (DMSM) based evolutionary learning algorithm is utilized to evolve parameters of networks. By using DMSM, the suitable groups can be identified for chromosome selection and such selection method would solve the random group selection problem caused by some typical cooperative coevolution algorithms [15, 57-60]. Moreover, to prevent the lost of potential fuzzy rules combinations, the good combinations of fuzzy rules evolved in PLE are reserved for being the initial populations of SLE. In SLE, the initial population are mated and mutated to produce new structure level of networks. Similar to PLE, the good fuzzy rules of evolved network in SLE are inserted into the PLE. Thus, by interacting two level evolutions, the parameters and structure of network can be evolved locally and globally, respectively. Besides, this dissertation combines variable antecedent-part crossover (VAC), variable antecedent-part mutation (VAM), and self-regulated mechanism (SRM) such that the variable length of chromosomes can be evaluated and the number of fuzzy rules can be adjusted automatically.

Regarding the CNFN-based 2D image alignment method, it is an application of RGLS-HCCA. Each CNFN contains multi-stage of TNFN and each TNFN is trained by the proposed RGLS-HCCA method. The aim of CNFN is to solve tasks that are too difficult to solve directly. Instead of trying to use one neural network to solve difficult problems, CNFN utilizes multi-stage of neural fuzzy networks to cover the whole problem. Each stage of networks manages a simple level problem and through each network cooperating, the combined network can be applied to a difficult level problem. For a 2D image alignment task, one-stage neural network have difficulty in estimating a large range of affine parameters accurately. Thus, CNFN utilizes multi-stage of networks to adapt image alignment to a larger

range of affine parameters. The input sensed image is sent into each network in turn to gradually reduce the image alignment error and finally obtain the desired accuracy.

Regarding the TNFN-based coarse-to-fine 3D image alignment method, it is an extended version of 2D image alignment task. The TNFN-based coarse alignment, which aims to improve PCA, utilizes multi-views of modified viewpoint feature histogram (MVFH) to be the input of TNFN and the corresponding 3D poses to be the output of a TNFN. Thus, once the training of TNFN has completed the relation between the input feature and output pose can be inferred and such relation results in more accurate pose estimation of the input 3D image than that of the PCA method. For the TNFN-based fine alignment method, which aims to improve ICP, it takes the notion of combining the surface modeling with the downhill simplex optimization method to iteratively reduce distance from the input image to the reference image. The major benefit of the TNFN-based fine alignment method is to avoid calculating the corresponding points, which is a problem that ICP suffer from.

1.4 Organization of Dissertation

This dissertation is divided into six chapters. Chapter 1 introduces the motivation, related work, approach, and organization of the dissertation. Chapter 2 provides the fundamental information used in the dissertation. The foundation includes regularized least squares method, neural fuzzy network, cooperative coevolutionary learning, 2D image alignment, and 3D image alignment. In Chapter 3, RGLS-HCCA is described. RGLS-HCCA consists of the RGLS method and the two-level evolutions: parameter level evolution and structure level evolution. Chapter 4 describes the methods of 2D and 3D image alignment which are the applications of RGLS-HCCA. In Chapter 5, three experiments are performed to demonstrate the superiority of RGLS-HCCA over other algorithms. The first experiment is a prediction of Mackey-Glass time series problem, which is a benchmark to verify the proposed algorithm. The second and third experiments, which are applications of RGLS-HCCA, are 2D and 3D

image alignment tasks, respectively. In Chapter 6, the conclusions and future work of the dissertation are discussed.

Chapter 2
Foundations

In this chapter, three major backgrounds of cooperative coevolutionary learning, 2D image alignment, and 3D image alignment are introduced. For the cooperative coevolutionary learning, the typical SANE method is used to specify how to perform evolutionary learning. For 2D and 3D image alignment, the procedures of aligning 2D and 3D images are described and alignment results of general 2D and 3D image alignment methods are briefly presented.

This chapter is divided into five subsections. The concepts of the regularized least squares method and neural fuzzy network are introduced in Section 2.1 and 2.2, respectively. In Section 2.3, the general method of cooperative coevolutionary learning is described. Section 2.4 and 2.5 will discuss how to perform 2D and 3D image alignments tasks.

2.1 Regularized Least Squares Method

Before discussing the regularized squares method, the least square method is introduced. Give a target vector y, and data matrix X. The most popular loss function used for regression problems is the residual sum of squared errors (RSS):

$$RSS = \|Xw - y\|_2^2. \tag{2.1}$$

The least square method is defined as setting w to minimize the expression. Thus, differentiating Eq. (2.1) with respect to w can obtain:

$$X^T(Xw - y). \tag{2.2}$$

By setting Eq. (2.2) with 0 to solve w:

$$w = (X^T X)^{-1} X^T y. \tag{2.3}$$

Unfortunately, the matrix $X^T X$ may be singular or nearly singular, which make it difficult to invert. To address this problem, Tikhonov [61] proposed a regularization to solve

the numerical instability of the matrix inversion. The method of regularization adds a positive constant to the diagonals of $X^T X$ to make the matrix nonsingular. Thus, the expression of Eq. (2.3) can be switched to:

$$w = (X^T X + \lambda I)^{-1} X^T y, \tag{2.4}$$

where λ is a regularization parameter. Since Eq. (2.4) is used to solve the least square problem, Tikhonov regularization is called regularized least squares [62], which is also called damped least squares [63-65]. Moreover, to differentiate from the abbreviation of recursive least square (RLS), this paper takes the idea from [66] to abbreviate regularized least squares to RGLS.

In addition to RGLS to solve the problem of the matrix $X^T X$ being singular, pseudo inverse is another solution. Thus, in the section of experimental results, this dissertation will compare regularized least squares with pseudo inverse.

2.2 Neural Fuzzy Network

In Lin and Peng's work [2], there are two typical types of neural fuzzy network (NFN) and they are Mamdani-type [5] and TSK-type [4]. According to [6] and [67], the authors have shown that the TSK-type NFN can offer better network size and learning accuracy than the Mamdani-type NFN. Thus, in this dissertation, only the TSK-type NFN is introduced and such NFN is applied to image alignment applications.

A TSK-type neuro-fuzzy network (TNFN) [4] employs a linear combination of the crisp inputs as the consequent part of a fuzzy rule. The fuzzy rule of the TSK-type neural fuzzy system is shown in Eq. (2.5), where n and j represent the dimension of the input and the number of the fuzzy rules respectively.

$$\textbf{IF } x_1 \textbf{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textbf{ and } x_2 \textbf{ is } A_{2j}(m_{2j}, \sigma_{2j}) \textbf{ and} \cdots \textbf{and } x_n \textbf{ is } A_{nj}(m_{nj}, \sigma_{nj})$$

$$\textbf{THEN } y' = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n. \tag{2.5}$$

The structure of TNFN is shown in Fig. 2.1, where n represents the dimension of the

input. It is a five-layer network structure. The functions of the nodes in each layer are described as follows:

Layer 1 (input node): Each node in this layer is called an input linguistic node, which corresponding one linguistic variable. These nodes only pass the input signal to the next layer.

$$u_i^{(1)} = x_i, \tag{2.6}$$

where $u_i^{(1)}$ denotes the *i*th node's input in the first layer and x_i denotes *i*th input dimension. The number of nodes in this layer is the dimension of input vector.

Layer 2 (membership function node): each node in this layer acts as a Gaussian membership function, and its output value specifies the degree to which the given input value belongs to a fuzzy set. Thus, the membership value in layer 2 can be calculated by:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right), \tag{2.7}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(2)}$ are the outputs of 1st and 2nd layers ; m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the *j*th term of the *i*th input variable x_i, respectively. In this paper, the reason of adopting the Gaussian membership function is that it can be a universal approximator of any nonlinear functions [6]. Besides, the number of nodes in this layer is the dimension of input vector multiplied by the number of fuzzy rules.

Layer 3 (rule node): The output in this layer is used to perform precondition matching of fuzzy rules. In the TNFN, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation:

$$u_j^{(3)} = \prod_i u_{ij}^{(2)}. \tag{2.8}$$

The number of nodes in this layer is the number of fuzzy rules.

Layer 4 (consequent node): each node in this layer calculates the consequent value. Each consequent value (linear combination of the crisp inputs) is weighted by the firing strength of

the fuzzy rule and it can be written by:

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i), \tag{2.9}$$

where the summation is the consequent part and w_{ij} is its corresponding parameters. The number of nodes in this layer is the dimension of output vector multiplied by the number of fuzzy rules.

Layer 5 (output node): The node in this layer computes output signal. The output node integrates with links connected to it and acts as a defuzzifier with:

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i)}{\sum_{j=1}^{M} u_j^{(3)}}, \tag{2.10}$$

where $u^{(5)}$ is the output of 5th layer , w_{ij} is the weighting value with *i*th dimension and *j*th rule node, and *M* is the number of a fuzzy rule. The number of nodes in this layer is the dimension of output vector.

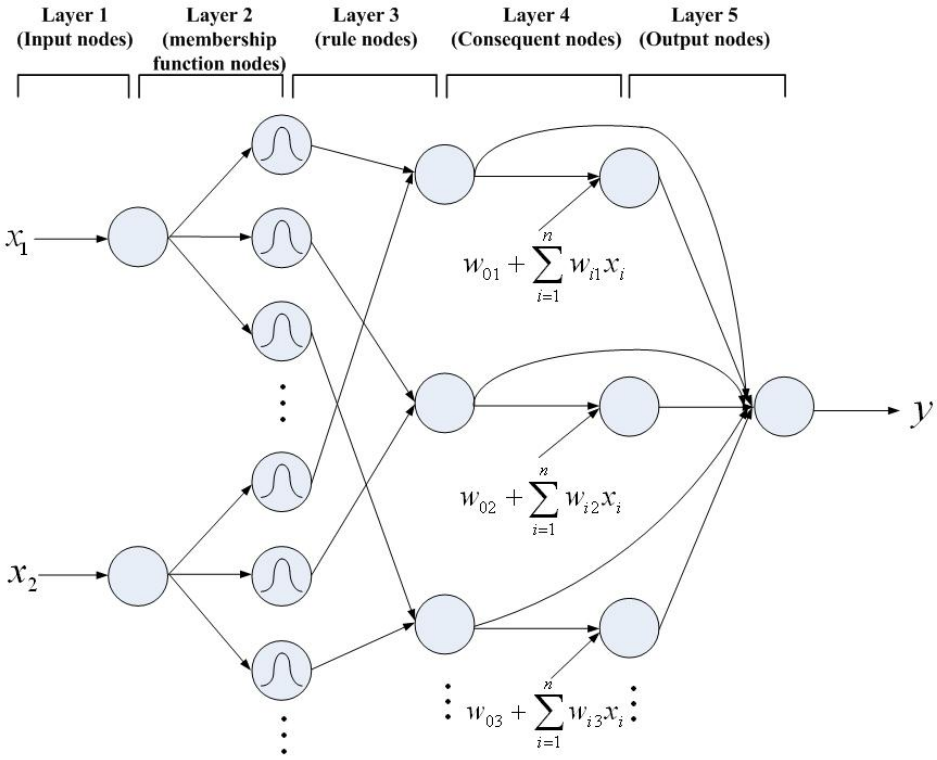


Figure 2-1: Structure of TNFN.

2.3 Cooperative Coevolutionary Learning

Evolutionary algorithms (EAs) are the methods for solving difficult problems using notions of Darwinian evolution. EAs have been applied to many applications and the major benefit of EAs over traditional local search methods is their parallel search ability. However, EAs have difficulty in scaling to large problem domains. For solving this problem, researches have extended EAs to cooperative coevolutionary algorithms (CCEAs). Instead of solving the entire problem, the notion of cooperative coevolutionary learning is to reduce the complex of difficult problems through modularization. In other words, a difficult complete problem can be divided into small simple problems. In CCEAs, each individual represents only a partial solution and a full solution is built by means of cooperating with other partial solutions. Thus, each individual can be evolved locally and recombined it with other well-performed individuals to form a good total solution.

Symbiotic adaptive neruoevolution (SANE) is one of typical CCEAs. In SANE, partial solutions can be viewed as specializations. It indicates that partial solutions specialize toward one aspect of the full solution. To concern with fitness evaluation, the fitness of an individual is calculated by summing all combinations of that individual with other individuals and dividing by the total number of combinations. Thus, the fitness value reflects an average value of combined full solutions. Fig. 2.2 presents the basic steps of SANE. As shown in this figure, there are nine steps of SANE and they are described as follows.

Step1. Initialization: in this step, all fitness values are clear and all genes of individuals are assigned a random value within a predefined range.

Step2. Selection: randomly select *n* individuals from the population.

Step3. Create a neural network: use the selected *n* individuals to build a neural network.

Step4. Evaluate the network: after the neural is created, the evaluation is performed according the given problem.

Step5. Selection times check: each individual must be selected sufficient times. If the selection time does not satisfy, then go to Step2 to continue the selection step.

Step6. In this step, the average fitness value of an individual is computed by dividing the total fitness value of each chromosome by the number of times that it has been selected to build networks.

Step7. Termination check: check the fitness value with respect the whole network not a single individual. If the fitness value of the whole network satisfies the pre-setting value, then SANE terminate.

Step8. Crossover: a one-point crossover strategy is used to exchange the site's values between the selected sites of individual parents to create new individuals, which are offspring inheriting the parents' merits.

Step9. Mutation: in the last step, the gene is mutated at the rate 0.1% drawn randomly from the domain of the corresponding variable. Then go to Step 2 to perform selection.

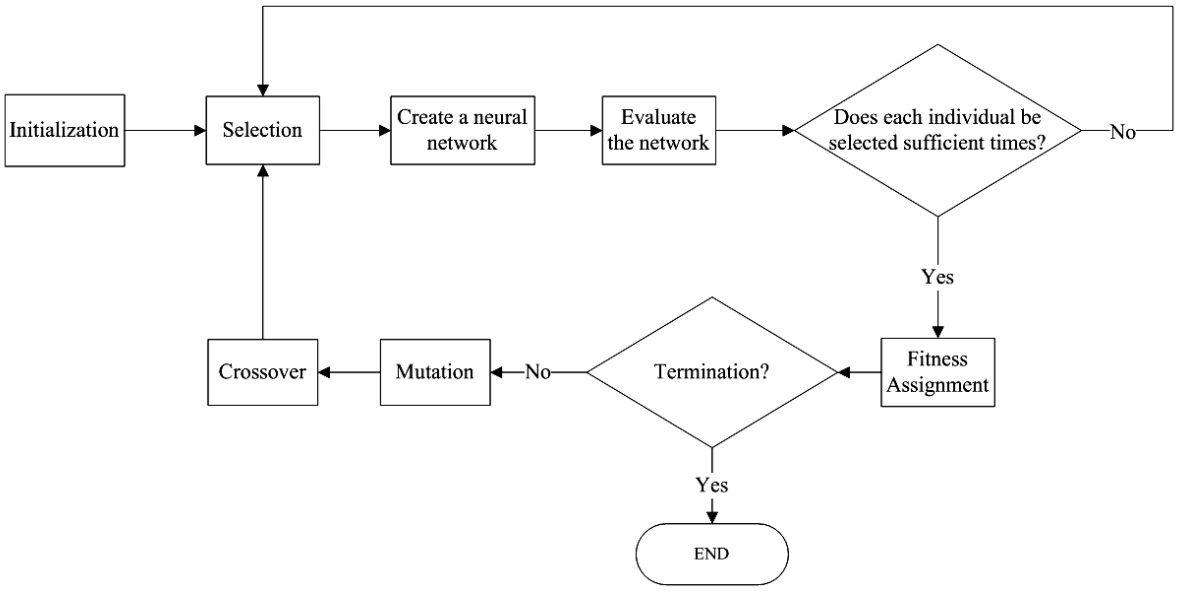


Figure 2-2: Basic Steps of SANE.

Although SANE can obtain better performance than traditional evolutionary approaches, it still has the problem that the algorithm cannot evaluate each partial solution independently. More specifically, SANE use only one population to evaluate every partial solution, this will

cause partial solutions too similar. Therefore, the algorithm may have less chance to obtain optimal solution. To this end, MGCSE [15], which is a previous evolutionary algorithm and similar to ESP, was proposed for evolving TSK-type neural fuzzy networks. Compare to SANE, MGCSE provide several groups to evaluate each partial solution. Each group in the MGCSE represents a group that consists of the set of the chromosomes that belongs to the partial solution. In MGCSE, the population consists of several sub-populations and each sub-population represents the set of the chromosomes that belongs to one fuzzy rule. The structure of the chromosome is shown in Fig. 2.3. In this figure, each fuzzy rule represents a chromosome that is selected from a group, P_{size} represents there are P_{size} groups in a population, and "M_k" represents M_k fuzzy rules are used to construct a TSK-type neural fuzzy network.

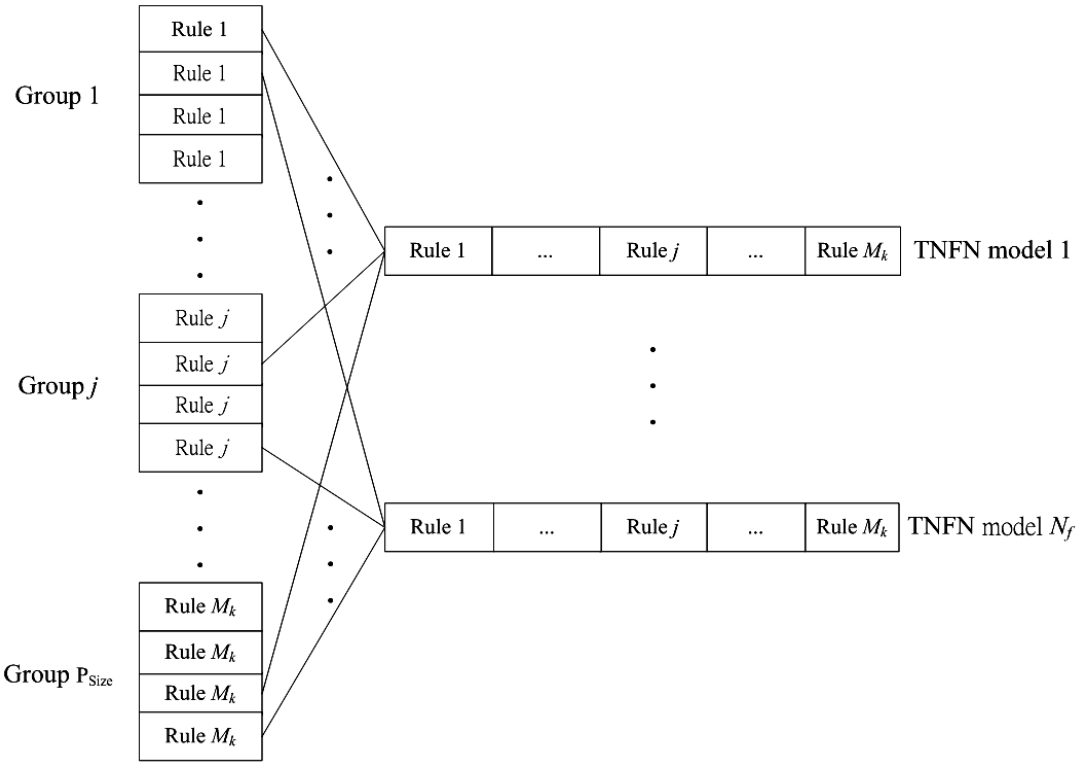


Figure 2-3: Structure of the chromosome in MGCSE.

The coding structure of the chromosome in MGCSE is shown in Fig. 2.4. This figure describes a fuzzy rule that has the form of Eq. (2.5), where m_{ij} and σ_{ij} represent a

Gaussian membership function with mean and deviation, respectively, and w_{ji} is the weight with *i*th dimension and *j*th rule node.

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	$\cdots$	m_{nj}	σ_{nj}	w_{j0}	w_{j1}	w_{j2}	$\cdots$	w_{jn}

Figure 2-4: Coding a fuzzy rule of a TNFN into a chromosome in MGCSE.

However, MGCSE have difficulty in scaling to more complex tasks or high input dimension of networks, conduct the problem of the random group selection of fuzzy rules, and the lost of potential fuzzy rules combinations. In consideration of the lost of potential fuzzy rules combinations, Gomez had proposed HESP to accomplish it. Nevertheless, HESP suffers from the problems that the lengths of chromosomes must be the same and the number of neurons has to be assigned in advance. To this end, this dissertation proposes RGLS-HCCA to address the above mentioned problems.

2.4 2D Image Alignment

In this subsection, a 2D image alignment task is introduced. Image alignment can be viewed as a mapping between two images by means of a geometric transformation. Typically, geometric transformation contains many types, including affine, similarity, and projective transformation. Among them, affine transformation is the most common used type and it composites of translation, rotation, and scaling. Thus, this paper adopts the affine transformation as the transformation model. Figure 2.5 shows an example of a remote controller with different transformation parameters. In Fig. 2.5 (a), it represents a reference image which other transformed images want to align with. In other words, if the pose of the transformed image is known, then the transformed image can be recovered to the original pose of the reference image by reversing the pose. Thus, a 2D image alignment task defined in this dissertation is to align transformed images with the reference image.

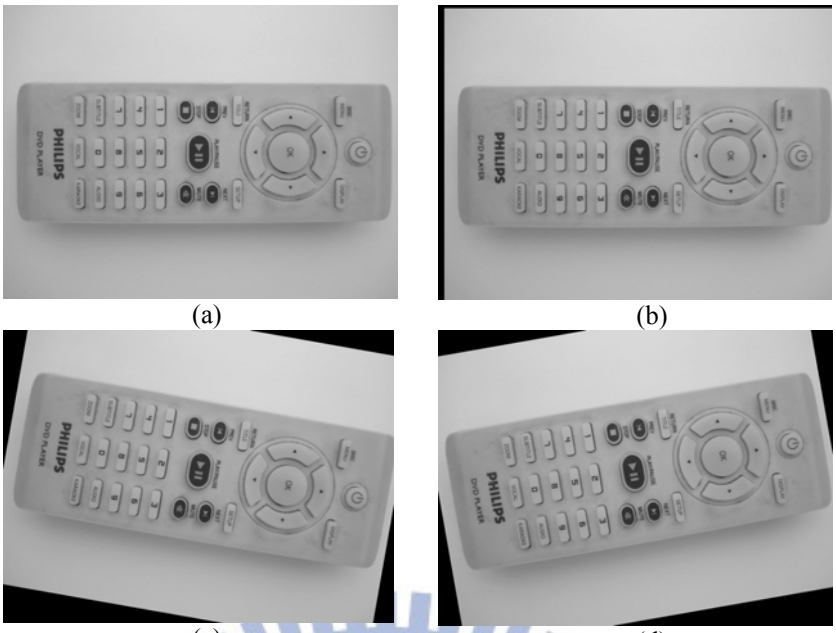


Figure 2-5**:** Example of generating training images with different affine transformation: (a) reference image, (b) translation, (c) clockwise rotation, and (d) counterclockwise rotation.

Since industrial inspection tasks are assumed, area-based alignment methods that adopt global descriptors are recommended. Thus, this study tries to focus on developing a good area-based alignment method. Figure 2.6 illustrates a typical procedure of an area-based 2D image alignment system. As shown in this figure, the sensed image is sent into the descriptor to extract the feature. Then, feed the feature into a pose estimation block to estimate the pose with respect to the reference image. Finally, the estimated affine transformation parameters can be used to align the sensed image with the reference image. Toward this end, seeking accurate affine transformation parameters is the most important fields for aligning images.

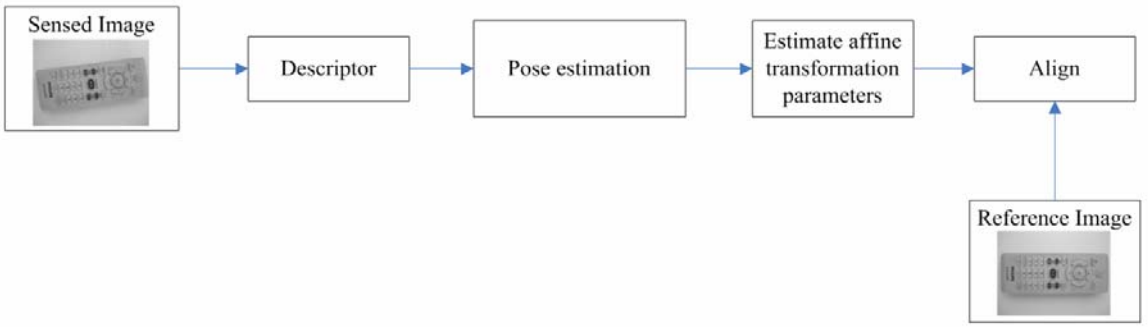
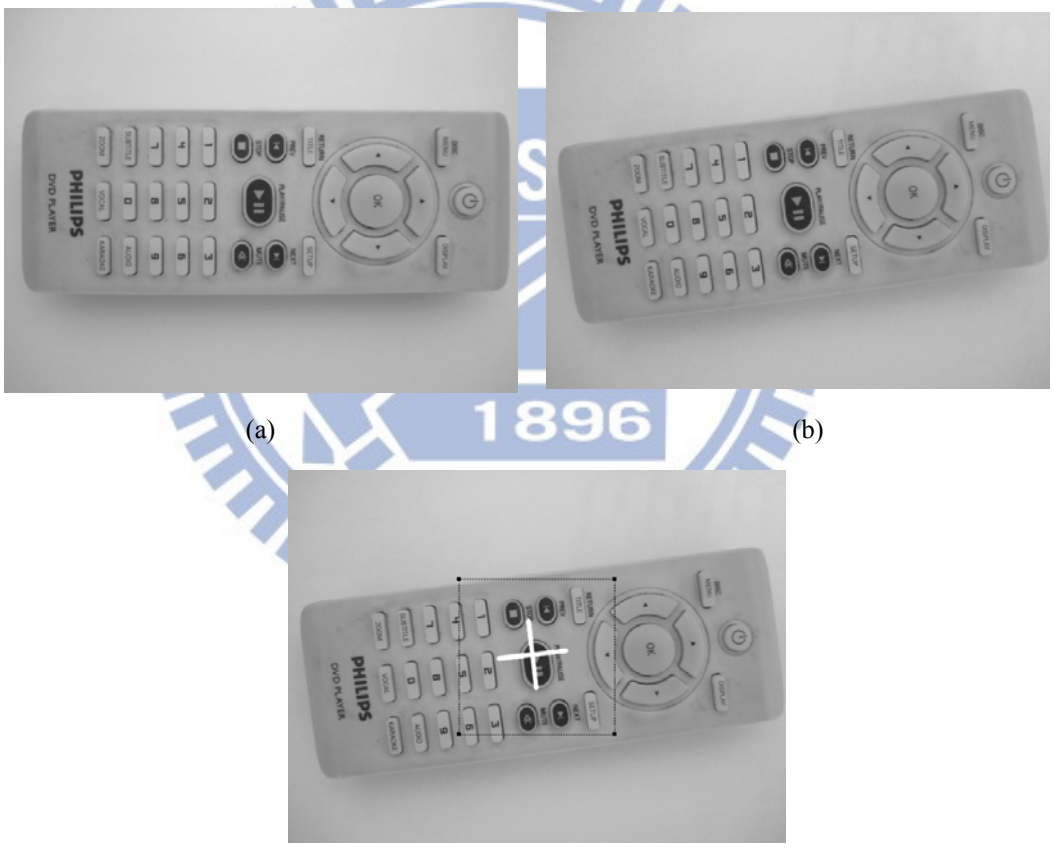


Figure 2-6: Typical procedure of an area-based 2D image alignment system.

Figure 2.7 illustrates an example of aligning 2D images where figure (a) is a reference image, figure (b) is an input image, and figure (c) is a alignment result of using neural network based alignment scheme defined in [44]. In Fig. 2.7 (c), the cross sign denotes the estimated results of Sarnel's work [44] and from the location of this cross sign, the alignment results is not good enough. The major drawback of such approach is that they have difficulty in applying to align images on a large range of affine transformation. Thus, this dissertation proposes a CNFN-based 2D image alignment method to perform coarse-to-fine alignment of the sensed image and the reference image.



(a) (b)

(c)

Figure 2-7: Example of 2D alignment: (a) reference image, (b) image with an affine transformed, and (c) alignment results of neural network based scheme.

2.5 3D Image Alignment

The 3D image defined in this dissertation is a range image which is scanned by an

imaging laser scanner. Each pixel in the range image reflects a range data which indicates a distance from the sensed point to the scanner. In other words, the range data can be considered as a 3D point with respect to the scanner. Thus, the scanner can be a center of a coordinate system to represent each sensed range data. Figure 2.8 presents an example of the range image, intensity image, and a 3D point cloud data. From this figure, the range image utilizes the color bar to represent the range data. The intensity image, which is also generated by the imaging lasers scanner, is used to be the corresponding map of range image. The 3D point cloud data, which is created by transforming range data to Cartesian coordinate, shows the 3D position of each pixel.

Figure 2.9 illustrates the procedure of a 3D image alignment task. From this figure, the 3D scene is scanned by a 3D imaging laser scanner where the size of the scanned scene is 256×256 with 20 degree field of view. The region of interest (ROI) is extracted by using the segmentation algorithm described in [68]. The reference model is a target 3D surface that the ROI wants to align with. Thus, the purpose of the 3D image alignment task is to align the ROI with the reference model.

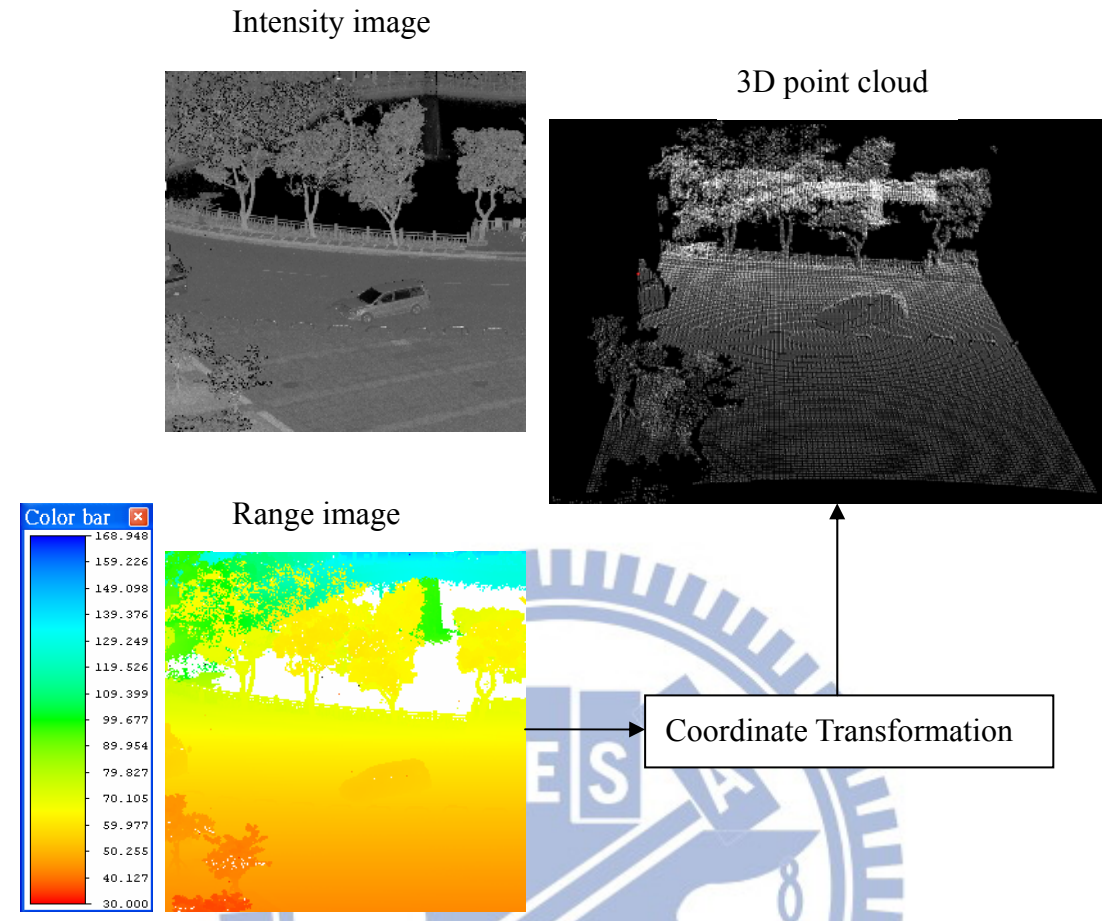


Figure 2-8: Example of 3D image.

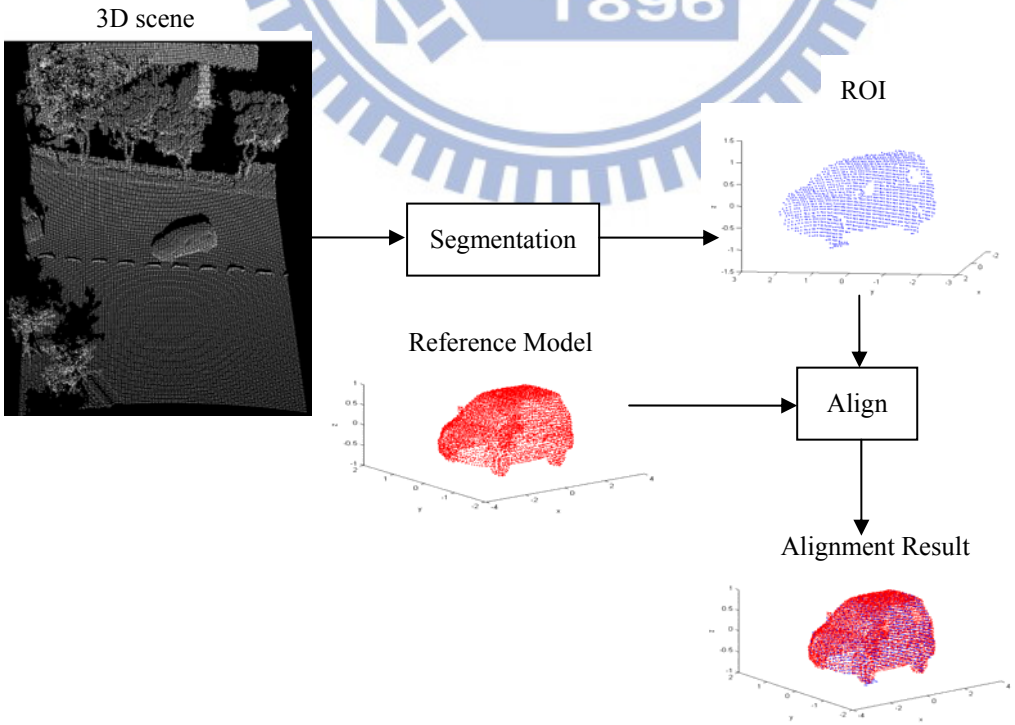


Figure 2-9: Procedure of a 3D surface alignment task

According to Chapter 1, a coarse-to-fine technique is a useful way to perform 3D image alignment tasks. In consideration of coarse image alignment, common methods [45] and [46] utilized PCA [51] for coarsely aligning two images due to its high-speed performance. In consideration of traditional fine alignment methods, iterative closest point (ICP) [52] is a typical method to iteratively calculate the rigid-body transformation to minimize the cost function.

Figure 2.10 illustrates an example of aligning an input 3D point with reference model using PCA. From this figure, (a) and (b) represents the principal axes of a 3D reference model and input 3D point data, respectively. Figure 2.10 (c) depicts the alignment results of PCA method. From Fig. 2.10 (a)-(c), we can know that since the input laser scanned 3D data is partial, its principal axes would be askew with respect to the 3D reference model and such case results in the large alignment error of PCA method (seen from Fig. 2.10 (c)). Based on this fact, this dissertation will propose a TNFN-based coarse alignment method that utilizes the pose estimation to replace of aligning principal axes.

Figure 2.11 illustrates an example of performing ICP fine alignment where figure (a) is the initial alignment yielded by PCA coarsely alignment and figure (b) is final fine alignment performed by ICP. Although ICP can get a good result for fine alignment, its heavy computational cost in searching corresponding points is a problem. To this end, this paper proposed a TNFN-based fine alignment method which combines surface modeling and the downhill simplex optimization method to improve the problem.

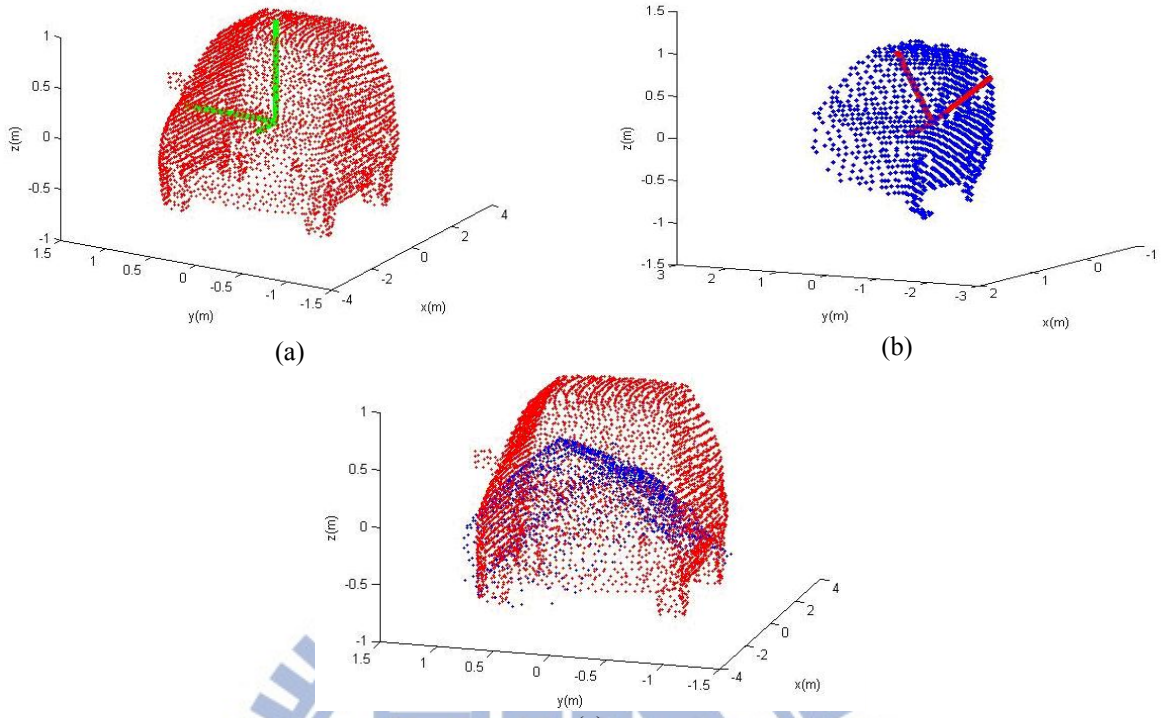


Figure 2-10: Example of coarse alignment using PCA: (a) the principal axes of the reference model, (b) the principal axes of the input 3D data, and (c) alignment results of the PCA method.

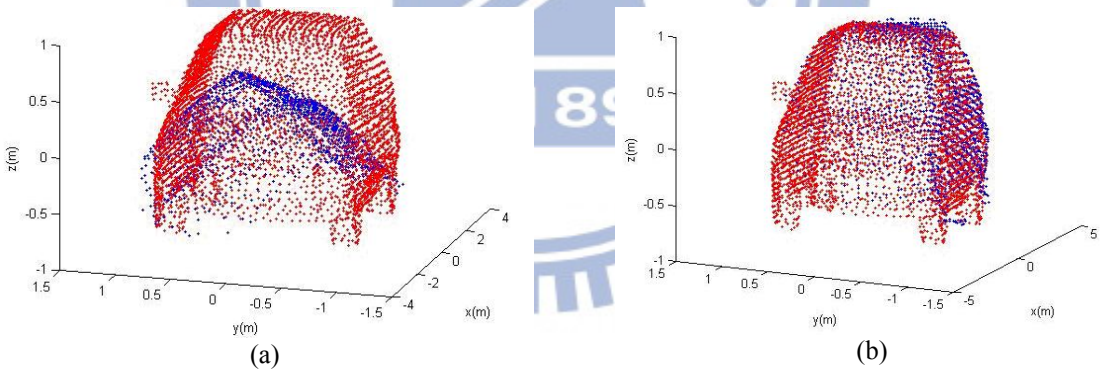


Figure 2-11: Example of fine alignment using ICP: (a) the initial alignment yielded by PCA and (b) alignment results of the ICP method.

Chapter 3
Regularized Least Squares Based Hierarchical Cooperative Coevolutionary Algorithm

The learning process of RGLS-HCCA is shown in Fig. 3.1. As show in this figure, RGLS-HCCA involves two major evolutions: parameter level evolution (PLE) and structure level evolution (SLE). The blocks of inserting good networks and inserting good neurons (i.e. good fuzzy rules) are the connection between the parameter and structure level evolution. These two operations indicate that good evolved results in one level evolution would be transferred to another level evolution. Once receiving good neurons or networks, the received chromosomes would be mated with other old chromosomes to yield some new offspring. Therefore, by exchanging the good information between two levels of evolution, we have more chance to find the global optimal solution.

This chapter is divided into two subsections to introduce the proposed two-level evolution. In Section 3.1, parameter level evolution is discussed. Section 3.2 describes how structure evolution works.

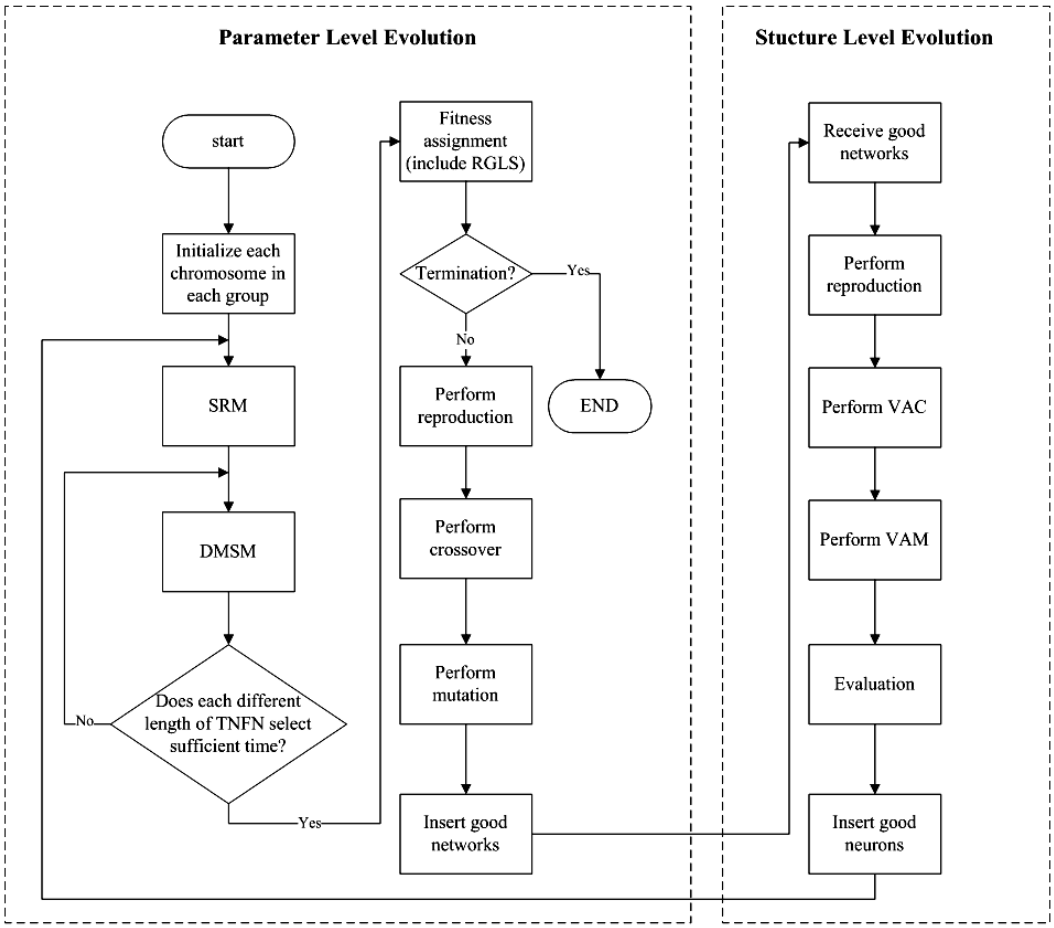


Figure 3-1: Learning process of RGLS-HCCA.

3.1 Parameter Level Evolution

In this subsection, we will discuss the parameter level evolution (PLE). In PLE, it aims to determine not only the suitable fuzzy rules of TNFN automatically but also the suitable individuals used to construct a TNFN. Regarding the former aim, PLE proposes a self-regulated mechanism (SRM) to determine the number of fuzzy rules automatically. SRM utilizes the probability vector to represent the suitability of TNFN with different fuzzy rules. In Fig. 3.2, SRM codes the probability vector P_{M_k} to represent the suitability of a TNFN with M_k rules where the number of fuzzy rules is limited to a certain bound, i.e., $[M_{\min}, M_{\max}]$. After the SRM is carried out, the probability of the suitable number of fuzzy rules in a TNFN will increase, and the probability of the unsuitable number of fizzy rules in a TNFN will decrease. Therefore, the number of fuzzy rules would be self-regulated. Regarding the later

aim, although SRM can determine the suitable number of rules, there is a need to identify the suitable groups used to select individuals to construct TNFN. More specifically, we should consider the well-performing groups of individuals to cooperate for producing better a generation than the current one. To face this issue, this study proposes a data-mining based selection method (DMSM) to determine which groups should be used to select individuals.

The DMSM involves two major parts, namely, finding frequent patterns and mining association rules. Regarding the former, the FP-growth algorithm [27] is used to find the frequent patterns that do not have candidate generation. Regarding latter, association rules are identified by using the confidence value. In DMSM, the FP-growth is used to find the sets of groups that occur frequently from transactions. In this paper, a "transaction" refers to the collection of groups that have good or bad performance. After the candidate sets of frequently occurring groups are found, DMSM identifies the association rules by setting the suitable confidence and uses the found association rules to determine M_k groups that are used to select M_k chromosomes that form TNFN with M_k rules. To this end, two actions are defined in this study: normal and explore actions. In the normal action, M_k groups are chosen randomly. In the explore action, M_k groups are chosen according to association rules. These two actions will be discussed in the procedures of PLE.

$P_{M_{\min}}$	$P_{M_{\min}+1}$	…	P_{M_k}	…	$P_{M_{\max}-1}$	$P_{M_{\max}}$

Figure 3-2: Coding the probability vector to represent the suitability of a TNFN with M_k rules.

To consider the structure of TNFN, unlike MGCSE encoding one fuzzy rule into a chromosome, PLE only encodes an antecedent part of a fuzzy rule into a chromosome. The consequent part of a fuzzy rule used in PLE is estimated by a regularized least square (RGLS) approach. The structure of chromosomes to construct TSK-type neuro-fuzzy networks (TNFNs) in PLE is shown in Fig. 3.3. In this figure, each antecedent part of a fuzzy rule represents a chromosome selected from a group, P_{size} denotes that there are P_{size} groups in a

population, and M_k indicates that there are M_k rules used in TNFN construction. In addition, PLE adopts the variable length of a combination of chromosomes with RGLS method to construct a TNFN. Thus, the length of combined chromosomes to construct TNFNs can be different.

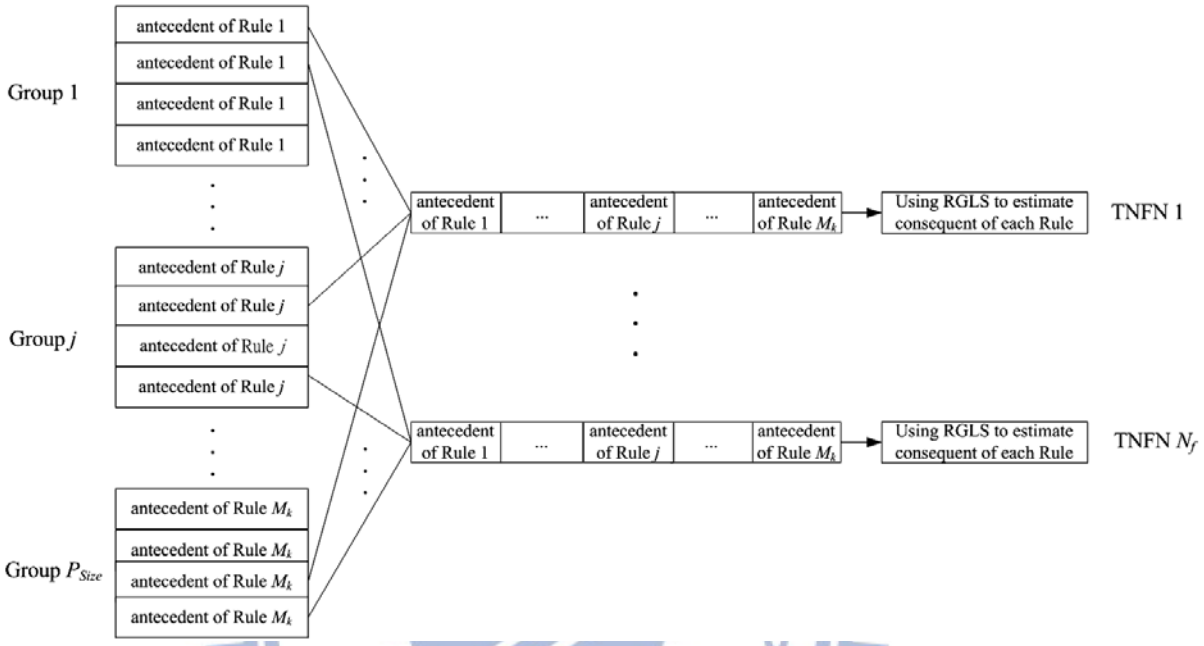


Figure 3-3: Structure of chromosomes to TNFN construction in PLE.

After discussing the structure of chromosomes to construct TNFNs, details of the coding step for PLE and RGLS method are described as follows:

(1) Coding Step:

The coding structure of chromosomes in the proposed PLE is shown in Fig. 3.4. This figure describes an antecedent part of a fuzzy rule that has the form in Eq. (2.5), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation of *i*th dimension and *j*th rule node, respectively. Besides, a pair of (m, σ) indicates a neuron in Layer 2 of a TNFN. Evolving an antecedent part of a fuzzy rule is likely to evolve a neuron which is a parameter of a neural network. Thus, the evolution of this level is called a parameter (i.e. neuron) level evolution.

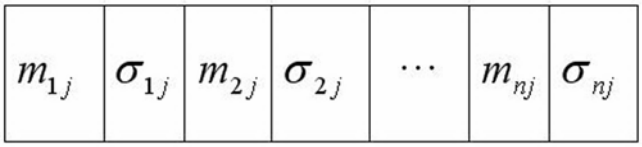


Figure 3-4: Coding an antecedent part of a fuzzy rule into a chromosome in PLE.

(2) RGLS method:

Assume a TSK-type neural fuzzy model composed of m fuzzy rules as the following form:

$$R_j: \textbf{ IF } x_1 \textbf{ is } A_1^j \textbf{ and} \cdots \textbf{and } x_n \textbf{ is } A_n^j \textbf{, THEN } y_j = w_o^j + w_1^j x_1 + \cdots + w_n^j x_n \textbf{,} \quad (3.1)$$

where $j = 1, \cdots, m$ and A_i^j is the linguistic part with respect the input *i* and *Rule j*. From Eq. (3.1), the output can be written as:

$$y = \frac{\sum_{j=1}^{m} u_j y_j}{\sum_{j=1}^{m} u_j} = \hat{u}_1 y_1 + \hat{u}_2 y_2 + \cdots + \hat{u}_m y_m, \quad (3.2)$$

where u_j is the firing strength of *Rule j*, and $\hat{u}_j = u_j / (u_1 + \cdots + u_m)$. Then it is possible to express the equation above into the form:

$$y = \hat{u}_1 (w_0^1 + w_1^1 x_1 + \cdots + w_n^1 x_n) + \cdots + \hat{u}_m (w_0^m + w_1^m x_1 + \cdots + w_n^m x_n) = aW, \quad (3.3)$$

where $W = [W_1^T \cdots W_m^T]^T, W_j = [w_0^j \cdots w_n^j]^T, j = 1, \cdots, m$, and

$a = [(\hat{u}_1 \ \hat{u}_1 x_1 \cdots \hat{u}_1 x_n) \ (\hat{u}_2 \ \hat{u}_2 x_1 \cdots \hat{u}_2 x_n) \cdots (\hat{u}_m \ \hat{u}_m x_1 \cdots \hat{u}_m x_n)]$.

Since y and a are known value, the only unknown value is the consequent part W. Suppose a given set of training inputs and desired outputs is $\{x(t), y_d(t)\}_{t=1}^{M}$. The Eq. (3.3) can be rewritten as:

$$AW = Y_d, \quad (3.4)$$

where $A = [a(1)\, a(2) \cdots a(M)]^T$.

In order to get the smooth estimation, the regularization is adopted. The approximation

solution can be written as follows:

$$\hat{W} = (A^T A + \lambda I)^{-1} A^T Y_d, \tag{3.5}$$

where λ is a regularization parameter which adjusts the smoothness. Thus, by getting Eq. (3.5), we finish the estimation of the consequent part of fuzzy rules. Based on this fact, this dissertation utilizes RGLS to calculate the consequent part of a TSK-type neural fuzzy network. This operation would not only reduce the number of parameters that must be trained but also increase the convergence rate of the evolutionary algorithm. Thus, the phenomenon of reducing training number and increasing convergence rate would promote the evolutionary algorithm to adapt the neural network to more complex tasks.

The learning process of PLE involves seven operators: initialization, self-regulated mechanism, data-mining based selection method, fitness assignment, reproduction, crossover, mutation, and insert good networks. The whole learning process is introduced below:

a. Initialization: Before we start the parameter level evolution, the initial groups of individuals should be generated. Thus, initial groups are generated randomly within a predefined range. The following formulations show how to generate the initial chromosomes in each group:

Deviation: $Chr_{g,c}[p]=random[\sigma_{\min}, \sigma_{\max}]$,

where $p=2, 4,\cdots, 2n$; $g=1, 2,\cdots, P_{size}$; $c=1, 2,\cdots, N_C$, (3.6)

Mean: $Chr_{g,c}[p]= random[m_{\min}, m_{\max}]$,

where $p=1, 3,\cdots, 2n-1$, (3.7)

where $Chr_{g,c}$ represents cth chromosome in the gth group, N_C is the total number of chromosomes in each group, p represents the pth gene in a $Chr_{g,c}$, and $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$ represent the predefined range to generate the chromosomes.

b. Self-regulated mechanism (SRM): To select fuzzy rules automatically, PLE proposes SRM to determine the suitability of TNFN models with different fuzzy rules. The

self-regulated mechanism encodes the probability vector P_{M_k} to stands for the suitability of a TNFN with M_k rules. In addition, in SRM, the minimum and maximum number of rules must be predefined to limit the number of fuzzy rules to a certain bound, i.e., [$M_{\min}$, $M_{\max}$]. The processing steps of SRM are described as follows:

Step 0. Initialize the probability vectors P_{M_k} :

$$\begin{aligned} &P_{M_k} = 0.5, \\ &\text{where} \quad M_k = M_{\min},\ M_{\min}+1, \cdots,\ M_{\max}. \end{aligned} \tag{3.8}$$

$$Accumulator = 0. \tag{3.9}$$

Step 1. Update the probability vectors P_{M_k} according to the following procedures:

(1) Evaluate the fitness value of TNFN with M_k rules:

$$\begin{aligned} &\text{if} \quad Fitness_{M_k} \geq (Best\_Fitness_{M_k} - ThreadFitnessvalue) \\ &\text{then} \quad fit_{M_k} = fit_{M_k} + Fitness_{M_k},\ fitcount = fitcount + 1 \end{aligned}, \tag{3.10}$$

where $Fitness_{M_k}$ represents the fitness value of TNFN with M_k rules, $Best\_Fitness_{M_k}$ represents the best fitness value of TNFN with M_k rules, fit_{M_k} is the sum of the fitness values of the TNFN with M_k rules and *fitcount* is a count as Eq. (3.10) satisfies.

(2) Calculate the average fitness value:

$$Avgfit_{M_k} = fit_{M_k} / fitcount, \tag{3.11}$$

$$Avg = \sum_{M_k = M_{\min}}^{M_{\max}} Avgfit_{M_k} / (M_{\max} - M_{\min} + 1), \tag{3.12}$$

where $Avgfit_{M_k}$ is a average value of fit_{M_k} and *Avg* represents the average fitness value in the whole population.

(3) Update the probability vectors:

$$Upt\_value_{M_k} = Avgfit_{M_k} \Big/ \sum_{M_k=M_{\min}}^{M_{\max}} Avgfit_{M_k}, \tag{3.13}$$

$$\begin{cases} P_{M_k} = P_{M_k} + (Upt\_value_{M_k} * r), & \text{if} \quad Avg \le Avgfit_{M_k} \\ P_{M_k} = P_{M_k} - (Upt\_value_{M_k} * r), & \text{otherwise} \end{cases}, \tag{3.14}$$

where $Upt\_value_{M_k}$ is a update value for M_k fuzzy rules and P_{M_k} is the probability vector, and r is a predefined ratio value.

Step 2. Determine the selection times of TNFN with different rules according to the probability vectors as follows:

$$Rp_{M_k} = (Selection\_Times) * (P_{M_k} / Total\_Velocy), \tag{3.15}$$

$$Total\_Velocy = \sum_{M_k=M_{\min}}^{M_{\max}} P_{M_k}, \tag{3.16}$$

where $M_k = M_{\min}, M_{\min+1}, \cdots, M_{\max}$, $Selection\_Times$ represents the total selection times in each generation and Rp_{M_k} represents the selection times of TNFN with M_k rules in one generation.

Step 3. In SRM, to prevent suitable selection times from falling into the local optimal solution, we uses two different procedures to update P_{M_k}. Such actions are defined as follows:

Procedure 1: update the probability vector

$$\begin{aligned} &\text{if} \quad Accumulator \le SRMTimes, \text{ then do Steps 1 to 2}, \\ &\text{if} \quad Best\_Fitness_g = Best\_Fitness, \text{ then } Accumulator = Accumulator + 1 \end{aligned} \tag{3.17}$$

where $SRMTimes$ is a predefined value, $Best\_Fitness_g$ represents the best fitness value of the best combination of chromosomes in the gth generation, and $Best\_Fitness$ represents the best fitness value of the best combination of chromosomes in the current generations.

To consider the amount of the computation in SRE, Eq. (3.14) is the major computation

process for SRM. Since the amount of the computation in Eq. (3.11)-(3.13) is not heavy (depend on the number of $Fitness_{M_k}$ and it is often not much), updating a P_{M_k} in Eq. (3.14) is also less computation. It implies that SRM is not a heavy computation procedure.

Procedure 2: initialize the probability vector

$$\text{if } Accumulator > SRMTimes, \text{ then do Step 0 and } Accumulator = 0, \quad (3.18)$$

If Eq. (3.18) is satisfied, it indicates that the suitable selection times may fall into the local optimal solution. At this time, the processing step of SRM should return to Step 0 to initialize the probability vector P_{M_k}.

c. The data-mining based selection method (DMSM):

After operating SRM, the selection times of TNFNs with different numbers of rules are determined. Thereafter, PLE performs the selection step, which involves the selection of groups and the selection of chromosomes. In selection of groups, this paper proposes DMSM to determine the suitable groups for chromosomes selection to form a TNFN.

In DMSM, suitable groups are selected according to the groups, which conduct from association rules that indicate good performance. To achieve these aims, DMSM utilizes the FP-growth [27] and the association rules mining. Regarding former, the FP-growth is used to identify frequently pattern. It was proposed by Han et al. [27], and it aims to find the frequently occurring patterns that do not have candidate generation. In the proposed DMSM, the FP-growth is used to find the frequently occurring groups from transactions. To reiterate, a transaction refers to a set of the groups that have good or bad performance. Regarding latter, after the frequently occurring groups are found, DMSM constructs the association rules by setting the suitable confidence. The association rules algorithm is a well-known approach in several fields [69-73]. The purpose of mining association rules is to identify good groups. After performing these two steps, the found association rules are utilized to selects M_k groups

that are used to choose chromosomes to form TNFNs with M_k rules. To prevent the selected groups from falling into the local optimal solution, DMSM uses normal and explore actions to select well-performed groups. The details of the DMSM are discussed below:

Step 1. Normal action:

If *Accumulator* don not exceed the *NormalTimes*, the current action is the explore action. The aims of this action include two parts: accumulate the transaction set and select groups which are described as follows:

Part 1**:** Accumulate the transaction set

The transactions are built, as in the following equations:

$$\begin{aligned} &\text{if} \quad Fitness_{M_k} \geq (Best\_Fitness_{M_k} - ThreadFitnessvalue) \\ &\quad Transaction_j[i] = TNFNRuleSet_{M_k}[i] \\ &then \\ &\quad Performance\ Index = g, \end{aligned} \tag{3.19}$$

$$\begin{aligned} &\text{if} \quad Fitness_{M_k} < (Best\_Fitness_{M_k} - ThreadFitnessvalue) \\ &\quad Transaction_j[i] = TNFNRuleSet_{M_k}[i] \\ &then \\ &\quad Performance\ Index = b, \end{aligned} \tag{3.20}$$

where $i = 1, 2, \cdots, M_k$, $M_k = M_{\min}, M_{\min}+1, \cdots, M_{\max}$, $j = 1, 2, \cdots, TransactionNum$, the $Fitness_{M_k}$ represents the fitness value of TNFN with M_k rules, $ThreadFitnessvalue$ is a predefined value, $TransactionNum$ is the total number of transactions, $Transaction_j[i]$ represents the *i*th item in the *j*th transaction, $TNFNRuleSet_{M_k}[i]$ represents the *i*th group in the M_k groups used for chromosomes selection, and $Performance\ Index = g$ and $Performance\ Index = b$ represent the good and bad performance, respectively. Hence, transactions have the form shown in Table 3.1. As shown in Table 3.1, the first transaction means that the three-rule TNFN formed by the first, fourth, and eighth groups have ”good” performance. In contrast, the second transaction indicates that the four-rule TNFN formed by the second, fourth, seventh, and the tenth groups have “bad” performance.

Table 3.1: Transactions in the DMSM.

Transaction index	Groups	Performance Index
1	1,4,8	g
2	2,4,7,10	b
…	…	…
TransactionNum	1,3,4,6,8,9	g

Part 2: Select groups

In the normal action, DMSM selects groups using the following equation:

$$\begin{aligned} &\text{if} \quad Accumulator \le NormalTimes \\ &\text{then} \quad GroupIndex[i] = Random[1,\ P_{Size}], \end{aligned} \tag{3.21}$$

where $i = 1, 2, \cdots, M_k$, $M_k = M_{\min}, M_{\min+1}, \cdots, M_{\max}$, *Accumulator* defined in Eq.(3.21) is used to determine which action should be adopted, $GroupIndex[i]$ represents the selected *i*th group of the M_k groups, and P_{Size} indicates that there are P_{Size} groups in a population in PLE. If the best fitness value does not improve for a sufficient number of generations (*NormalTimes*), then DMSM selects groups according to explore action.

Step 2. Explore action:

If *Accumulator* exceeds the *NormalTimes*, the current action switches to the explore action. The objective of this action is to adopt the notion of DMSM to explore suitable groups in transactions. The major operations of DMSM include FP-growth performing, association rules generating, and suitable groups selecting. The details of these three operations are presented below.

i. FP-growth performing

In this operation, only good groups, whose performance index showed "g" in Table 3.1, are performed with FP-growth and bad groups are skipped. Thus, frequently occurring groups can be found according to the predefined *Minimum_Support*, which stands for the minimum fraction of transactions containing the item set. After *Minimum_Support* is defined, data mining using FP-growth is performed (detail procedures of FP-growth can be found in [27]).

In FP-growth, frequently occurring groups can be found by exploring the FP-tree [27]. After exploring the frequently occurring groups in the FP-tree, FP-growth data mining is completed by the concatenation of the suffix group [27] with the generated frequently occurring groups. Thus, in this paper, *frequent groups* denote the frequently occurring groups found by FP-growth algorithm.

ii. Association rules generating

Once the frequently occurring groups are found, we can produce association rules from these frequent ones. For the purpose of identifying the association rules with good performance, the frequent groups must combine the groups owing bad performance shown in Table 3.1 to count the confidence degree. The confidence degree can be computed by the following formula:

$$\begin{aligned} &confidence(\,frequent\ groups \Rightarrow good\,) \\ &= P(good \mid frequent\ groups) \\ &= \frac{supp(\,frequent\ groups \cup good\,)}{supp(\,frequent\ groups \cup good\,) + supp(\,frequent\ groups \cup bad\,)}, \end{aligned} \tag{3.22}$$

where $P(good \mid frequent\ groups)$ is the conditional probability, $frequent\ groups \cup good$ or *bad* means the union of frequent groups and good or bad performance, and *supp(frequent groups* $\cup$ *good* or *bad)* stands for the counts of *frequent groups* with good or bad performance occurring in transactions. Then the rule is valid if

$$confidence(\,frequent\ groups \Rightarrow good\,) \geq minconf\,, \tag{3.23}$$

where *minconf* represents the minimal confidence given by user or expert. Hence, we can infer that if a rule satisfies Eq. (3.23), then the frequent groups can be viewed as the suitable groups, otherwise they would be unsuitable groups. For instance, if the confidence of {1,3,6}=>{g} is bigger than the minimum confidence, then we construct this association rule. This rule indicates that the combination of the first, third, and sixth groups results in "good" performance. After doing so, the frequent groups are conduct to the association rules and generate the *AssociatedGoodPool* which contains all frequent groups satisfied Eq. (3.23).

iii. Suitable groups selecting

After the association rules are identified, DMSM selects groups according to the association rules. The group indexes are selected from the associated good groups as the following equations:

$$\begin{aligned} &\text{if} \quad NormalTimes < Accumulator \le ExploreTimes \\ &\text{then} \quad GroupIndex[i] = w, \\ &\text{where} \quad w = Randm[1, P_{size}] \ \ and\ w \in GoodItemSet[q] \\ &GoodItemSet[q] = Random[AssociatedGoodPool], \end{aligned} \tag{3.24}$$

where $q = 1, 2, \cdots, AssociatedGoodPoolNum$, $i = 1, 2, \cdots, M_k$, $M_k = M_{\min}, M_{\min+1}, \cdots, M_{\max}$, $ExploreTimes$ is a predefined value that judge to perform the exploring action, $AssociatedGoodPool$ represents the sets of good item set that obtain from association rules, $AssociatedGoodPoolNum$ presents the total number of sets in $AssociatedGoodPool$ and $GoodItemSet[i]$ presents a good item set that select from $AssociatedGoodPool$ randomly. In the Eq. (3.24), if M_k greater than the size of $GoodItemSet$, remain groups are selected by Eq. (3.21).

Step 3. If the best fitness value does not improve for a sufficient number of generations (*ExploreTimes*), DMSM selects groups based on the normal action (Step 1).

Step 4. After the M_k groups are selected, M_k chromosomes are selected from M_k groups as follows:

$$ChromosomeIndex[i] = q, \tag{3.25}$$

where $q = Random[1, N_c]$, $i = 1, 2, \cdots, k$, N_c represents the total number of chromosomes in each group, and $ChromosomeIndex[i]$ represents the index of a chromosome that is selected from the *i*th group.

d. Fitness assignment: To assign a fitness value of an individual, the following detailed steps in the fitness value assignment are performed:

Step 1. Take the DMSM selected M_k antecedent part of fuzzy rules and use RGLS method to calculate the consequent part of fuzzy rules. These two actions are repeated to construct TNFNs Rp_{M_k} times from M_k groups with size N_C.

Step 2. Evaluate every TNFN that is generated from Step1 to obtain a fitness value. In this paper, the fitness value is designed according to the following formulation:

$$\text{Fitness Value}=1/(1+E(y,\bar{y})), \tag{3.26}$$

$$\text{where } E(y,\bar{y})=\sum_{i=1}^{N}(y_i-\bar{y}_i)^2, \tag{3.27}$$

where y_i and $\bar{y}_i$ represents the desired and predicted values of the *i*th output, respectively, $E(y,\bar{y})$ is an error function and N represents the number of the training data in each generation.

Step 3. Accumulate the divided fitness value to the antecedent part of fuzzy rules with their fitness value records.

Step 4. Divide the accumulated fitness value of each chromosome from M_k groups by the number of times that it has been selected.

e. Reproduction: Reproduction is a procedure of copying individuals according to their fitness value. This study adopted our previous research-elite-based reproduction strategy (ERS) [15] to perform reproduction. In ERS, every chromosome in the best combination of M_k groups must be kept by performing reproduction step. In the remaining chromosomes in each group, this study uses the roulette-wheel selection method [74] and [75] for this reproduction process. The well-performed chromosomes in the top half of each group [18] proceed to the next generation. The other half is created by executing crossover and mutation operations on chromosomes in the top half of the parent individuals.

f. Crossover: Although DMSM can be used to select suitable individuals for TNFN construction, it does not create any new individual. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurs for a selected pair with a crossover rate. In this paper, a two-point crossover strategy [76] is adopted and shown in Fig. 3.5. In the figure, exchanging the site's values between the selected sites of individual parents creates new individuals. The advantage of the two-point crossover is its ability of introducing a higher degree of randomness into the selection of genetic material [77]. Moreover, such crossover strategy generally yields better performance than one-point crossover due to its larger search step size [76].

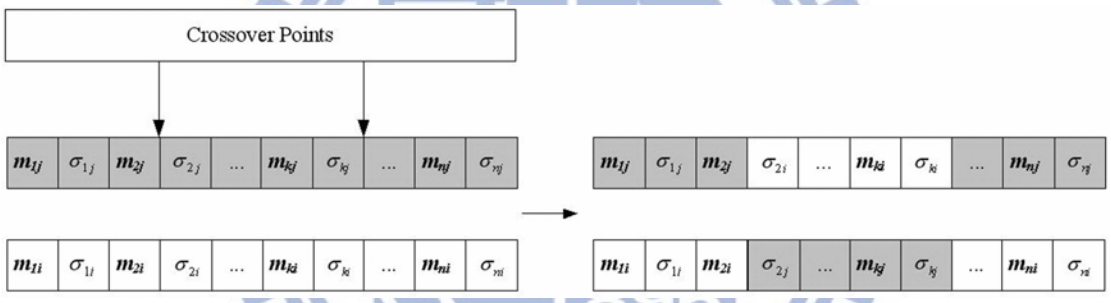


Figure 3-5: Two-point crossover.

g. Mutation: Although the crossover strategy produces many new strings, these strings do not provide any new information to every group at the site of an individual. Mutation can randomly alter the allele of a gene. In this paper, to emphasize the capability of the SRM and the DMSM, the PLE attempts to simplify the mutation operation. Uniform mutation [74] and [78] is therefore adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable. The benefits of uniform mutation are not only to generate new information into a population but also to keep a highly diverse array of information, which is useful to the fitness of individuals [79].

h. Insert good networks: Since there are "*Selection_Times*" networks constructed in every generation, the fitness value of each network is recorded and compares it with the structure

evolution level. If the fitness of the network is better than the worst network in the structure evolution level, then this network is inserted into the structure evolution level.

To consider the termination criterion, if the learning steps meet one of the following conditions, RGLS-HCCA is terminated and output the final results.

(1) The number of generations reaches a predefined maximal iteration value.

(2) Fitness value is greater than a fitness threshold.

Consequently, the whole learning process of PLE is summarized in Fig. 3.6.

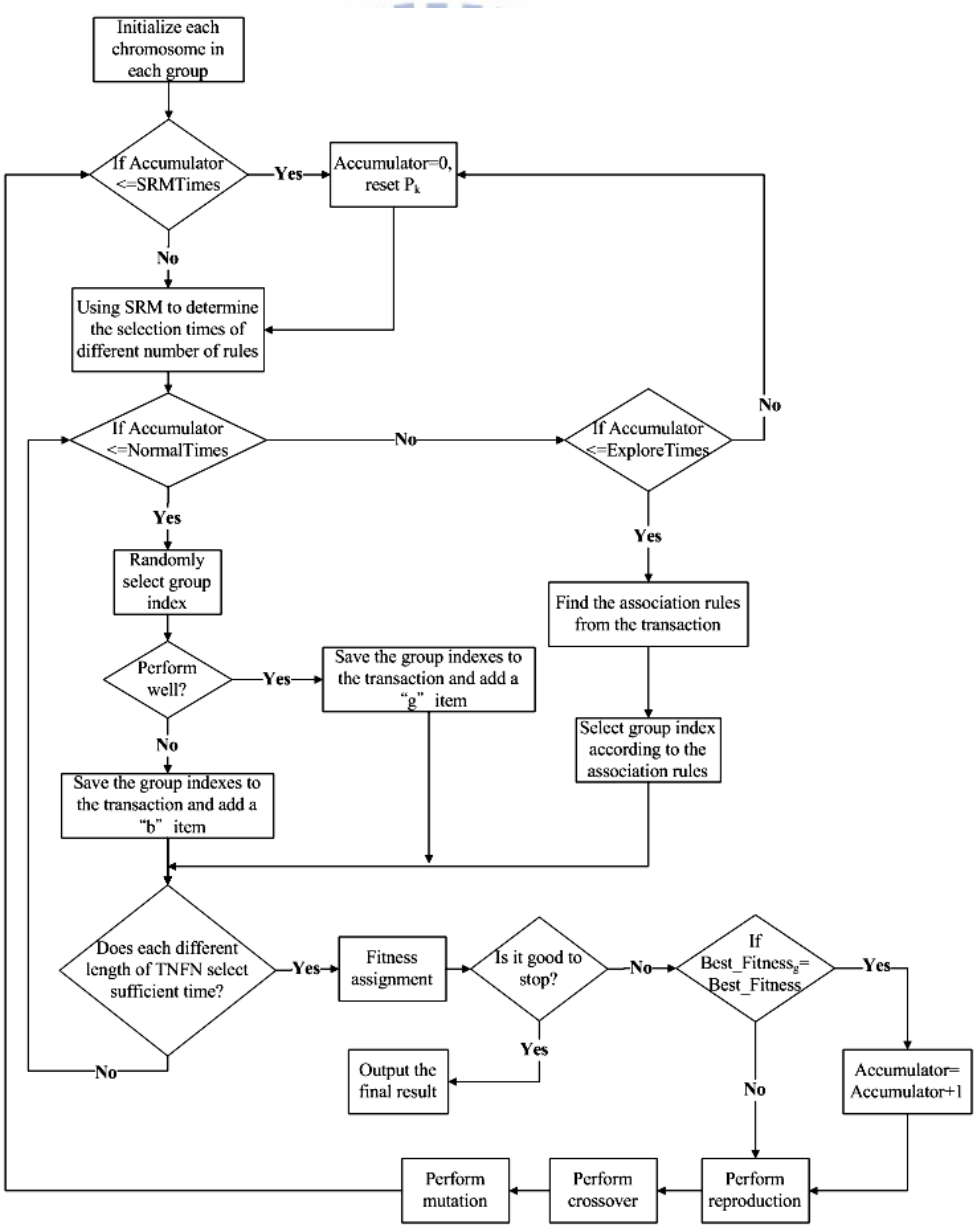


Figure 3-6: The learning process of PLE.

3.2 Structure Level Evolution

In this subsection, the structure level evolution (SLE) is discussed. The main processes of SLE involve six operations: receive good networks, reproduction, variable antecedent-part crossover, variable antecedent-part mutation, evaluation, insert good neurons. The details of these operations are described as follows:

a. Receive good networks: Before the structure evolution starts, we receive N well-performed networks from parameter level evolution to be chromosomes. The coding structure of chromosomes in the structure level evolution is shown in Fig. 3.7. In this figure, each block of a chromosome describes an antecedent part of a fuzzy rule that has the form in Eq. (2.5), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation of ith dimension and jth rule node, respectively. The consequent part of a fuzzy rule is skipped to encode into chromosomes since regularized least squares is proposed to estimate the consequent part. After that, we sort the chromosomes to prepare for performing reproduction.

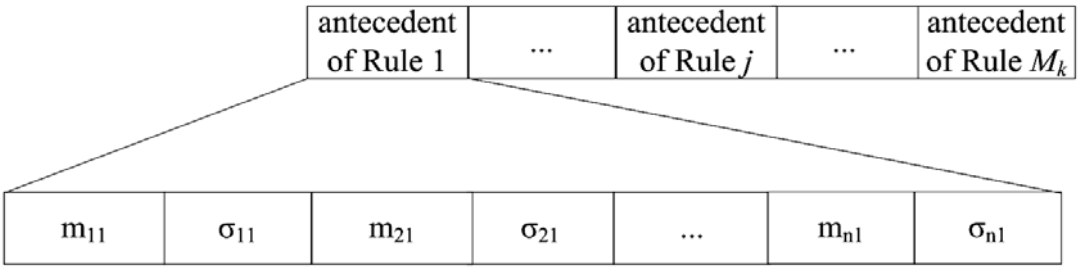


Figure 3-7: The coding the antecedent part of fuzzy rules into a chromosome in the structure level evolution.

b. Reproduction: Reproduction is a process in which string are copied according to their fitness value. In this step, roulette-wheel selection method is adopted for the reproduction process. The well-performed chromosomes in the top half of each group proceed to the next generation. The other half is generated by executing variable two-part and variable two-part operations on chromosomes in the top half of the parent individuals.

c. Variable antecedent-part crossover: In the structure level evolution, the variable

antecedent-part crossover (VAC) is proposed to perform crossover. In VAC, two parents are selected by using the roulette-wheel selection method [74]. Because the selected parents may be with different length, the misalignment of individuals must be avoided in the crossover operation. Thus variable antecedent-part crossover is proposed to address this problem. The antecedent part means that only the antecedent of fuzzy rule is performed crossover operation. In VAC, two-point crossover [76] is adopted to execute crossover. Thus, new individuals are generated by exchanging the site's values between the selected sites of the parents' individuals. In VAC, to avoid the misalignment of individuals in the crossover, the selection of the crossover points would not exceed the shortest length chromosome of two parents. Two individuals with different lengths using VAC operation are shown in Fig. 3.8. where AR_j represents the parameters of the antecedent part of the jth rule in the TNFN, and R_k represents there are k fuzzy rules in a TNFN. After performing the VAC, the new offspring can replace the individuals with poor performance.

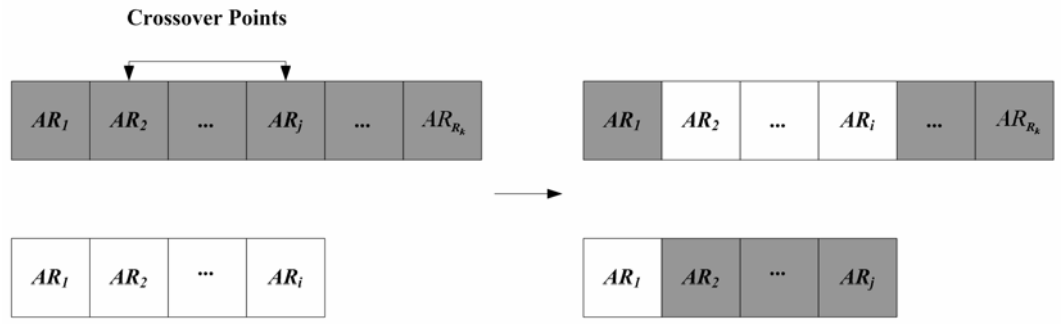


Figure 3-8: Variable antecedent-part crossover operation in the structure level evolution.

d. Variable antecedent-part mutation: The mutation operator can randomly alter the allele of a gene. It provides new information to every population at the site of an individual. In the structure level evolution, the variable antecedent-part mutation (VAM) is adopted to perform the mutation operation. The benefit of VAM is to be applied to different length of chromosomes. The VAM operation of each individual is shown in Fig. 3.9 where AR indicates antecedent part of fuzzy rule In VAM, uniform mutation [78] is adopted, and the

mutated gene is drawn randomly from the domain of the corresponding variable.

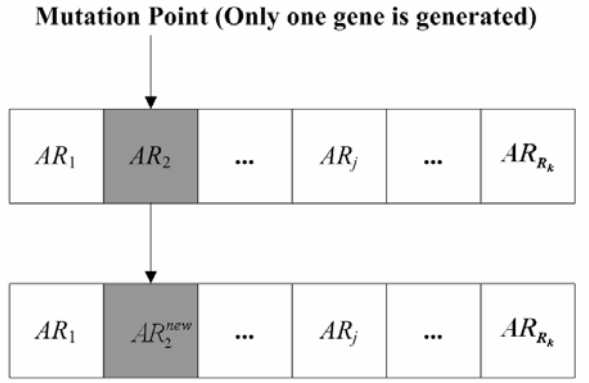


Figure 3-9: Variable antecedent-part mutation operation in the structure level evolution.

e. Evaluation: The evaluating step is to evaluate the fitness of each chromosome that has not already been evaluated in a population. The higher a fitness value indicates the better performance. Since each chromosome only includes the antecedent part of fuzzy rules, the consequent part of fuzzy rules is not defined. Thus, similar to the fitness assignment in PLE, the RGLS method is used to estimate the consequent part of fuzzy rules. After the antecedent and consequent part are determined, the TNFN is constructed. Then, evaluate every TNFN to obtain a fitness value. In this paper, the fitness value is designed according to Eq. (3.26) and (3.27).

f. Insert good neurons: After the evaluation operation, if a network has a higher fitness value than the best network in the parameter level, then insert the neurons into the corresponding groups of subpopulation in the parameter level evolution.

Thus, the whole learning process of SLE is summarized in Figure 3.10.

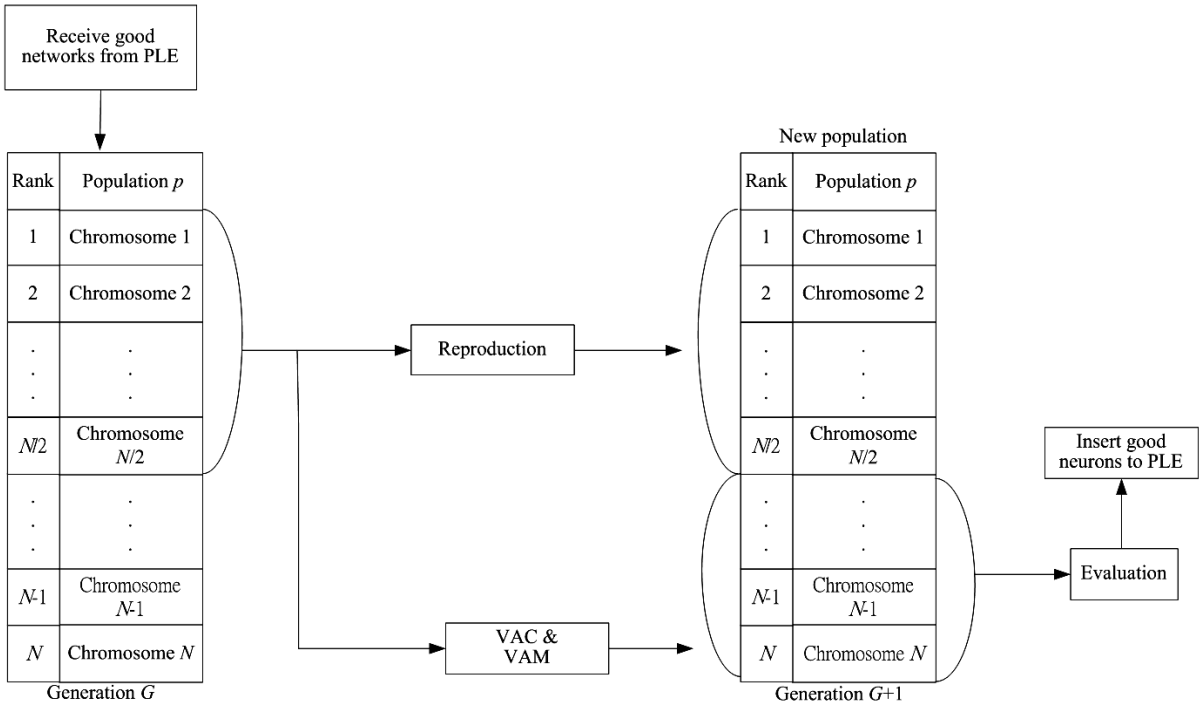


Figure 3-10: Whole learning process of SLE.

In short, the purpose of SLE is to reserve the good combinations of fuzzy rules produced by PLE and evolve the structure of the produced neural fuzzy networks. Thus, the utility of SLE is to fine tune the evolved results of PLE. To this end, PLE would be a major evolution to evolve TNFNs and it affects the effectiveness of the proposed RGLS-HCCA model.

Chapter 4
Image Alignment Applications

To demonstrate the applicability of RGLS-HCCA to real world problems, two image alignments tasks are taken to consideration: 2D image alignment and 3D image alignment. For a 2D image alignment problem, it is considered of great importance in numerous industrial applications including automatic visual inspection, electronic component assembly automation, circuit board inspection, and robotic machine vision. Among them, an automatic visual inspection system [80-82] is one of the most important fields for seeking an accurate geometric transformation to align images. To this end, neural network based methods have widespread to face this problem. The reason is that such methods often extract global features from images and feed them into a trained neural network to estimate geometric transformations parameters. In this dissertation, RGLS-HCCA can be used to develop a neural fuzzy network-based image alignment system to demonstrate high performance.

For a 3D image alignment problem, it is considered a critical step in object recognition [83], surface reconstruction [84], and image-guided surgery [85]. Two major concerns for the alignment task are execution time and alignment accuracy. Recently, neural network-based methods have become very popular due to their high efficiency. Thus, a TNFN-based coarse-to-fine 3D surface alignment scheme is proposed in the current dissertation.

In this chapter, two subsections are used to introduce the proposed alignment systems. Section 4.1 presents how the proposed 2D image system works. In Section 4.2, the proposed TNFN-based coarse-to-fine 3D image alignment system is described.

4.1 2D Image Alignment System

The flow chart of the proposed image alignment algorithm, which consists of off-line and on-line procedure, is illustrated in Fig. 4.1. During the off-line procedure, the synthesized

training images are created by applying the reference image to affine transformations with randomly selected parameters, and then use the Gabor-weighted gradient orientation histograms (Gabor-WGOH) descriptor to represent these training images as feature vectors. Finally, the feature vectors and desired targets are employed to train a CNFN using RGLS-HCCA. During the executing phase, the sensed image is sent to the Gabor-WGOH descriptor to extract a feature vector and then feed it into the RGLS-HCCA trained CNFN to estimate affine transformations parameters. Then, the estimated parameters are taken to align the sensed image with the reference image. The following subsections will introduce the process of the proposed 2D image alignment scheme.

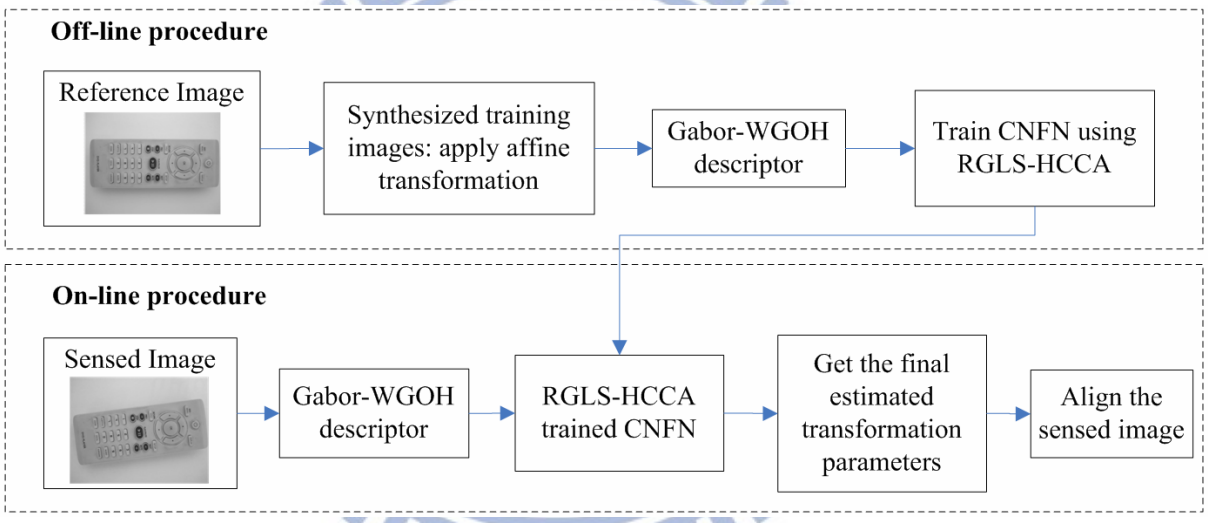


Figure 4-1: Flow chart of the proposed image alignment algorithm.

4.1.1 Off-line Procedure

The objective of the off-line procedure is to train CNFN. Four main parts in the procedure are synthesized training images creating, Gabor-WGOH descriptor generating, self-organized training data yielding, and CNFN training. These parts are described as follows.

(a) Synthesized Training Images Creating

The synthesized training images can be generated by applying various combination of translation, rotation, and scaling transformations within a predefined range. The

transformation model is affine transformation which can be described by the following matrix equations:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = s\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x_1 - x_c \\ y_1 - y_c \end{pmatrix} + \begin{pmatrix} x_c + \Delta x \\ y_c + \Delta y \end{pmatrix}, \tag{4.1}$$

where (x_1, y_1) indicates the original image coordinate, (x_2, y_2) indicates transformed image coordinate, s is a scaling factor, $(\Delta x, \Delta y)$ is a translation vector, θ is a rotation angle, and (x_c, y_c) is the center of rotation.

(b) Gabor-WGOH Descriptor Generating

The WGOH descriptor has been compared by several global descriptors [40-42, 86-88] using a nearest-neighbor search of the feature vector proposed by [89] and [90]. Thus, WGOH was proven a good descriptor [86] and [90], inspired by Scale Invariant Feature Transform (SIFT) descriptor [91], and presented by Bradley et al. to show its high speed [92]. The main idea of the WGOH is that it calculates the orientation histograms within a region, and uses the magnitude of the gradient at each pixel and the 2D Gaussian function to weight the histogram [86]. Therefore, for the WGOH descriptor, there are four steps for representing an image:

1 For each image, we capture the template window, whose location is at the center of the image, to be a place of extracting features. Within the window, we divide the length and width of the window into 4 equal parts to form 4×4 grids. Each grid is considered a sub-image. Thus the template window can be split into 4×4 sub-images.

2 On each pixel of the sub-image $I(x, y)$, the gradient magnitude $m(x, y)$, and orientation $\theta(x, y)$ is computed using pixel difference which the equations can be written as

$$m(x, y) = \sqrt{(I(x+1, y) - I(x-1, y))^2 + (I(x, y+1) - I(x, y-1))^2}, \tag{4.2}$$

$$\theta(x, y) = \tan^{-1}((I(x, y+1) - I(x, y-1))/(I(x+1, y) - I(x-1, y))). \tag{4.3}$$

3 Calculate the 8-bin orientation histograms (each bin cover 45 degree) within each sub-image which are weighted by the gradient magnitude, and the Gaussian function.

4 Concatenate 8-bin histograms of 16 sub-images into a 128-element feature vector, and normalize it to a unit length. To reduce strong gradient magnitudes, the elements of the feature vector are limited to 0.2, and this vector is normalized again.

Consequently, each image can be represented by a 128-elemet feature vector. Fig. 4.2 illustrates an example of WGOH computation steps. However, using pixel difference to compute the gradient is sensitive to noise. To avoid such sensitivity, Moreno et al. combined a Gabor filter with WGOH descriptor to suppress noise [93]. Based on this fact, we adopt the Gabor-WGOH descriptor for representing an image.

Because the 128-elemet feature vector is still too high to train a TSK-type neuro-fuzzy network, there is a requirement of finding a dimensionality reduction method to lower the dimension of the feature vector. In order to lower the dimension of feature vector, we further employed principal component analysis method (PCA) to reduce the 128-elemet feature vector into a 33-element one. Therefore, each image can be represented by a 33-elemet feature vector.

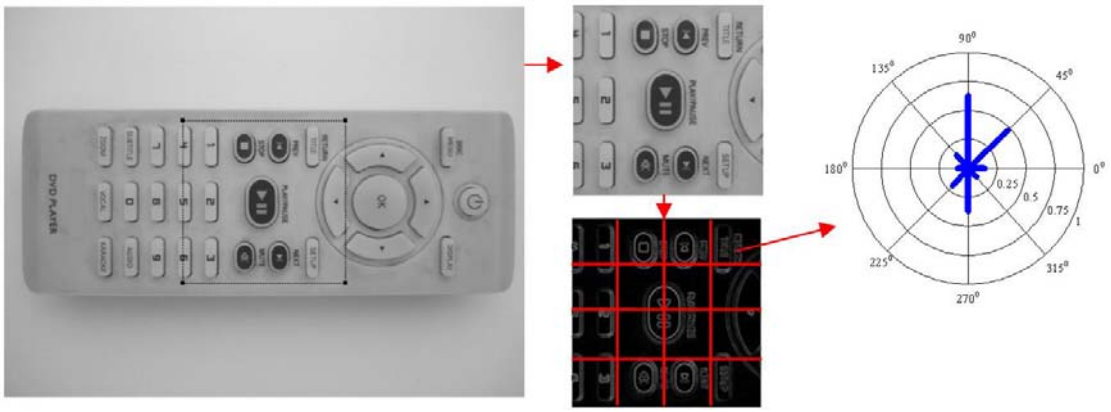


Figure 4-2: Steps for creating a WGOH feature vector.

(c) Yielding self-organized training data

After describing the Gabor-WGOH descriptor, this paper proposes a self-organized training data-creating method to provide an appropriate training data set for training neural

fuzzy networks. The major advantages of the proposed training data-creating method are that it can prevent the generation of the redundant data and supply a self-organized training data set for training a neural fuzzy network efficiently. The steps for yielding the self-organized training data are as follows:

Step 1: First, generate a small training data set $\{S_{train}\}$.

Step 2: Then, utilize the training data set to train a neural fuzzy network.

Step 3: Input a fixed number of testing data set $\{S_{test}\}$ into the neural network to create the alignment error $\{E_{test}\}$. Check each error $\{E_{test}(i)\}$:

If $E_{test}(i) > PdError$, then $\{S_{test}(i)\} \xrightarrow{\text{insert}} \{S_{train}\}$ and $ErAcc = ErAcc + 1$.

for $i = 1,2,\cdots, N_{test}$, (4.4)

where *PdError* is the predefined error, *ErAcc* is the accumulator of large error counts, and N_{test} is the number of the test data set.

Step 4: If $ErAcc < t_{er}$, then accumulate the *LoopNum*=*LoopNum*+1. Otherwise, set *LoopNum*=0. The symbol t_{er} indicates the threshold of the error accumulator, and *LoopNum* means the accumulating number of loop.

Step 5: If *LoopNum* > loop threshold t_{loop}, terminate the training and output the training set $\{S_{train}\}$. Otherwise, go to step 2 to run recursive training.

In Step 3, the insert testing data is the data that the neural fuzzy network does not perform well. Therefore, inserting such data can enhance the learning ability of the neural network and prevent the selection of the redundant training data. Moreover, from Step 5, $LoopNum > t_{loop}$ means that the amount of training data set has converged. At this time, it also indicates that the training data set is self-organized. Thus, we can utilize the self-organized training data-creating method to provide the training data for training CNFNs.

(d) Cooperative Neural Fuzzy Network (CNFN) training

The notion of the cooperative neural fuzzy network is to combine several networks to all

cooperate in adapting to a large range of affine transformation. The aim of this operation is to improve the problem of applying a large range of affine transformation to traditional one-stage neural network which can cause a large amount of training data; such a network is difficult to train. The cooperative networks can be seen a coarse-to-fine aligning the captured image with reference image.

Figure 4.3 presents the process of cooperative neural fuzzy network. From this figure, each stage deals with a certain range of affine parameters and they cooperate to get a large range of affine parameters. As input an image with an unknown pose, the cooperative neural fuzzy network would gradually reduce the pose difference between the input and reference image. Thus the final pose with respect to the reference image can be written as the following equation:

$$P_{final} = P_1 + P_2 + \cdots + P_N, \tag{4.5}$$

where P_1, P_2, and P_N indicates the estimated pose from 1st, 2nd, and Nth stage of the neural network.

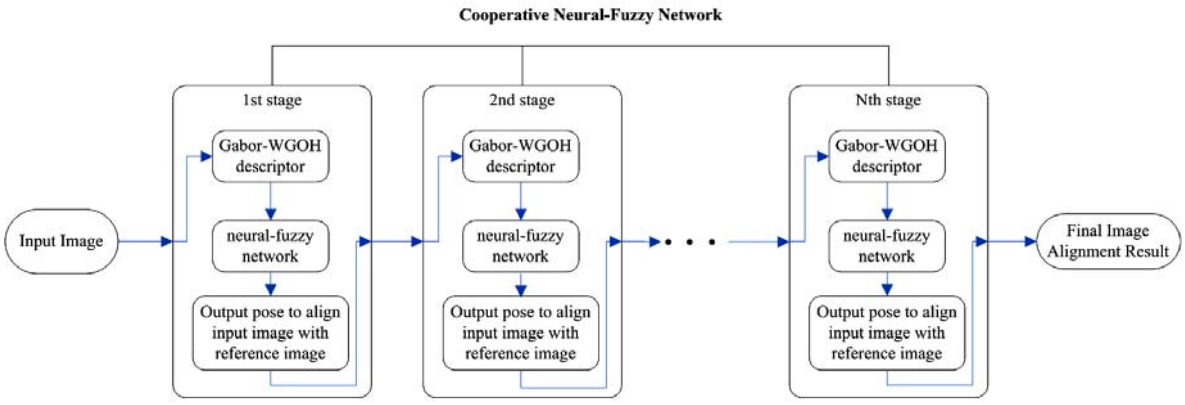


Figure 4-3: Process of cooperative neural fuzzy networks.

To perform training CNFN with providing the training data, this study proposes RGLS-HCCA to accomplish it. In CNFN, once the dynamic image alignment range of each stage has been determined, each network can be trained independently. Thus, the training process of each stage of CNFN is similar, and the only difference is its training parameters. To

this end, RGLS-HCCA is used to train each stage of CNFN to estimate the pose with respect to the input image.

4.1.2 On-line Procedure

In the on-line phase, the sensed image (input image) is sent to the Gabor-WGOH descriptor to extract a feature vector and then feed it into RGLS-HCCA trained CNFN to estimate transformation parameters, which include the scaling factor s, rotation angle θ, and translation (Δx, Δy), to be taken into aligning images. More specifically, the proposed CNFN performs *N*-stages of neural fuzzy network (as shown in Fig. 4.3) to gradually align the sensed image with the reference image. Thus, the image alignment error will be reduced stage by stage and finally get the best aligning pose with the reference image.

4.2 3D Image Alignment System

According to Chapter 2, each pixel in a 3D image can be considered as a 3D point cloud data with respect to the laser scanner. Thus, a 3D image is viewed as a collection of 3D point clouds and these point clouds can represent arbitrary 3D surface. Based on this fact, aligning two 3D images is likely to align two 3D surfaces and other researches also call 3D image alignment to be 3D surface alignment (or registration) [45]. In this dissertation, the objective of a 3D image alignment is to align a captured 3D image (i.e. 3D surface) of an object in an arbitrary view with the 3D surface of the reference model.

Figure 4.4 presents the flow diagram of the proposed 3D image alignment system. In the learning phase, two data flows are performed for training TNFNs to adapt two levels of image alignment: one for coarse alignment and the other one for fine alignment. In the executing phase, the trained TNFNs are utilized to implement a coarse-to-fine 3D image alignment task. These two phases are explained in detail to show how the process of 3D image alignment works.

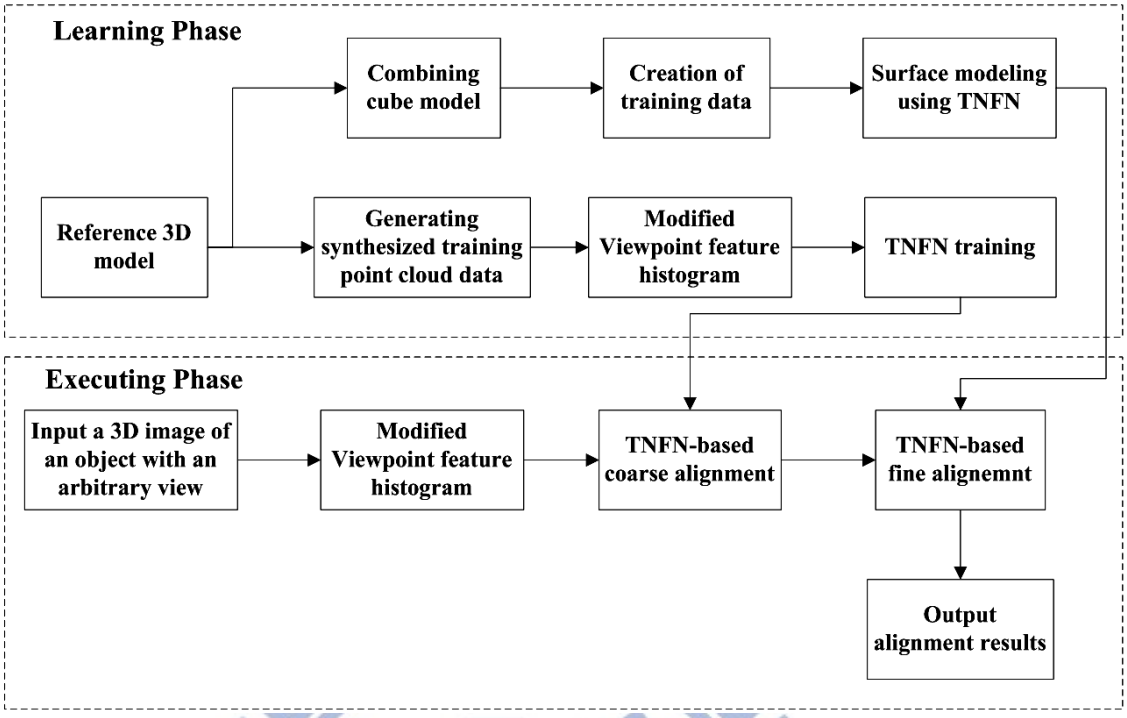


Figure 4-4: Flow diagram of the proposed 3D image alignment system.

4.2.1 Learning Phase

The objective of the learning procedure is to train two TNFNs for applying coarse-to-fine 3D image alignment. These two major parts of the procedure are the coarse alignment learning and the TNFN-based surface modeling. These parts are described in the following contents.

(a) Coarse alignment learning

The goal of coarse alignment is to determine an approximate rigid transformation that coarsely aligns the reference model with the input point clouds. The coarse alignment must be quick to provide a good initial transformation for the fine alignment task. Thus, TNFN is utilized to learn any case of rigid transformation within the predefined range. Once the training of TNFN is completed, input arbitrary view of point clouds would yield the estimate pose with respect to the reference model. Therefore, the executing phase of the TNFN is simple and efficient.

The procedures of proposed coarse alignment learning involves generating synthesized training point cloud data, yielding the modified viewpoint feature histogram (MVFH), and

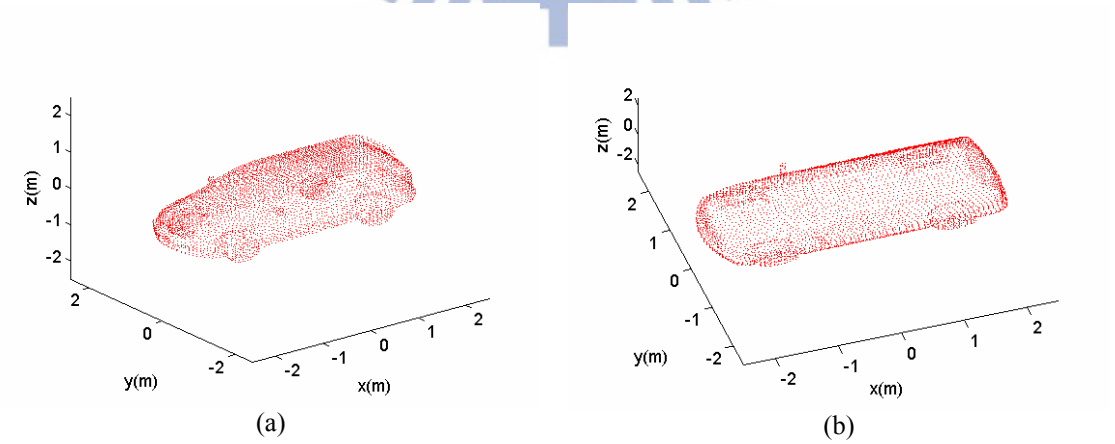
training the TNFN. These operations are introduced as follows:

(i) Generating synthesized training point cloud data

Figure 4.5 depicts the point cloud data of the reference model. The reference model is an integrated model constructed by collecting multi-views of point cloud data. To generate the synthesized training point cloud data, various combinations of translation and rotation transformations within a predefined range are applied in the reference model. The transformation can be considered a rigid transformation, which can be written as follows:

$$m = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \cdot s + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{4.6}$$

where R is a rotation matrix, T is a translation vector, s is an original set of point cloud data and m is a transformed set of point cloud data. Furthermore, to simulate the real case in a 3D scene, point cloud data that cannot be seen in the viewpoint direction are eliminated. Figure 4.6 presents an example of the simulated training data. As shown in this figure, the point cloud data is only a partial of reference model and the unseen point clouds have been eliminated. Therefore, after the training point data has been generated, the following operation is to extract the feature of the point cloud data.



(a) (b)

Figure 4-5: Point cloud data of the reference model: (a) Front view and (b) Top view

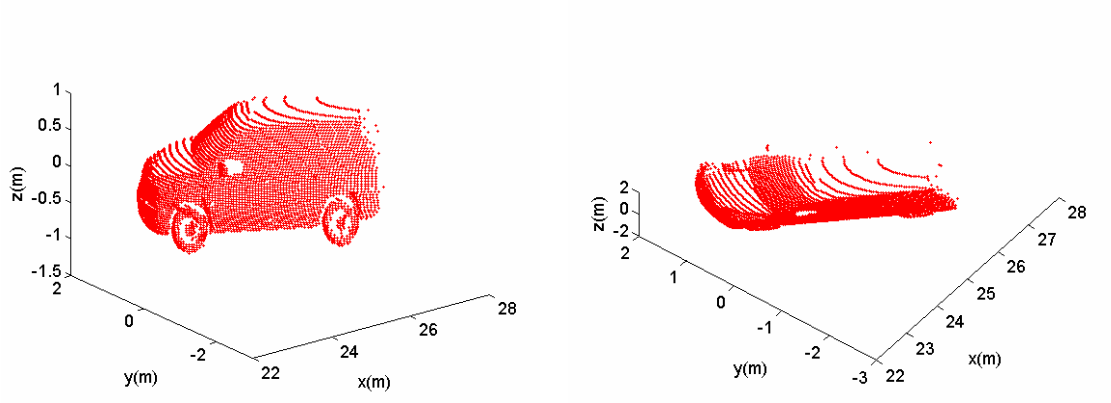


Figure 4-6: Example of the simulated training data: (a) Front view and (b) Top view.

(ii) Modified Viewpoint Feature Histogram

Modified Viewpoint Feature Histogram (MVFH) is the modification of Viewpoint feature histogram (VFH), which was presented by Rusu et al. [94], to show its computationally efficient 3D feature. To introduce VFH in advance, this descriptor is computed by accumulating a histogram of the angles between the central viewpoint direction and each normal of point cloud. Figure 4.7 illustrates the idea of VFH.

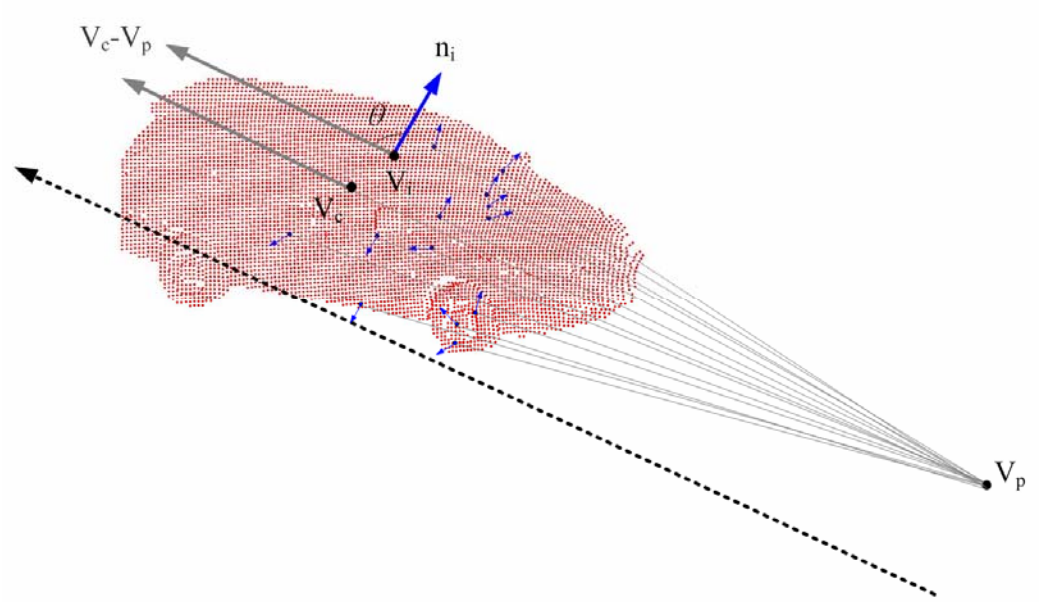


Figure 4-7: Creation of viewpoint feature histogram.

Suppose the central point is V_c and the viewpoint is V_p. Then the central viewpoint direction is $V_c - V_p$. Thus the angle θ between the central viewpoint direction ($V_c - V_p$) and

each normal n_i of point cloud V_i can be computed by the following equation:

$$\theta = \cos^{-1}\left(\frac{(V_c - V_p) \bullet n_i}{\| V_c - V_p \| \cdot \| n_i \|}\right). \tag{4.7}$$

Thereafter, the *N*-bin orientation histograms (each bin cover 180/*N* degree) can be calculated by accumulating the angle described in Eq. (4.7). The histogram in each bin is normalized by dividing the total number of point clouds. Thus, such histogram indicates the percentage of point clouds falling in each bin. However, in 3D surface alignment tasks, the viewpoint direction angle to represent the 3D surface might be not appropriate because VFH in some much different view angles would yield similar feature, especially in the case of symmetrical objects with 180 degree view angle difference. Figure 4.8 illustrates an example of similar VFH with much different view angle. As shown in this figure, the object is at two much different viewpoints but they have similar viewpoint feature histogram.

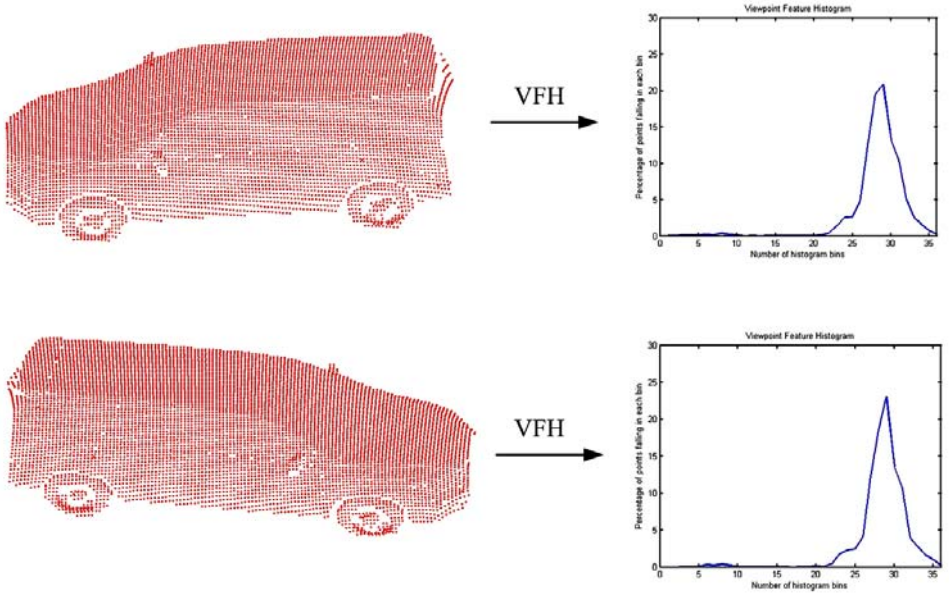


Figure 4-8: Example of similar viewpoint feature histograms in much different view.

Although Rusu et al. used ideas from Point Feature Histogram (PFH) [95] to assemble with VFH, the PFH descriptor is a local feature, which indicates PFH to be view independent, such that the combined VFH-PFH still cannot solve the problem presented in Fig. 4.8. In our 3D surface alignment case, the captured 3D feature must be view dependent. The reason is

that the 3D feature is utilized to identify the view angle and if the 3D feature is view independent, the captured feature would be similar in each view such that it is impossible to differentiate the exact view angles in an object. Regarding this fact, we modify the original viewpoint feature histogram by calculating another viewpoint direction related angle to improve the viewpoint feature histogram. Then we name such viewpoint direction as modified viewpoint feature histogram (MVFH). Figure 4.9 presents a diagram that describes two viewpoint direction related angles where θ is the original angle used by VFH, ϕ is new added angle used by MVFH, the central point is V_c, the viewpoint is V_p, and V_i is a certain 3D point.

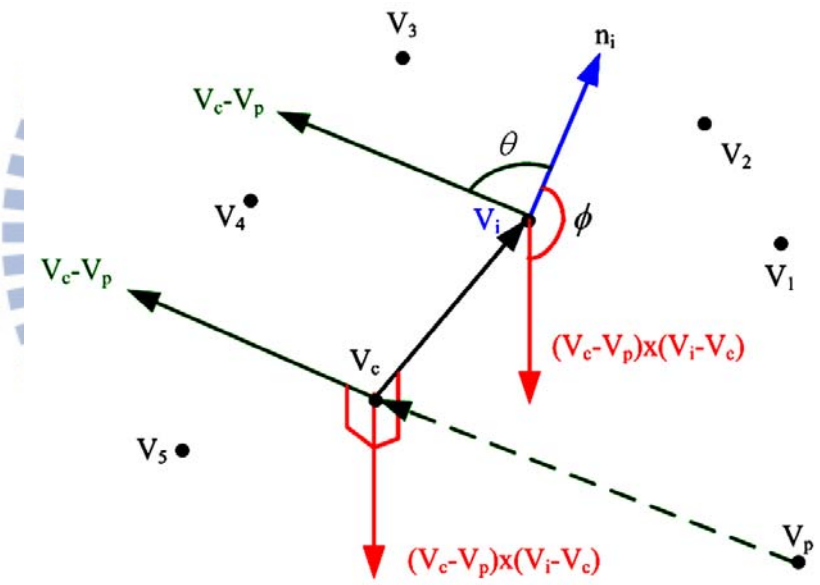


Figure 4-9: Diagram describes two viewpoint direction related angles θ and ϕ.

The new added angle ϕ can be computed by the following equation:

$$\phi = \cos^{-1}\left(\frac{((V_c - V_p)\times(V_i - V_c))\bullet n_i}{\| (V_c - V_p)\times(V_i - V_c)\| \cdot \| n_i \|}\right). \tag{4.8}$$

Then the *N*-bin orientation histograms (each bin cover 180/*N* degree) can be computed by accumulating the angle ϕ. Thus, MVFH is finished by dividing the total number of point clouds to normalize histogram in each bin. To demonstrate the improvement of the modified viewpoint feature histogram, we utilize the previous example presented by Fig. 4.8, which has

similar VFH in much different view, to re-computed MVFH. Figure 4.10 depicted the computed MVFH. As shown in this figure, the first histogram and the second histogram have different shape. This example clarifies that MVFH correct the error of much different view with similar VFH.

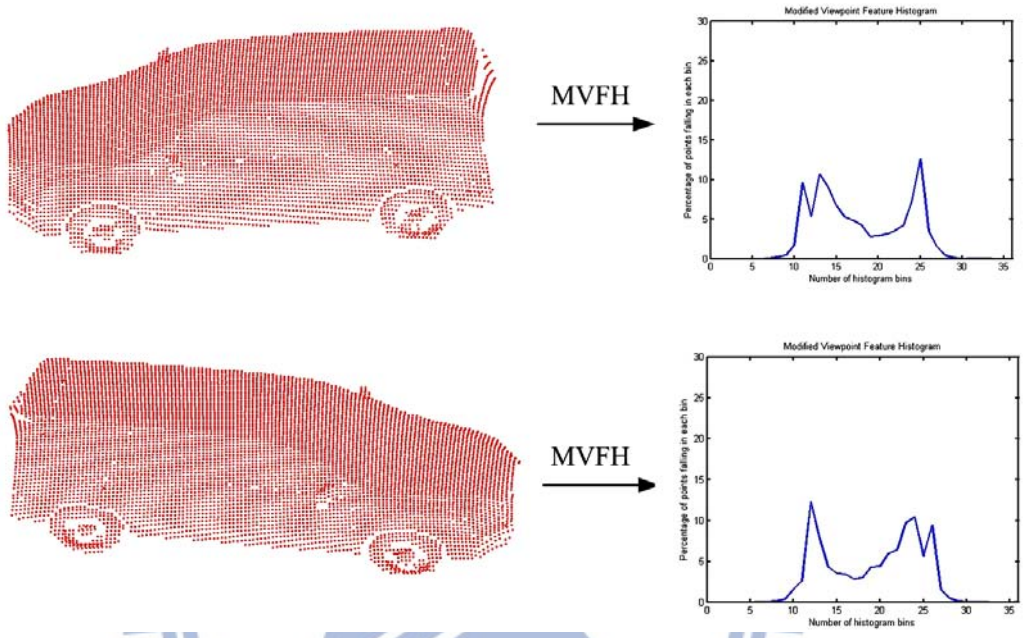


Figure 4-10: Example of modified viewpoint feature histograms in much different view.

(iii) TNFN Training

After extracting MVFH from a 3D object, let MVFH be the input neurons of TNFN and let the desired pose be the output neurons of TNFN. The desired pose comprises six degrees of freedom, including three rotation angles (ϕ, φ, θ) and three translation parameters (x, y, z). Thus, the use of TNFN is to model the relationship between the MVFH and the desired pose. Once receiving a MVFH from capturing a certain view of point clouds, the TNFN would output an estimated pose, which can be used to coarsely align the input point clouds with the reference model. To this end, training of a TNFN to provide the required pose would affect the alignment accuracy.

To perform training of a TNFN, the reference model is used as a basis for synthesizing a set of point clouds constituting a training-set. Each training point data is generated by

applying the transformation defined in Eq. (4.6). To reduce the correlations between training point clouds, the six parameters are selected randomly and independently within the predefined boundaries. After the training-set has been generated, the MVFH method is used to represent the training point clouds as input features of a TNFN. Subsequently, the proposed RGLS-HCCA would be adopted to begin training of a TNFN and the training procedure would stop as the stopping condition is satisfied. Although the training phase is lengthy, the executing phase of the proposed coarse alignment method merely consists of computing the MVFH descriptor and then feeding it into TNFN to estimate the corresponding pose.

(b) TNFN-based surface modeling

The purpose of the TNFN-based surface modeling is to provide an evaluation method for performing the fine alignment of 3D surface. The evaluation is to measure how close the distance from the reference surface to input point clouds is. Thus, the major part of the TNFN-based surface modeling is to use TNFN to model the 3D surface that maps the 3D Euclidean input space (input 3D point (x,y,z)) into 1D Euclidean output space (the shortest distance to the reference surface). Such mapping can be considered a cost function that evaluates the distance between the input point clouds and the reference model. Thus, the TNFN mapping can combine with the downhill simplex optimization method to iteratively compute the rotation matrix R and translation vector T to perform the fine alignment of 3D surface. The detail of the combination of the TNFN mapping and the downhill simplex optimization will be discussed in the executing phase.

The procedures of modeling the 3D surface involve combining the cube model, creation of training data, and surface modeling using TNFN. These operations are explained bellow.

(i) Combing cube model

To model the reference surface, uniform distributed point clouds are needed to prepare the training data. In this study, a cube model is generated to be combined with the reference model. The cube model encloses the reference surface, and the point clouds within the cube

are sampled uniformly. Thus, the point clouds around the reference model can serve as the training data for modeling the reference surface. Figure 4.11 depicts the locations of cube and reference model where the reference model is located at the center of the cube.

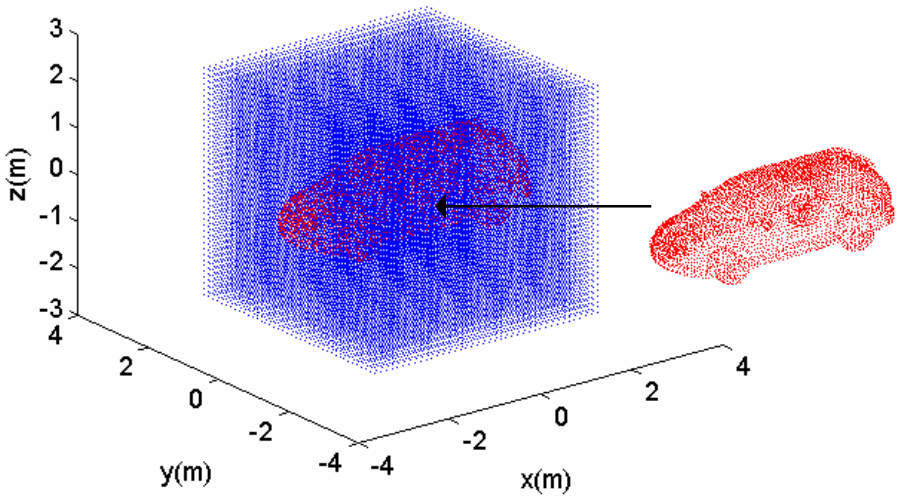


Figure 4-11: Location of cube and reference model.

(ii) Creation of training data

In the creation of training data, we extract the point clouds enclosed the cube satisfying the distance from a point (x,y,z) to the reference model less than a *predefined value*. The *predefined value* is set by observing the alignment error yielded from the coarse alignment case. Therefore, the point clouds (x,y,z) satisfies

$$Dist(x,y,z) \le predefined\ value \tag{4.9}$$

will be used for training the TNFN. In general, the *predefined value* must be set sufficient large to involve all the coarse alignment cases. Thus, to simply the creation of training data, this paper set all point clouds inside the cube model to be the training data.

(iii) Surface modeling using TNFN

Similar to the TNFN training in the coarse alignment learning case, the structure shown in Fig. 4.5 is used to model the 3D surface. The input of the TNFN is defined as the coordinates (x,y,z) of a point cloud, and the output of the TNFN is the unsigned shortest distance from a

point (x,y,z) to the reference model. Thus, the surface of the reference model can be modeled using the TNFN to map the 3D coordinate of point cloud data into the 1D distance between the cube data and the reference model. The representation of the modeling function can be written as follows:

$$Dist = f(x, y, z). \tag{4.10}$$

The total distance between the cube data and the reference model can be computed as follows:

$$TotDist = \sum_{i=1}^{N} f(x_i, y_i, z_i), \tag{4.11}$$

where N is the number of the cube model. Thus, when the resolution of the cube model is sufficiently high, any arbitrary point clouds inside the cube can be send into a trained TNFN to estimate the distance between the input point clouds and the reference model.

In consideration of training a TNFN to model the reference surface, as well as the coarse alignment learning, RGLS-HCCA is also utilized to perform training the TNFN.

4.2.2 Execution Phase

In the execution phase, the input point clouds are aligned with the reference model by means of MVFH extraction, TNFN-based coarse alignment, and TNFN-based fine alignment. MVFH extraction has been discussed in Section 4.2.1 (Part (a)), whereas the TNFN-based coarse and fine alignments are described bellow.

(a) TNFN-based coarse alignment

Assuming the MVFH descriptor has been calculated, the descriptor is forwarded to the trained TNFN to obtain the rotation angles (ϕ, φ, θ) and translation parameters (x, y, z). Then, the six parameters are used to compute the rotation matrix R and translation vector T defined in Eq. (4.6). Based on R and T, we obtain the estimated pose to coarsely align the input point clouds with the reference model.

(b) TNFN-based fine alignment

The procedure of the TNFN-based fine alignment consists of the TNFN mapping and the downhill simplex optimization [96]. In the TNFN mapping, the TNFN maps each 3D point cloud (x, y, z) into a 1D distance function $f(x, y, z)$ (defined in Eq.(4.16)). The total distance function $\sum f(x, y, z)$ is computed by summing of each distance mapping of 3D point cloud. Thus, the total distance function is used as the cost function of the subsequent downhill simplex optimization. In downhill simplex optimization, iterative calculation of rigid transformation between input point clouds and reference model is adopted to minimize the cost function. Each iterative loop uses the downhill simplex method to compute the rotation matrix R and translation vector T to perform fine alignment. Once the downhill simplex optimization is completed, the final R and T are used to calculate the estimate pose that align the input point clouds with reference surface.

Detail steps of the downhill simplex optimization [97] for fine alignment of 3D surface are described as follows:

Step 0: Under 3D rigid body transformation, we choose six degrees of freedom (three rotation angles (ϕ, φ, θ) and three translation parameters (x, y, z)) as the vertex of simplex. Then we randomly generate 6+1 initial vertices of simplex within a fixed range where 6 represents the dimension of vertex vector. In this study, the 7 initial vertices are denoted as $X_0, X_1, \cdots, X_6$.

Step 1: Two procedures are performed in this step.

(1) Evaluation: Based on each vertex of simplex, we can compute the corresponding rigid transformation matrix defined in Eq. (4.6). According to the transformation matrix, the input point clouds yielded by coarse alignment are mapped into new coordinates. The new point clouds are forwarded into the trained TNFN to get distance function $f(x, y, z)$. Then, $\sum f(x, y, z)$ can be calculated by sum of all mapping of new point clouds.

(2) Sorting: Here we choose total distance function $\sum f(x, y, z)$ as the cost function and re-define the symbol to be $C(X_i)$ where X_i indicates the i-th vertex of simplex. Then

we sort the $C(X_i)$ and set the order as follows:

$$C(X_0) < C(X_1) < \cdots < C(X_6). \tag{4.12}$$

Step 2: In this step, the reflection point X_6^R is calculated. The downhill simplex optimization utilizes the reflection point as the first candidate point to replace the worst point X_6. The reflection point is calculated as follows:

(a) First find centroid of the remaining point $(X_0 \sim X_5)$:

$$M = \frac{1}{6}\sum_{i=1}^{5} X_i. \tag{4.13}$$

(b) Then seek the reflection point:

$$X_6^R = M + \alpha(M - X_6), \tag{4.14}$$

where $\alpha > 0$ and the default value is $\alpha = 1$.

(c) Finally, $C(X_6^R)$ can be calculated by the means of evaluation method described in Step 1.

Step 3: There are 3 cases are discussed in this step.

Case 1: If $C(X_6^R) \geq C(X_0)$ and $C(X_6^R) < C(X_5)$, choose X_6^R to replace X_6. Then we re-sort the simplex and forward to Step 4.

Case 2: If $C(X_6^R) < C(X_0)$, compute the expansion point X_6^E as follows:

$$X_6^E = X_6^R + \gamma(X_6^R - M), \tag{4.15}$$

where $r > 0$ and the default value is $r = 1$. Then calculate the $C(X_6^E)$. If $C(X_6^E) < C(X_0)$, choose X_6^E to replace X_6. Otherwise, choose X_6^R to replace X_6. After that, we re-sort the simplex and forward to Step 4.

Case 3: If $C(X_6^R) \geq C(X_0)$ and $C(X_6^R) \geq C(X_5)$, compute the contraction point X_6^C as follows:

$$X_6^C = M + \beta(\overline{X}_6 - M), \tag{4.16}$$

where $0 < \beta < 1$ and the default value is $\beta = 0.5$. If $C(X_6^R) < C(X_6)$, then $\overline{X}_6 = X_6^R$. Otherwise, if $C(X_6^R) \geq C(X_6)$, then $\overline{X}_6 = X_6$. Subsequent, check the case of $C(X_6^R)$ as follows:

(i) If $C(X_6^C) < C(X_6)$, choose X_6^C to replace X_6. Then, we re-sort the simplex and forward to Step 4.

(ii) If $C(X_6^C) \geq C(X_6)$, shrink the whole simplex toward X_0. After shrinking, the new simplex is expressed as:

$$\left[X_0, (\eta X_0 + (1-\eta)X_1), \cdots, (\eta X_0 + (1-\eta)X_5), (\eta X_0 + (1-\eta)\overline{X}_6)\right], \tag{4.17}$$

where $0 < \eta < 1$ and the default value is $\eta = 0.5$.

Step 4: If the least cost function meet one of the following conditions, the downhill simplex method is terminated, and output the final results.

(a) The number of loops reaches a predefined maximal iteration value.

(b) The value of cost function is less than a minimal threshold.

Otherwise, if the least cost function does not meet the above conditions, then we feedback to the Step 2 to continue the optimization procedure.

To sum up, the final results of the downhill simplex method would output the best vertex of simplex. Then we decode it to the six degrees of freedom ($\phi, \varphi, \theta, x, y, z$). These parameters can be transfer to a rotation matrix *R* and translation vector *T*. The fine alignment of 3D surface is completed by computing the rigid body transformation according the *R* and *T*.

Chapter 5
Experimental Results

In this chapter, the performance of RGLS-HCCA is demonstrated on three problems. The first one is a problem of prediction of Mackey-Glass time series. This problem is a common benchmark for examining different learning algorithms. By applying RGLS-HCCA to the benchmark, RGLS-HCCA would show how fast the algorithm converges and lower estimating error comparing with other learning algorithms. Subsequently, two real world problems, which are 2D and 3D image alignment tasks, are used to verify the applications of RGLS- HCCA. The proposed RGLS-HCCA would act from a simulator to a real system. The experiments would evaluate the proposed method of aligning 2D and 3D images in comparison with other typical alignment systems.

This chapter is divided into three subsections. In Section 5.1, the prediction of Mackey-Glass time series is used to examine the learning performance of RGLS-HCCA. In Section 5.2 and 5.3, RGLS-HCCA is applied to 2D and 3D image alignment problems, respectively.

All experiments in this chapter are performed by using an Intel Core i7 860 chip with a 2.8GHz CPU, a 3G memory, and the Matlab 7.5 simulation software.

5.1 Prediction of Mackey-Glass Time Series

To verify the proposed RGLS-HCCA, Mackey-Glass time series is utilized to compare RGLS-HCCA with that of other methods. The initial parameters of the proposed RGLS-HCCA are determined by parameter exploration methods ([98] and [99]). As shown in [98], a small population size is good for the initial performance, a large population size is good for long-term performance and a low mutation rate is good for on-line performance, a high mutation rate is good for off-line performance. Moreover, in [99], parameters for genetic

algorithms can be adjusted by exploring the predefined range in increments of a small value. For instance, the population size has the range from 10 to 100 in increments of 10. Thus, this study adjusts parameters of RGLS-HCCA according to the criteria mentioned in parameter exploration methods. The results of parameters used in this study are listed in Table 5.1 where "none" in SLE indicates "not used" in the learning phase.

Moreover, since A^TA (with size of 50 × 50 under conditions of 10 fuzzy rules and four input in a TNFN) in Eq. (3.5) is singular (rank of A^TA is about 47) for the example of Mackey Glass time series prediction, this dissertation incorporates RGLS to make (A^TA+λI) is non-singular. To consider the RGLS parameter (λ), this paper adopts the cross-validation method [100] to adjust it. The notion of the cross-validation method is to divide the training data set into training data and validation data and increase λ with small increments to balance the error of training data set and validation set. Thus, this paper uses cross-validation method to optimize the RGLS parameter (λ) and final adjusted λ of this example is listed in Table 5.1.

Table 5.1: Initial parameters of RGLS-HCCA before training.

Parameters	Value	
	PLE	SLE
P_{size}	30	20
N_c	20	none
Selection_Times	40	none
NormalTimes	10	none
ExploreTimes	15	none
Crossover Rate	0.6	0.6
Mutation Rate	0.2	0.3
[M_{min}, M_{max}]	[6, 15]	[6, 15]
[m_{min}, m_{max}]	[-5, 5]	[-5, 5]
[σ_{min}, σ_{max}]	[3, 20]	[3, 20]
Minimum_Support	TransactionNum/2	none
Minimum_Confidence	60%	none
RGLS parameter (λ)	0.00001	0. 00001

The Mackey-Glass time series is a common benchmark for examining different learning algorithms or fuzzy modeling research communities. In earlier work [101], Lapedes and Farber used a back propagation network to predict Mackey-Glass time series. After that, other

researches [102] followed Lapedes and Farber's work to be a benchmark to examine algorithms. Thus, we utilize such Mackey-Glass time series to perform an analysis on our proposed algorithm and other evolutionary algorithms.

The Mackey-Glass time series is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{5.1}$$

For this time series prediction problem, Jang [103] extracted 1000 input-output data pairs $\{x, y^d\}$ from t=118 to t=1117, which consisted of four past values of $x(t)$, that is

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)], \tag{5.2}$$

where τ=17 and $x(0)$=1.2 and $x(t)$=0 for t<0. The reason choosing four past values to predict time series is from Jang's [103] work which wanted to allow comparison with other researches' algorithms (Lapedes and Farber [101], Moody [104], Crower [102]). Thus, there are four input to RGLS-HCCA, corresponding to these values of $x(t)$, and one output representing the value $x(t+\Delta t)$, where Δt is a time prediction into the future. The first 500 pairs [from x(118) to x(617)] are the training data set, and the remaining 500 pairs [from x(618) to x(1117)] are the testing data set used for validating the proposed method. The values are floating-point numbers assigned using the RGLS-HCCA initially. The fitness function in this case is defined in Eq. (3.26) and (3.27) to train the neural fuzzy network. The evolution learning processes 500 generations and it is repeated 50 times. For comparative analysis, the present study adopts the root mean square error (RMSE), which is defined as follows:

$$RMSE = \left[\frac{1}{N_t} \sum_{l=1}^{N_t} (Y_l(t+6) - Y_l^d(t+6))^2 \right]^{1/2}, \tag{5.3}$$

where N_t is the number of testing data, $Y_l^d(t+6)=x(t+6)$ is the desired value, and $Y_l(t+6)$ is the predicted value by the model with four inputs and one output.

In this example, RGLS-HCCA is compared the performance with the HESP [23], ESP

[14], and SANE [13]. In these models, the learning parameters, which are determined according the parameter exploration method [98] and [99], are shown in Table 5.2. To perform training, the evolution learning processes for 500 generations. Figure 5.1(a)-(d) show the prediction results of the three models. The symbol "o" represents the desired output of the time series, and the symbol "*" represents the output of the four models. Figures 5.2(a)-(d) illustrate the error between the desired and four models' outputs. As shown in Fig. 5.1-2, the performances of the RGLS-HCCA are better than those of others. Fig. 5.3 shows the learning curves of the four models. As shown this figure, the proposed RGLS-HCCA model converges faster than those of other three models.

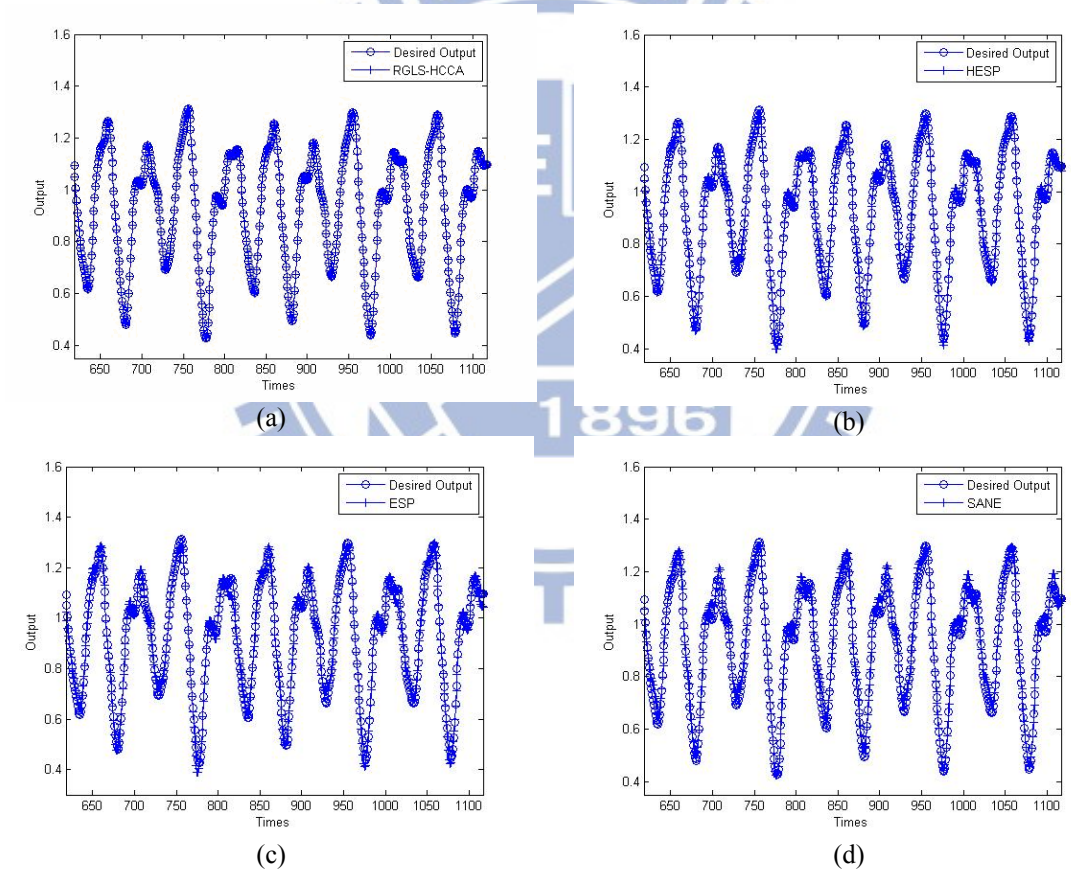


Figure 5-1: Prediction results of the (a) proposed RGLS-HCCA, (b) HESP, (c) ESP, and (d) SANE.

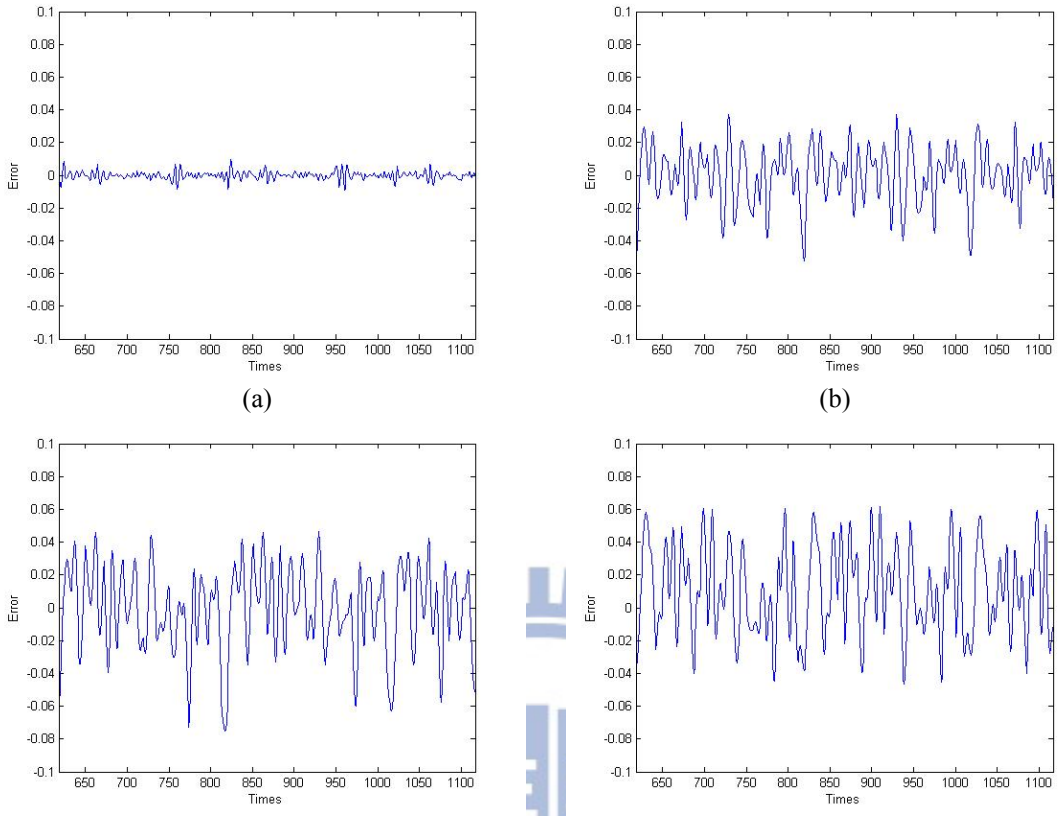


Figure 5-2: Prediction errors of the (a) proposed RGLS-HCCA, (b) HESP, (c) ESP, and (d) SANE.

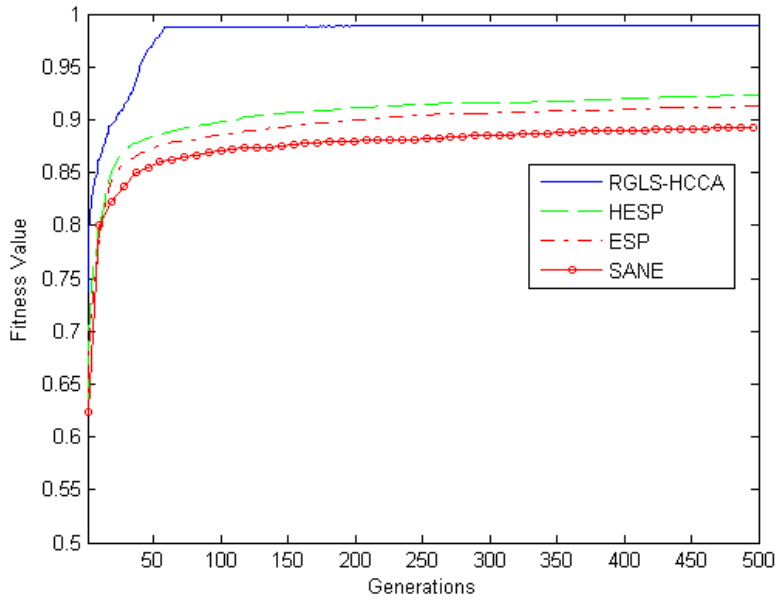


Figure 5-3: Learning curves of the proposed RGLS-HCCA, HESP, ESP, and SANE.

In addition HESP, ESP, and SANE, to further show the effectiveness and efficiency of the proposed RGLS-HCCA model, we also apply MGCSE [15], and traditional genetic algorithm (TGA) [16] to the same problem. To compare with theses algorithms, according the

parameter exploration method [98] and [99], 14, 13, 12, 14, and 12 fuzzy rules are set for HESP, MGCSE, ESP, SANE and TGA, respectively. In addition, the population size has the range of 10 to 250 in increments of 10, the crossover rate has the range of 0.1 to 1 in increments of 0.1, and the mutation rate has the range of 0 to 0.4 in increments of 0.01. To this end, the parameters used for HESP, MGCSE, SANE and TGA are listed in Table 5.2. In addition, as same with RGLS-HCCA, the evolution learning of each method processes for 500 generations and is repeated 50 times. Table 5.3 lists the generalization capabilities of the proposed RGLS-HCCA, HESP, MGCSE, ESP, SANE, and TGA. Clearly, as shown in Table 5.3, RGLS-HCCA obtains a lower RMSE than other methods. In TGA, according to [13], cooperative coevolutionary algorithms can find solutions faster and solve harder problems than TGA. Thus, RGLS-HCCA and other methods (HESP, MGCSE, ESP, and SANE) exhibit lower RMSE than TGA. In SANE, symbiotic evolution is adopted. Since symbiotic evolution only used one population to evaluate every partial solution, the evaluation would cause partial solutions too similar. Instead, the proposed RGLS-HCCA provides several groups to evaluate each partial solution. Thus, the proposed model has more chance to obtain optimal solution. The explanation can specify that the proposed method has better performance than SANE. To consider group-based evolutionary algorithms (HESP, MGCSE, and ESP), when faced with complex problems, the dimension of chromosomes is still high such that low convergence rate occurs. Thus, this dissertation incorporates RGLS to reduce the dimension of chromosomes and proposes HCCA to self adjust the parameters and structure of TNFN. Based on this fact, the proposed model would be superior to HESP, MGCSE, and ESP.

Moreover, to compare RGLS with the pseudo inverse method, this paper also performs the same experiment on the pseudo inverse method. The average RMSE of the pseudo inverse method for 50 runs is 0.0025, which is slightly larger than RGLS (0.0023). Thus, in this example, RGLS would be better than the pseudo inverse method.

Table 5.2: Initial parameters of four learning models.

Method	Parameters			
	Populations size	Crossover rate	Mutation rate	Fuzzy rules
HESP	14(30 for subpopulations size)	0.7	0.03	14
MGCSE	13(30 for subpopulations size)	0.6	0.04	13
ESP	12(30 for subpopulations size)	0.7	0.05	12
SANE	120	0.1	0.15	14
TGA	140	0.8	0.01	12

Table 5.3: Performance comparison of various existing models.

Method	RMSE			
	Best	Mean	Worst	STD
RGLS-HCCA	**0.0017**	**0.0023**	**0.0026**	**0.0005**
HESP	0.0118	0.0149	0.0193	0.0017
MGCSE	0.0100	0.0158	0.0190	0.0019
ESP	0.0110	0.0172	0.0219	0.0026
SANE	0.0145	0.0219	0.0313	0.0039
TGA	0.0192	0.0271	0.0747	0.0079

Furthermore, this example also compares the running time of RGLS-HCCA with that of other methods. The running time defined in this case is used to measure the time when the fitness of the algorithm exceeds the predefined value (0.85). The results of four algorithms over 50 runs are reported in Table 5.4. As shown in this table, the proposed RGLS-HCCA is faster than HESP, MGCSE, ESP, SANE, and TGA.

Table 5.4: Comparison of the running time of various algorithms.

Method	Best(seconds)	Worst(seconds)	Mean(seconds)
RGLS-HCCA	**6.07**	**43.02**	**23.28**
HESP	16.46	121.49	32.61
MGCSE	11.85	162.27	38.27
ESP	18.54	177.83	40.16
SANE	16.99	231.18	54.80
TGA	17.52	180.27	103.36

5.2 Results of 2D Image Alignment

In the 2D image alignment experiment, visual inspection images, which are 640 by 480

pixels size, are used to examine the utility of the proposed CNFN-based image alignment method. Figure 5.4 depicts an example about such images where the left side is a reference image and the right side is a transformed image by a scaling, rotation and translation. Also in this figure, the dashed window represents a template window (the size is 200×200, and feature vectors are extracted within this window), and the cross sign denotes the reference location of the template.

(a) (b)

Figure 5-4: (a) Reference image. (b) Testing image with scale=0.9, rotation=-10°, vertical translation=5, horizontal translation=10.

In the following 2D image alignment experiments, two kinds of neural works are performed. The first one is a one-stage of CNFN (OS-CNFN), which is taken into consideration of applying to the medium range of affine parameters and examining different learning methods. The second one is a multi-stage of CNFN (MS-CNFN), which is used to apply the trained networks to adapt to a large range of affine parameters.

5.2.1 Alignment Results of One-stage Neural Fuzzy Network

In Table 5.5, four types of experimental images are prepared for simulation. The first three types of images are the synthesized ones generated randomly within the range in Table 5.6. In the last type of images are real ones captured from a camera. Moreover, Table 5.6 indicates the searching range for image alignment. If the affine transformation exceeds the range, the image alignment system may not promise high accuracy. Thus, the range of the image alignment defined in this subsection is restricted in Table 5.6.

Table 5.5: Experimental images preparation.

Image Type	Image Preparation
Synthesized Images	800 images are generated with randomly selected affine parameters within the predefined range.
Training Images	The 50% of synthesized images
Testing Images	The 50% of synthesized images
Real Images	Images are acquired from CCD camera with different pose from the reference image.

Table 5.6: Range of affine transformation parameters used in experiments.

Affine transformation parameter	The range of affine transformation parameter
Scale	[0.7 1.3]
rotation(degrees)	[-30 30]
vertical translation(pixels)	[-20 20]
horizontal translation(pixels)	[-20 20]

The following parts will discuss the comparison with existing learning methods and with existing image alignment systems.

Part 1: Comparison with existing learning methods

Three typical evolutionary learning methods, which are HESP [23], ESP [14] and SANE [13], are implemented carefully (the learning parameters are found using the method given in [98] and [99]) to compare with the proposed RGLS-HCCA. Moreover, to explore the number of fuzzy rules for HESP, ESP and SANE, the fuzzy rules are tuned by setting the range of 20-100 in increments of 5. Thus, the results find that 85, 80 and 80 rules are suitable for SANE, ESP, and HESP respectively.

In this experiment, 800 synthesized images are generated randomly by the way in Table 5.5 where 50% of images are for training set and another 50% ones are for testing set. Then 33-element feature vectors are obtained by applying Gabor-WGOH with PCA dimensionality reduction to above-generated images. Moreover, before training, the initial parameters of RGLS-HCCA are given in Table 5.7. The initial parameters are tuned by the parameter exploration method (where the RGLS parameter (λ) is adjusted by cross-validation method) which has been described in section 5.1.

To consider SRM in RGLS-HCCA, Figure 5.5 shows the best results of the probability vectors for 15 runs in different training and testing images. As shown in Fig. 5.5, the highest probability means the most suitable number of fuzzy rules of the TNFN model in the best run. Therefore, the suitable number of fuzzy rules is 24. It represents that in most cases a 24-rule TNFN would have higher probability to obtain better performance than other rules within $[M_{\min}, M_{\max}] = [18, 25]$.

Figure 5.6 depicts the learning curves of four models. From this figure, RGLS-HCCA demonstrates faster convergence speed than those of HESP, ESP and SANE. Moreover, to examine the learning accuracy, the testing data would be sent into the trained TNFNs to get the estimated pose including scale, rotation, vertical translation, and horizontal translation. Then, by comparing the desired pose, four alignment errors (i.e. ErrScale, ErrAngle, ErrDx, and ErrDy) are generated. Table 5.8 presents the learning accuracy of four evolutionary models. From this table, the proposed RGLS-HCCA exhibits the lowest errors among four models. To this end, the proposed model not only promotes its leaning speed but also sustains the high learning accuracy.

Table 5.7: Initial parameters before training.

Parameters	Value	
	PLE	SLE
P_{size}	40	20
N_c	20	none
Selection_Times	50	None
NormalTimes	10	None
ExploreTimes	15	None
Crossover Rate	0.6	0.7
Mutation Rate	0.2	0.4
$[M_{\min}, M_{\max}]$	[18, 25]	[18, 25]
$[m_{\min}, m_{\max}]$	[-10, 10]	[-10, 10]
$[\sigma_{min}, \sigma_{max}]$	[3, 15]	[3, 15]
Minimum_Support	TransactionNum/2	none
Minimum_Confidence	60%	none
RGLS parameter (λ)	0.003	0.003

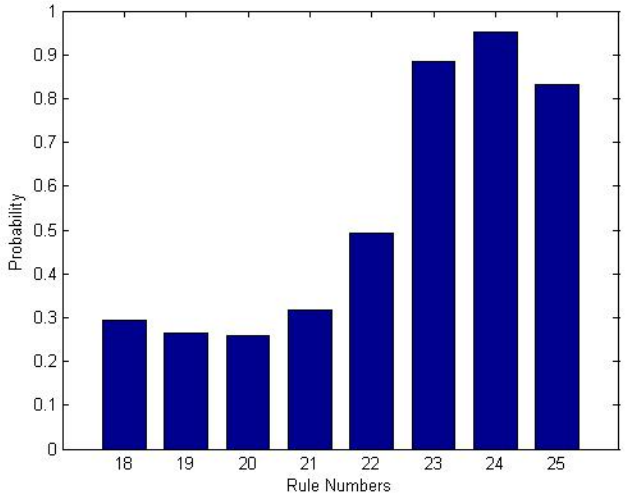


Figure 5-5: Best results of the probability vectors for 15 runs in SRM.

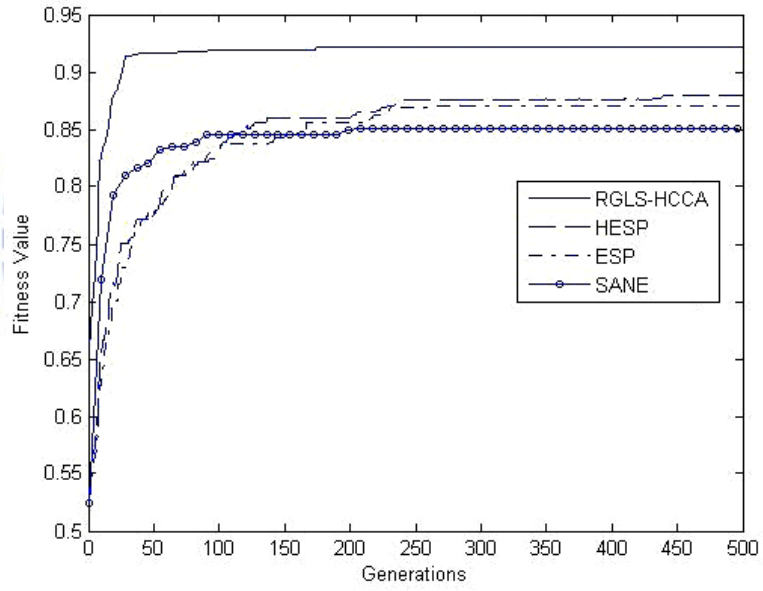


Figure 5-6: Learning curves of the RGLS-HCCA, HESP, ESP, and SANE methods.

Table 5.8: Leaning accuracy of the RGLS-HCCA, HESP, ESP, and SANE methods.

Method	Mean Errors			
	ErrScale	ErrAngle (degrees)	ErrDx (pixels)	ErrDy (pixels)
RGLS-HCCA	**0.0066**	**0.3252**	**0.4953**	**0.5058**
HESP	0.0223	1.4431	1.1309	1.1600
ESP	0.0229	2.0470	1.1051	1.6137
SANE	0.0247	2.0311	1.4620	1.8132

Part 2: Comparison with existing image alignment systems

To evaluate OS-CNFN (i.e. the proposed system) in comparison with other existing systems ([42], [44], [87], [88], and [91]), the implementation of these existing systems are carefully cited their original paper. The comparison in this section consists of the alignment accuracy, alignment speed, robustness and real image alignment case. These comparisons are discussed in the following parts.

A. Alignment accuracy

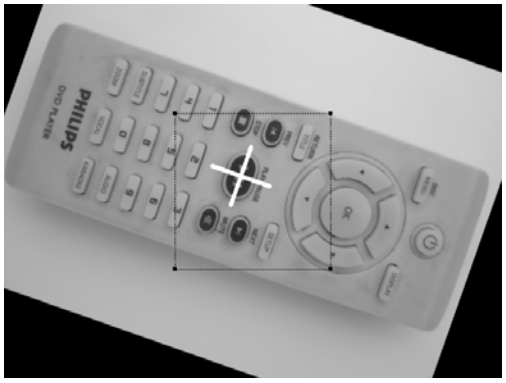
To compare the alignment accuracy of different systems, the training images, which are used to train neural networks, and the testing images, which are used to check the alignment accuracy, are generated by the way described in Table 5.5.

Figure 5.7 depicts an alignment example for a testing image on six different systems. The cross sign in this figure denotes the estimated results. From this figure, OS-CNFN can estimate more accurate position and orientation of the cross sign than other systems.

In addition, 15 runs using different training and testing images are performed to further examine the alignment accuracy of the proposed system. The simulation results are shown in Table 5.9, which presents the average and standard deviation error of six image alignment systems. From this table, OS-CNFN exhibits the lowest alignment error than other systems. Moreover, the simulated data indicates that the alignment reaches the high accuracy level; thus, OS-CNFN can provide a useful way to align images very accurately.

Table 5.9: Alignment errors in different image alignment systems.

Method	Errors							
	ErrScale		ErrAngle (degrees)		ErrDx (pixels)		ErrDy (pixels)	
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
OS-CNFN	**0.0061**	**0.0065**	**0.3184**	**0.3106**	**0.4820**	**0.3985**	**0.5175**	**0.4260**
DCT [44]	0.0067	0.0098	0.6330	1.0681	1.4490	1.5832	0.9576	1.1290
FFT [87]	0.0121	0.0149	0.8020	1.0177	5.4070	4.9574	2.7508	2.4640
KICA [88]	0.0176	0.0192	1.4147	1.6462	0.9929	0.9172	1.2090	1.1842
ISOMAP[42]	0.0294	0.0268	2.0809	2.0043	1.6430	1.6123	2.2356	2.7793
SIFT[91]	0.0387	0.0775	0.4312	0.8516	1.0764	1.5838	2.1186	3.5750



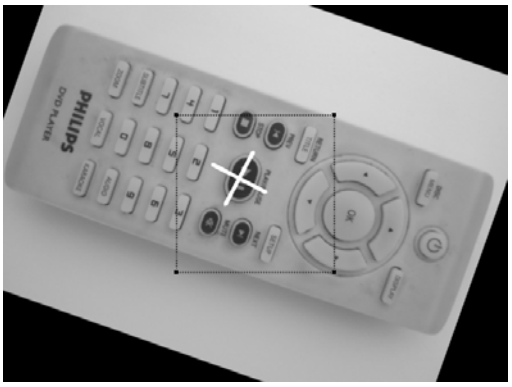
(a)

(b)

(c)

(d)

(e)



(f)

(g)

Figure 5-7: Alignment results for different systems: (a) Ground Truth, (b) OS-CNFN, (c) DCT, (d) FFT, (e) KICA, (f) ISOMAP, and (g) SIFT.

B. Alignment speed

To demonstrate the alignment speed, the execution time required in performing one image alignment task is discussed. In this paper, the steps of performing one image alignment task consists of capturing the template window from the input image, computing the feature within the window, and feeding the calculated feature into the trained network to get the affine parameters.

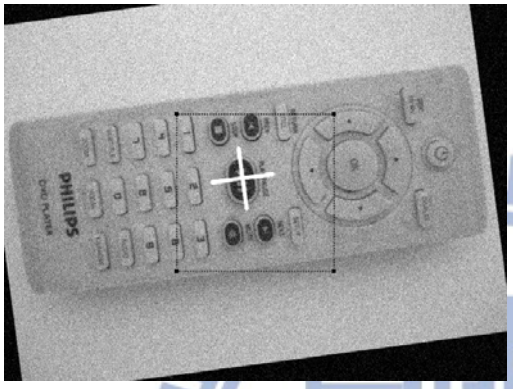
In this experiment, we utilize 400 testing images to perform image alignment tasks. The average execution time of OS-CNFN, DCT, FFT, KICA, ISOMAP, and SIFT take about 30ms, 26ms, 28ms, 65ms, 330ms, and 57ms respectively. From this result, it is obviously that OS-CNFN is almost as fast as the FFT and DCT systems and is more efficiently than other three systems.

C. Alignment Robustness

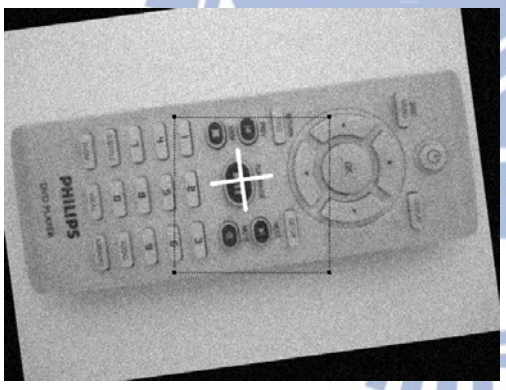
Next, the robustness of OS-CNFN under different levels of random additive Gaussian noise is discussed. In this experiment, 400 testing images are randomly generated with the addition of various strengths of Gaussian noise to examine the robust performance of different image alignment systems. Figure 5.8 illustrates an example of aligning a testing image with the reference image under 10 dB signal-to-noise ratio (SNR) condition. As shown in this figure, OS-CNFN estimates the rotation and translation of the cross sign more accurately than other methods.

The simulation results of the absolute estimating errors of affine parameters under eight levels of SNR is presented in Figure 5.9(a)-(d). From these figures, OS-CNFN demonstrates lower affine parameters error than other systems, especially as SNR is larger than 15 dB. It stands for OS-CNFN with high robustness against noise.

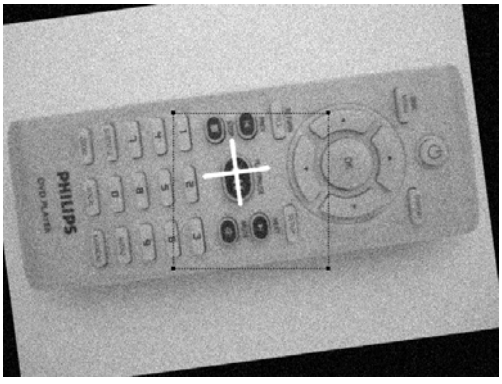
(a)

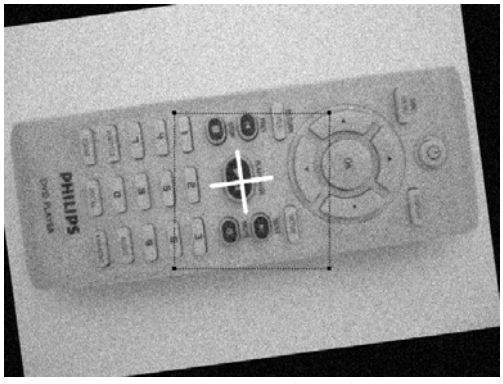


(b) (c)



(d) (e)





(f) (g)

Figure 5-8: Alignment results for different systems under 10 dB SNR condition: (a) Ground Truth, (b) OS-CNFN, (c) DCT, (d) FFT, (e) KICA, (f) ISOMAP, and (g) SIFT.

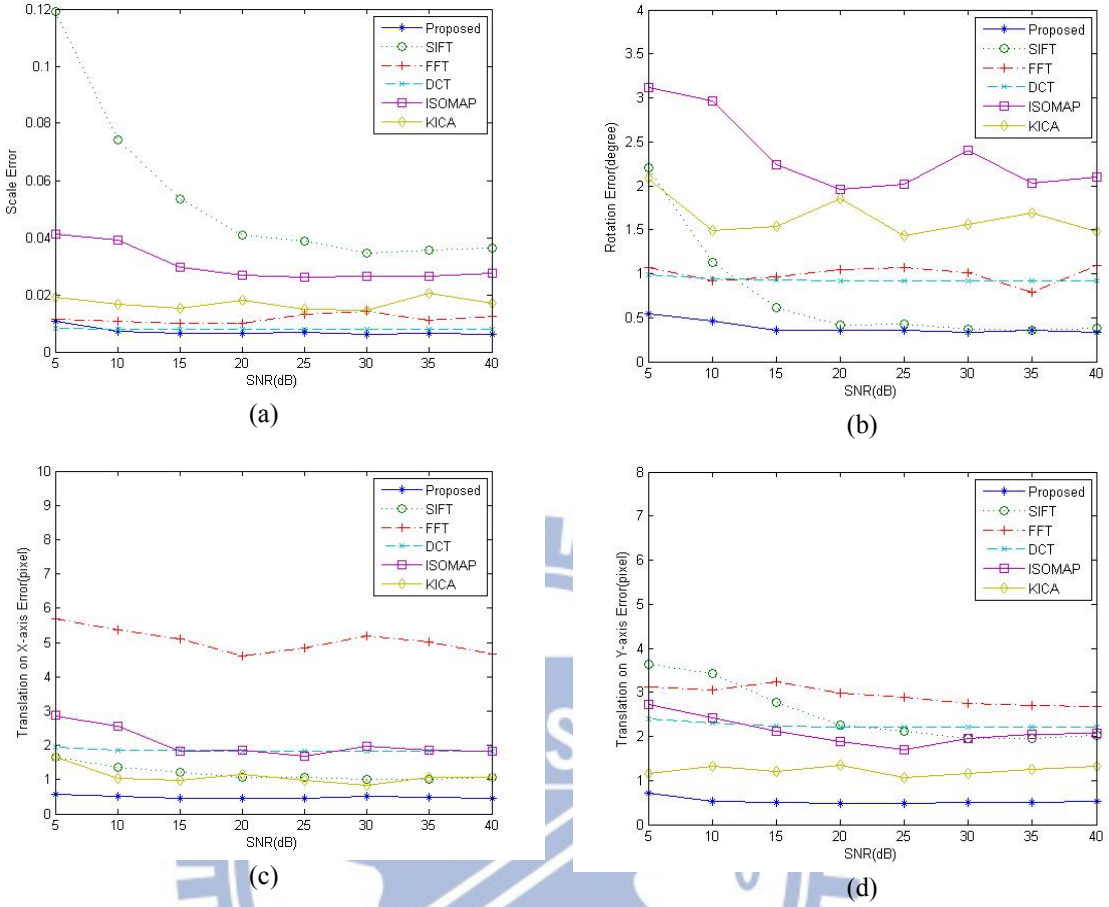


Figure 5-9: Average affine transformation errors comparison using OS-CNFN, DCT, FFT, KICA, ISOMAP, and SIFT under various SNR. Error with respect to (a) scale, (b) rotation, (c) translation on X-axis, and (d) translation on Y-axis.

D. Real Image Alignment Case

In this part, real images are utilized to verify the effectiveness of the proposed system. Figure 5.10 (a)-(d) presents the results of aligning the same real image using OS-CNFN, DCT, FFT, KICA, ISOMAP, and SIFT respectively. As shown in this figure, OS-CNFN demonstrates more accurate rotation and position of the cross sign than other alignment systems. Thus applying the proposed image alignment system to real image cases is feasible.

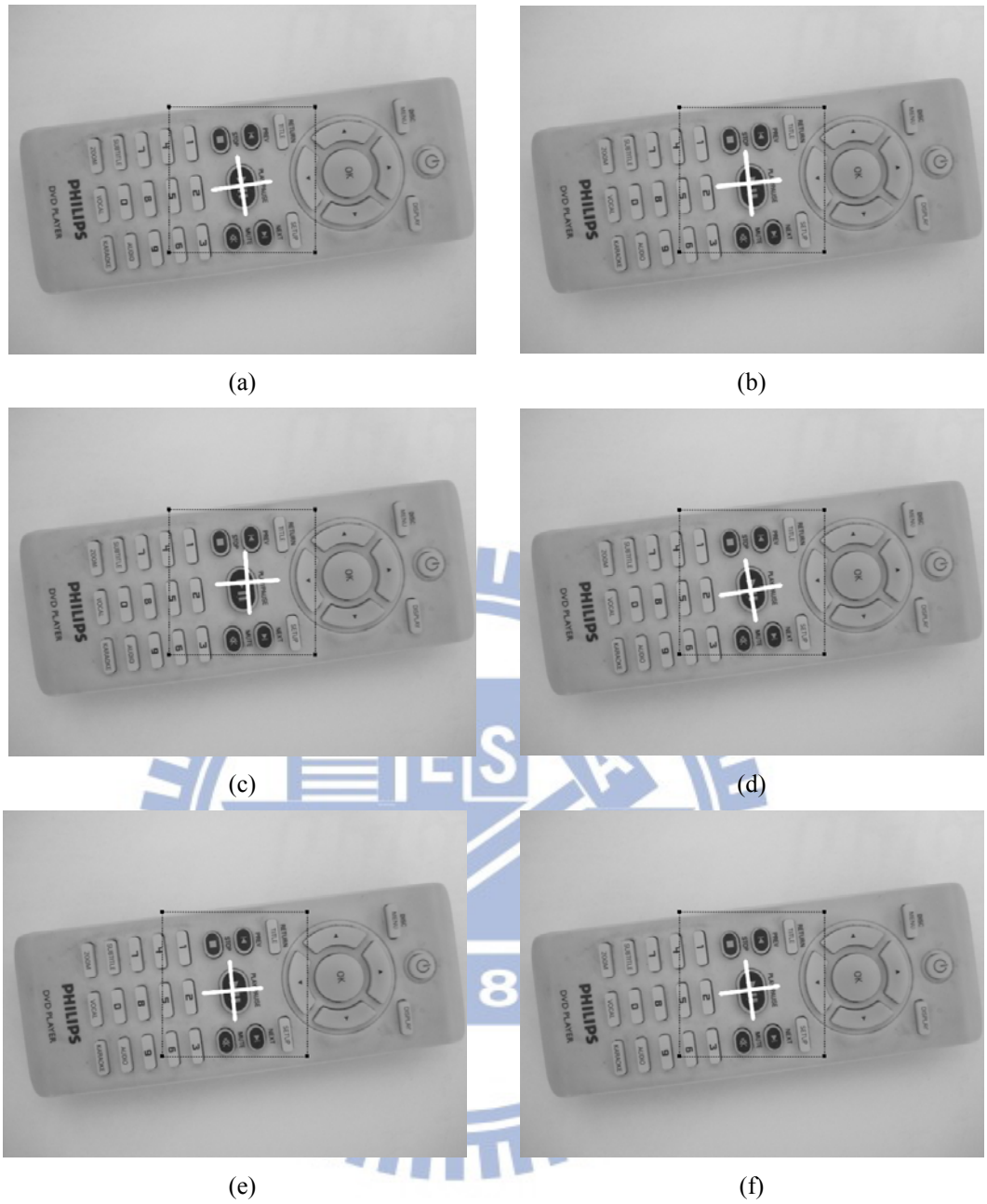


Figure 5-10: Results of image alignment on real images: (a) OS-CNFN, (b) DCT, (c) FFT, (d) KICA, (e) ISOMAP, (f) SIFT.

5.2.2 Alignment Results of Multi-stage Neural Fuzzy Networks

Table 5.10 defines the target alignment range for aligning the visual inspection images. All image alignment systems mentioned in this subsection are implemented to reach the target alignment range.

The experimental results of multi-stage neural networks contain two parts. In part 1, the CNFN with RGLS-HCCA training is performed. In part 2, synthesized and real images are

used to compare the proposed image alignment system with other systems.

Table 5.10: Target alignment range.

Affine parameter	The range of affine parameter
Scale	[0.7 1.3]
rotation(degrees)	[-100 100]
vertical translation(pixels)	[-100 100]
horizontal translation(pixels)	[-100 100]

(1) Cooperative Neural Fuzzy Network with the RGLS-HCCA training

To achieve the target alignment range defined in Table 5.10, we choose three ranges of affine parameters described in Table 5.11 to accomplish the three-stage of CNFN (i.e. MS-CNFN). In this table, each range contains a single neural fuzzy network, and these ranges cooperate to adapt to a target alignment range. For the supply suitable training data for networks, this paper uses the self-organized training data-yielding method to generate 1165, 137, and 219 training data for coarse, medium, and fine alignment ranges, respectively. The map of recursive loop versus increased training data for each range defined in Table 5.11 is shown in Fig. 5.11. Based on this figure, the number of the increased training data decreases gradually and then self-organizes.

Prior to performing the training, the initial parameters of RGLS-HCCA are given in Table 5.12. Based on the training feature vectors and initial parameters, we perform the coarse, medium, and fine RGLS-HCCA training individually. These three-stage training stops when the fitness is greater than the predefined value. Therefore, once the training process has been performed, our image alignment system can be concluded to reach the target range defined in Table 5.10.

Table 5.11: Affine parameters range of three-stage CNFNs.

Affine parameter	The coarse range of affine parameter	The medium range of affine parameter	The fine range of affine parameter
Scale	[0.7 1.3]	[0.85 1.15]	[0.9 1.1]
rotation(degrees)	[-100 100]	[-50 50]	[-5 5]
vertical translation(pixels)	[-100 100]	[-30 60]	[-5 5]
horizontal translation(pixels)	[-100 100]	[-30 60]	[-5 5]

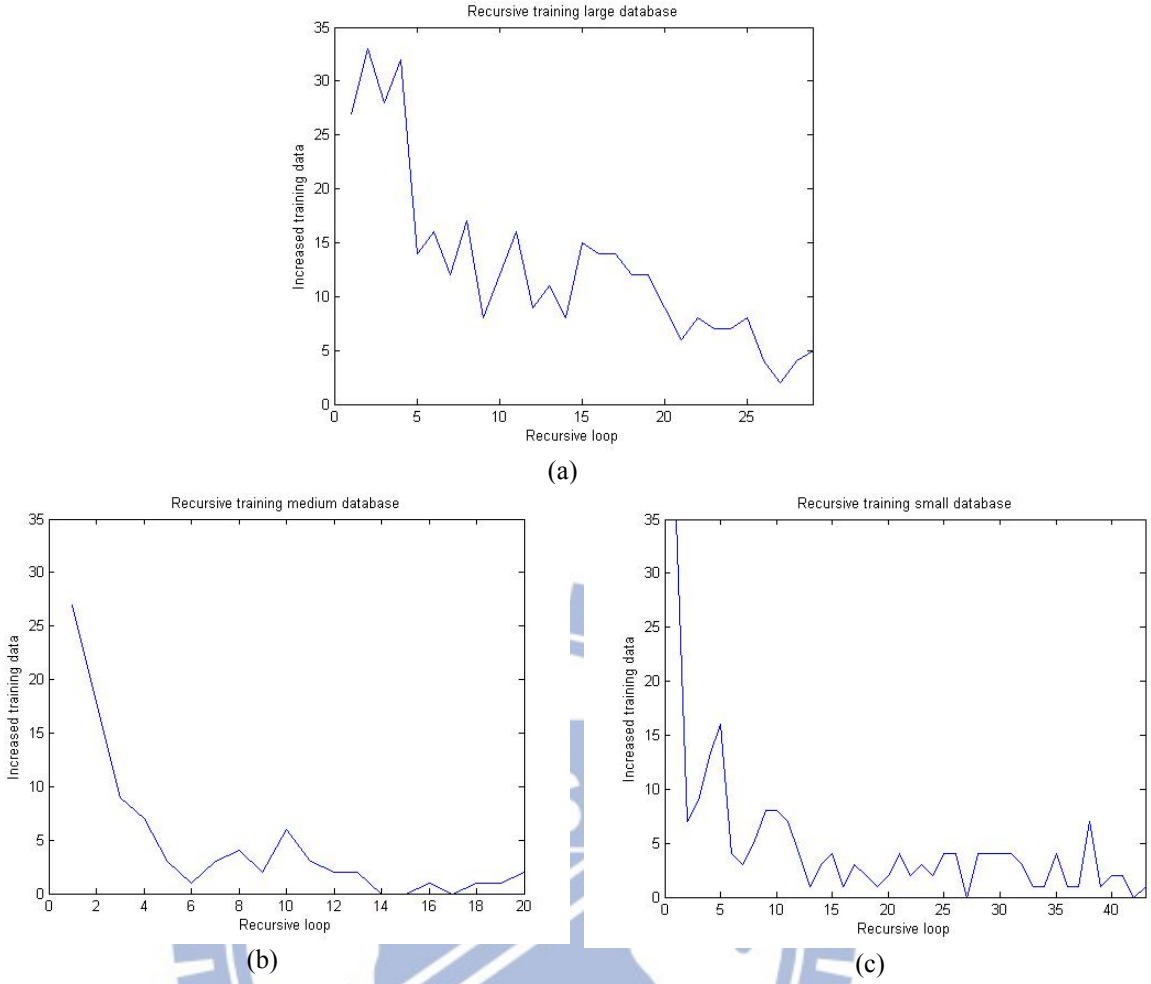


Figure 5-11: Recursive training curve of performing self-organized training data yielding method: (a) Coarse range, (b) Medium range, and (c) Fine range.

Table 5.12: Initial parameters of RGLS-HCCA training.

Parameters	Value of coarse range		Value of medium range		Value of fine range	
	PLE	SLE	PLE	SLE	PLE	SLE
P_{size}	60	20	40	20	40	20
N_c	20	none	20	none	20	none
Selection_Times	50	none	50	none	50	none
NormalTimes	10	none	10	none	10	none
ExploreTimes	15	none	15	none	15	none
Crossover Rate	0.6	0.7	0.5	0.5	0.6	0.5
Mutation Rate	0.2	0.1	0.2	0.1	0.1	0.05
$[M_{\min}, M_{\max}]$	[38, 45]	[38, 45]	[18, 25]	[18, 25]	[18, 25]	[18, 25]
$[m_{\min}, m_{\max}]$	[-9.5, 9.5]	[-9.5, 9.5]	[-8.5, 8.5]	[-8.5, 8.5]	[-14.5, 14.5]	[-14.5, 14.5]
$[\sigma_{\min}, \sigma_{\max}]$	[14, 16]	[14, 16]	[13, 15]	[13, 15]	[40, 43]	[40, 43]
Minimum_Support	Transaction Num/2	none	Transaction Num/2	none	Transaction Num/2	none
Minimum_Confidence	60%	none	60%	none	60%	none
RGLS parameter (λ)	0.004	0.004	0.003	0.003	0.001	0.001

(2) Comparison with existing neural network based image alignment systems

To compare the proposed MS-CNFN with other existing neural network-based systems

([42], [44], [87], and [88]), this paper carefully implements these systems according to the descriptions in their original paper. In this experiment, three typical comparisons including the alignment accuracy, speed, robustness, and real-image alignment testing are discussed in the following parts.

A. Alignment accuracy

In the training phase, since as using the same number of training images (i.e. 1165+137+219=1521) as the proposed CNFN on traditional neural network-based methods [42, 44, 87, 88] can yield large alignment error, we randomly generate another 4400 training images from the target alignment range described in Table 5.10 for training traditional methods. In the testing phase, we examine the alignment accuracy of MS-CNFN and other systems by using the same 600 testing images randomly generated from the target alignment range.

Figure 5.12 presents an example of a synthesized testing image on five different systems. The cross sign in Fig. 5.12 denotes the estimated results. In this figure, MS-CNFN can estimate more accurate position and orientation of the cross sign than other systems.

To proceed to analyze the alignment accuracy, Table 5.13 describes the average and standard deviation error of five image alignment systems for 15 runs using different testing images. From this table, MS-CNFN exhibits the lowest alignment error than other systems. The result indicates that the proposed MS-CNFN not only gets much higher alignment accuracy but also using fewer training data to reach better performance than other one-stage neural network methods.

To compare RGLS with the pseudo inverse method, 600 testing images are also used on the pseudo inverse method. The alignment results of the pseudo inverse method are: the average scaling error is 0.0097, the rotation error is 0.2619, the translation error for x-axis is 3.6024, and the translation error for y-axis is 1.9263. Thus, from the comparison of above alignment results with Table 5.13, RGLS would be superior to the pseudo inverse method.

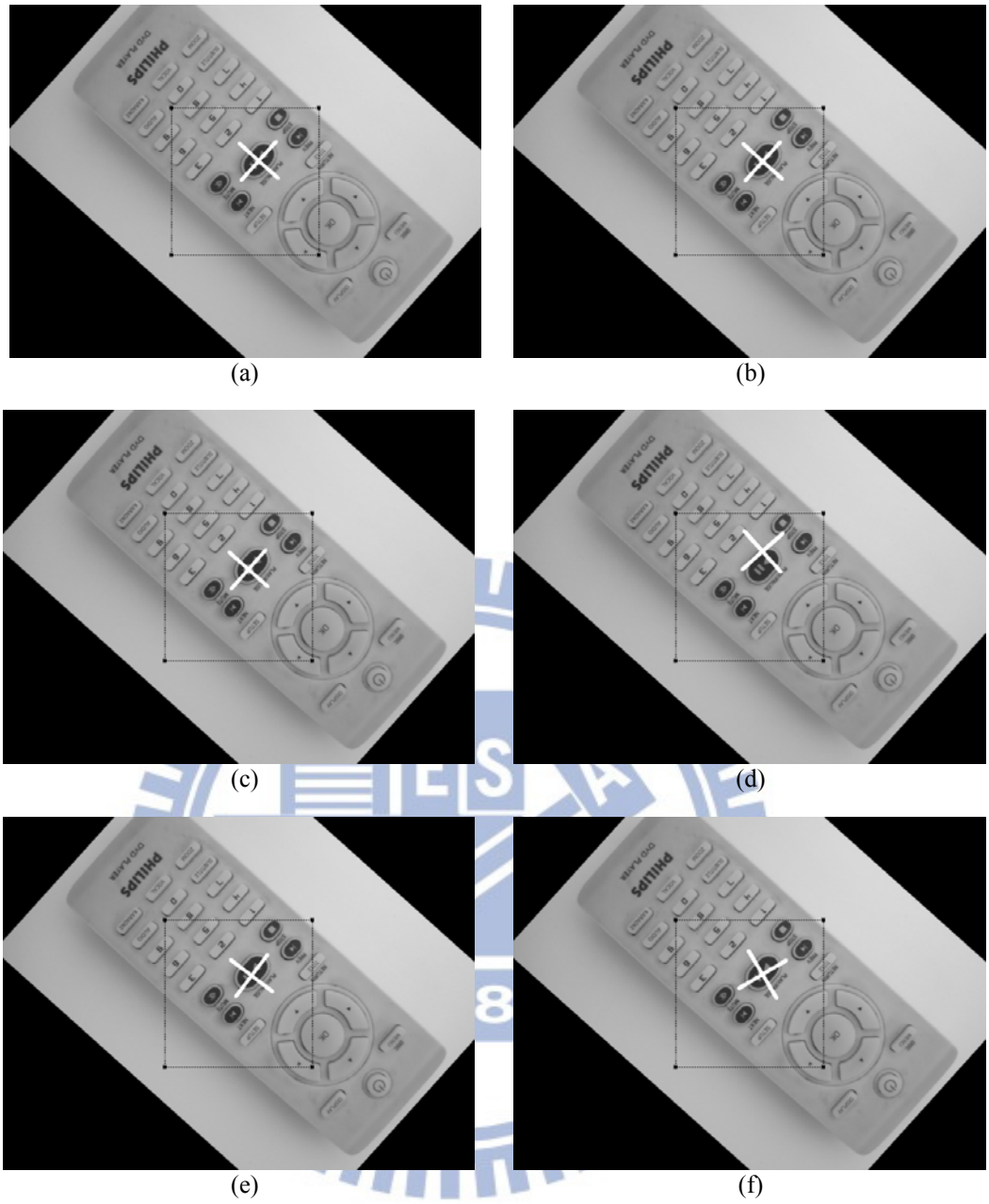


Figure 5-12: Alignment results for different systems: (a) Ground Truth, (b) MS-CNFN, (c) DCT, (d) FFT, (e) KICA, and (f) ISOMAP.

Table 5.13: Alignment errors in different image alignment systems.

Method	Errors							
	ErrScale		ErrAngle (degrees)		ErrDx (pixels)		ErrDy (pixels)	
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
MS-CNFN	**0.0095**	**0.0215**	**0.0344**	**0.1776**	**0.2766**	**0.1976**	**0.3883**	**0.4195**
DCT [44]	0.0302	0.0350	6.8495	8.8052	6.7206	10.0008	6.3597	10.6839
FFT [87]	0.0229	0.0348	7.9348	8.8924	9.7631	10.2108	9.0485	9.4451
KICA [88]	0.0333	0.0370	9.8534	14.1339	6.6953	10.9533	6.0219	9.5207
ISOMAP [42]	0.0670	0.0557	14.3922	21.0862	8.4077	14.4331	7.3752	9.7249

B. Alignment speed

In this experiment, 600 testing images are used to check the image alignment speed. The average execution time of MS-CNFN, DCT, FFT, KICA, and ISOMAP take about 0.103s, 0.078s, 0.085s, 0.226s, and 0.428s, respectively. From this result, the execution time of MS-CNFN is slightly slower than DCT and FFT methods, and is more efficient than KICA and ISOMAP methods.

C. Alignment Robustness

In this subsection, we further verify the robustness of MS-CNFN by adding different levels of random Gaussian noise. To achieve the aim of testing the robustness, 600 testing images are randomly generated with the addition of various strengths of Gaussian noise to examine different image alignment systems. Figure 5.13(a)-(d) presents the results of the absolute errors of the affine parameters under eight levels of SNR. As shown in these figures, MS-CNFN demonstrates much lower affine parameters error than other systems. This result indicates that the adopted Gabor-WGOH descriptor is not disturbed by a high noise level and so is the proposed RGLS-HCCA trained MS-CNFN. Figure 5.14 illustrates an image alignment example under a 10 dB signal-to-noise ratio (SNR) condition. From this figure, MS-CNFN depicts more accurate cross sign location than other methods.

Furthermore, except for Gaussian noise, the salt and pepper noise is add to the testing image at different pose from Fig. 5.14 which is used to check the robustness of the proposed system and other four other neural network-based systems. Figure 5.15 illustrates the alignment results of five methods. From this figure, MS-CNFN demonstrates more accurate cross sign location than other methods.

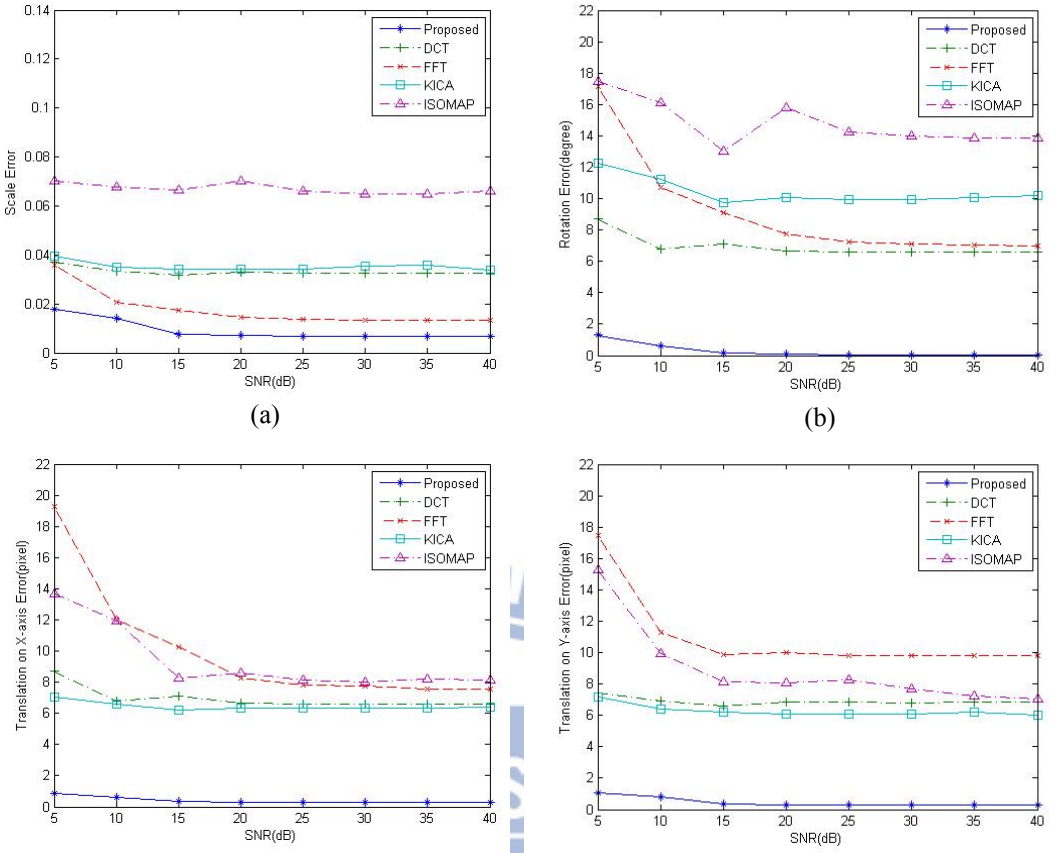
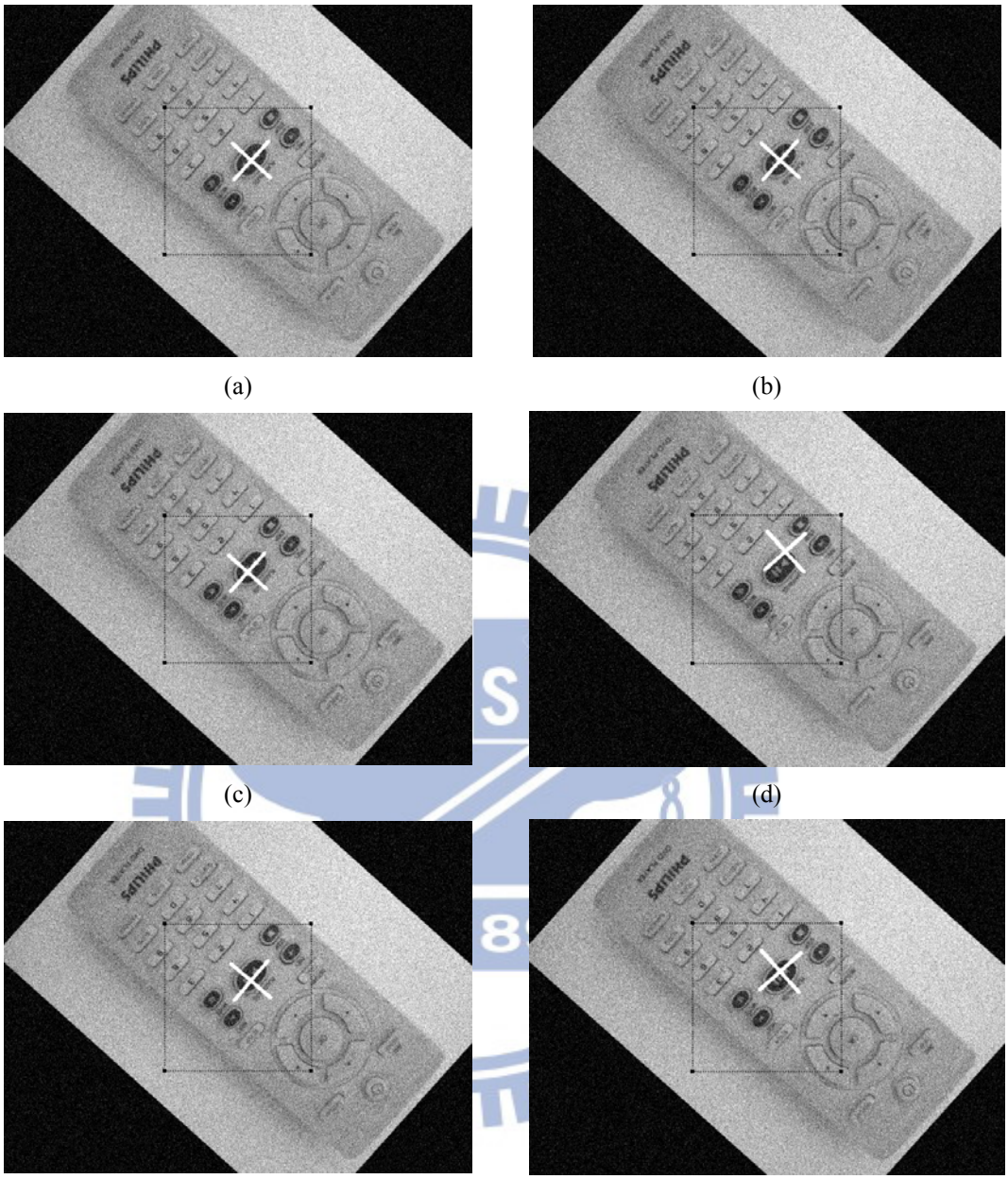


Figure 5-13: Average affine transformation errors comparison using MS-CNFN, DCT, FFT, KICA, ISOMAP under various SNR. Errors with respect to (a) scale, (b) rotation, (c) translation on X-axis, and (d) translation on Y-axis.



(a) (b)

(c) (d)

(e) (f)

Figure 5-14: Alignment results for different systems under 10 dB SNR condition: (a) Ground Truth, (b) MS-CNFN, (c) DCT, (d) FFT, (e) KICA, and (f) ISOMAP.

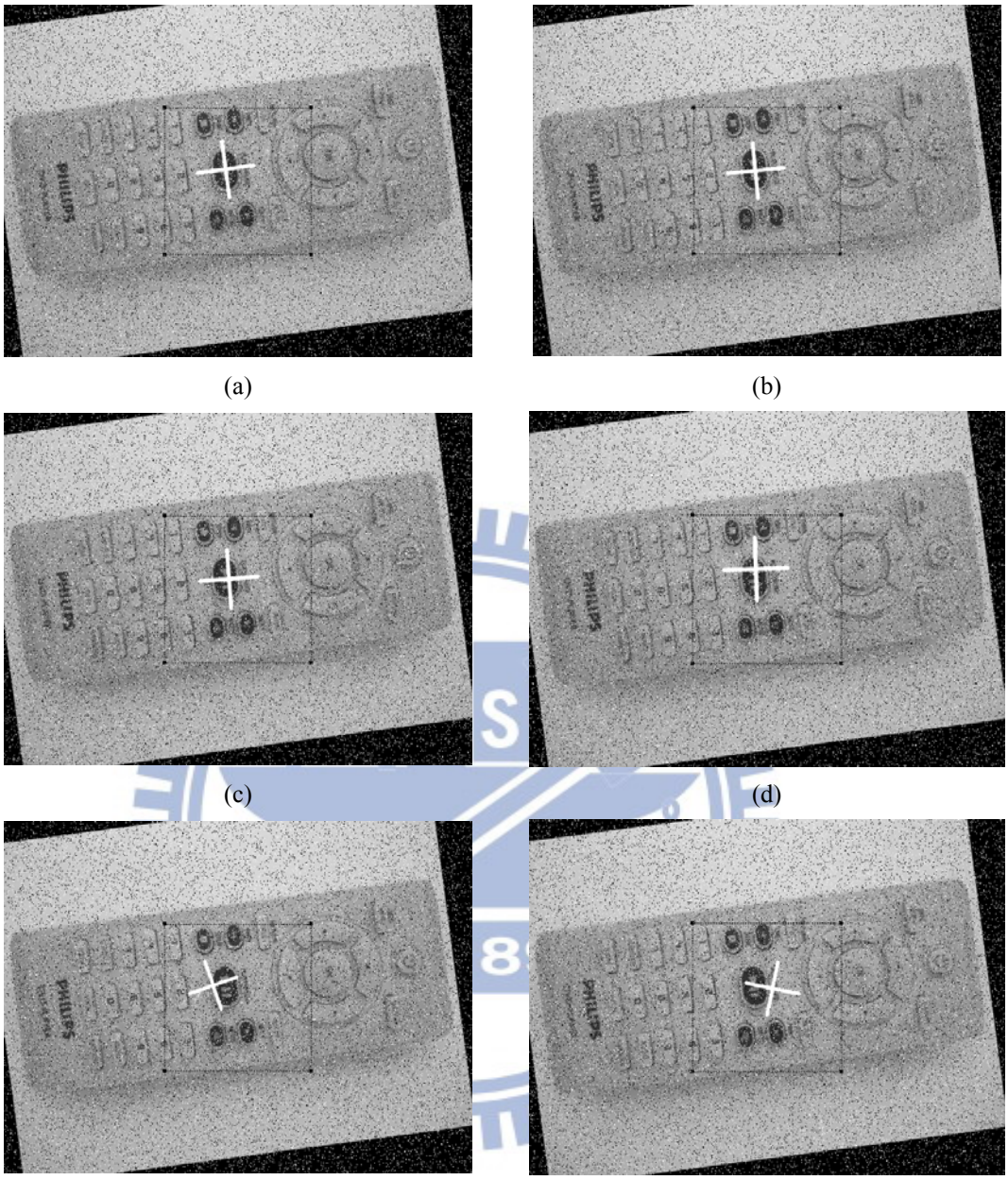


Figure 5-15: Alignment results for different systems under salt and pepper noise: (a) Ground Truth, (b) MS-CNFN, (c) DCT, (d) FFT, (e) KICA, and (f) ISOMAP.

D. Real-Image Alignment Testing

In addition to the synthesized images, real-image testing cases are used to verify the alignment performance of the proposed system. Figure 5.16(a)-(e) depicts the experimental results of aligning the same real image utilizing MS-CNFN, DCT, FFT, KICA, and ISOMAP, respectively. MS-CNFN demonstrates a more precise position and rotation of the cross sign

than other systems. Thus, applying the proposed image alignment system to real-image alignment cases with respective to large range of affine parameters is feasible.

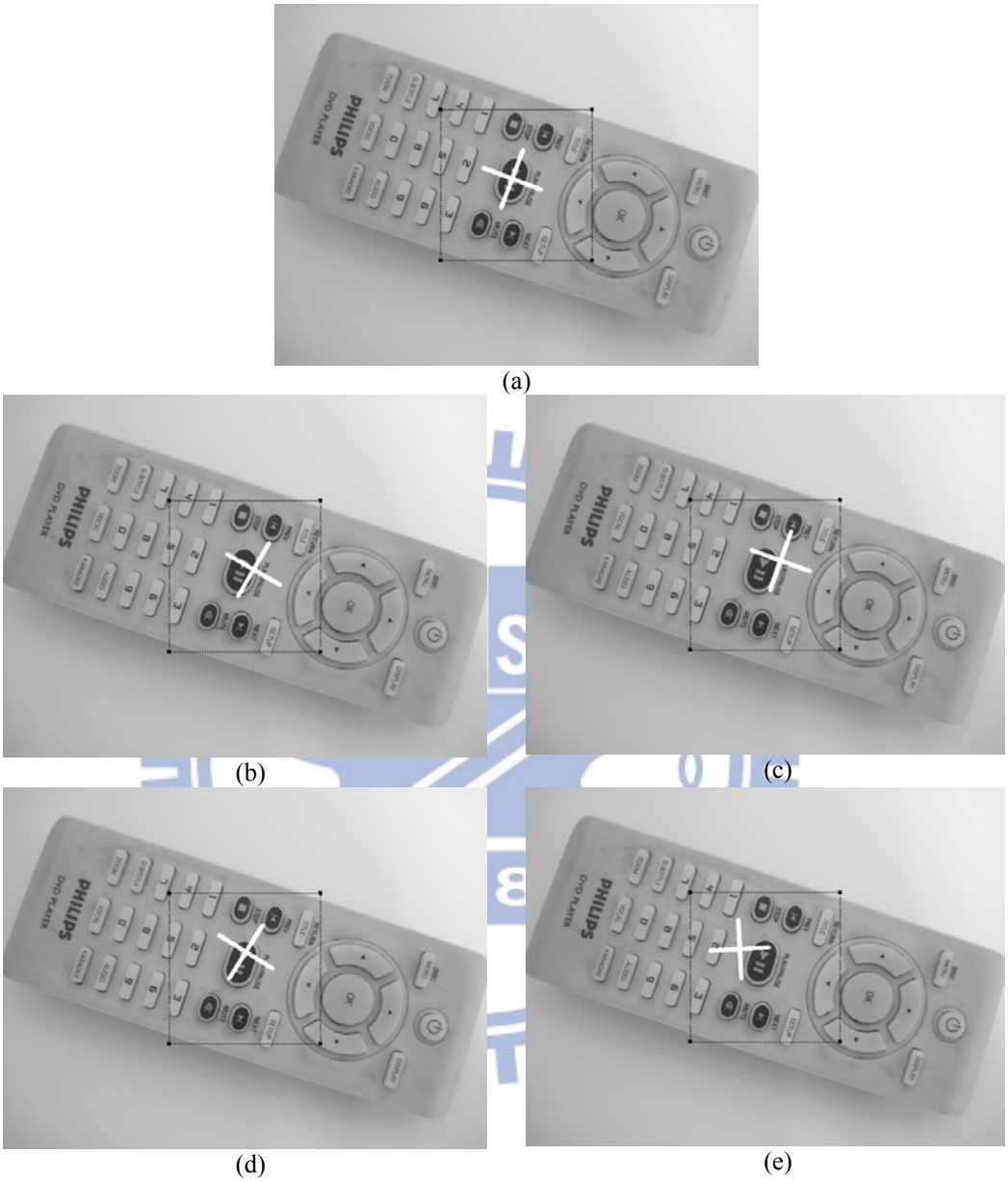


Figure 5-16: Results of image alignment on real images: (a) MS-CNFN, (b) DCT, (c) FFT, (d) KICA, and (e) ISOMAP.

Moreover, the circuit board inspection is another case of the real image testing. Figure 5.17(a) presents a template of a circuit board. Figure 5.17 (b)-(f) illustrate the alignment results of a circuit board with five different poses. As shown in these figures, every cross sign is located at an accurate position with a precise rotation. Therefore, the results imply that the proposed 2D image alignment system can be applied to a circuit board inspection system.

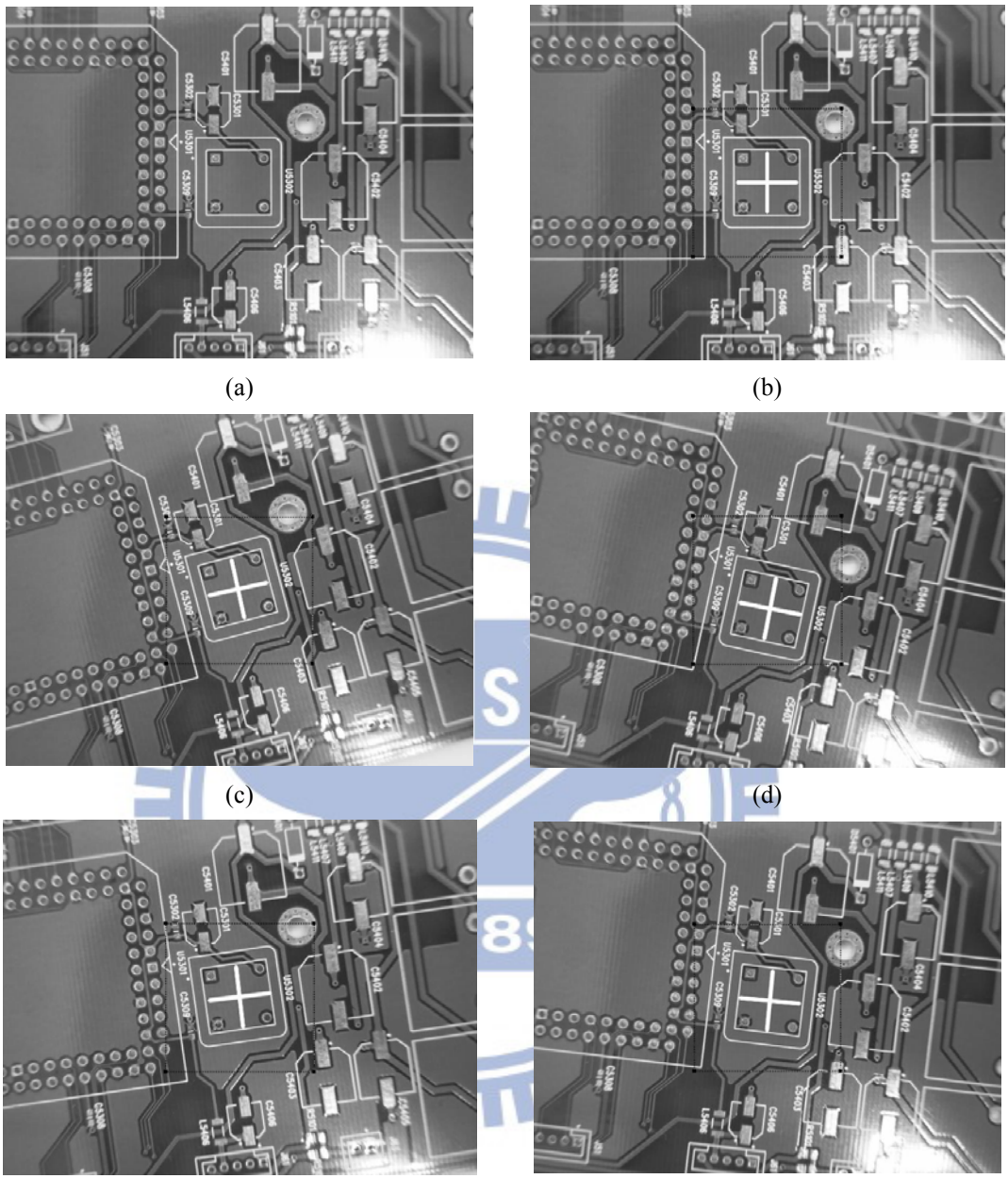


Figure 5-17: Results of image alignment on circuit board inspection images: (a) the template, (b) without rotation, (c) counterclockwise rotation, (d) clockwise rotation, (e) counterclockwise rotation, and (f) clockwise rotation.

5.3 Results of 3D Image Alignment

In the current section, a vehicle model depicted in Fig. 4.5 is selected as a reference model. The reference model is constructed by 4907 point clouds which are uniformly distributed on its surface. Thus, the aim of the 3D surface alignment task defined in the experiment is to align the arbitrary input 3D images (i.e. point clouds) with the reference model.

The experimental results comprise two parts. The first part uses the synthesized point cloud sets to test the proposed TNFN-based coarse alignment approach. In the second part, real 3D point cloud data scanned by a 3D imaging laser scanner are used to validate the alignment accuracy of the proposed fine alignment method. In both parts of the experiments, the alignment algorithm is compared with the neural network method (NNM) [46] and ICP [45] to demonstrate superior performance of the proposed coarse-to-fine scheme.

A. Testing using synthesized 3D point cloud data

To perform the coarse alignment learning, 2000 synthesized point cloud sets are generated randomly within the range described in Table 5.14. For training the TNFN, 50% of point clouds (1000) are prepared for training data set and the remaining 50% of point clouds (1000) are prepared for testing data set. The learning parameters for the TNFN training are defined in the left side of Table 5.15. Thus, after the coarse alignment learning completes, the output of TNFN is an estimated pose that coarsely aligns the input points with the reference model.

In TNFN-based surface modeling, we produce a cube model with the size of 5m×5m×5m that encloses the entire reference model. Within the cube model, 64000 point clouds are uniformly sampled according the resolution setting (0.125 m). Thus, the sampled point clouds are utilized for training TNFN to model the reference surface. The learning parameters of the TNFN-based surface modeling are defined in the right side of Table 5.15. Once the training of TNFN-based surface modeling is completed, the TNFN modeling is combined with the downhill simplex optimization method to execute the fine alignment of 3D surface.

Table 5.14: Range of 3D rigid transformation parameters.

3D rigid transformation parameter	Range of rigid transformation parameter
ϕ (degree), for roll	[-10 10]
φ (degree), for yaw	[-90 90]
θ (degree), for pitch	[0 90]
x(m)	[-0.2 0.2]
y(m)	[-0.2 0.2]
z(m)	[-0.2 0.2]

Table 5.15: Learning parameters for the TNFN training.

Parameters of training the TNFN	Value for coarse alignment		Value for surface modeling	
	PLE	SLE	PLE	SLE
P_{size}	40	25	80	25
N_c	20	none	20	none
Selection_Times	50	none	50	none
NormalTimes	10	none	10	none
ExploreTimes	15	none	15	none
Crossover Rate	0.6	0.6	0.8	0.7
Mutation Rate	0.3	0.4	0.1	0.4
$[M_{min}, M_{max}]$	[20, 35]	[20, 35]	[35, 40]	[35, 40]
$[m_{min}, m_{max}]$	[-15, 15]	[-15, 15]	[-2, 2]	[-2, 2]
$[\sigma_{min}, \sigma_{max}]$	[13, 15]	[13, 15]	[0.3, 0.9]	[0.3, 0.9]
Minimum_Support	TransactionNum/2	none	TransactionNum/2	none
Minimum_Confidence	60%	none	60%	none
RGLS parameter (λ)	0.0001	0. 0001	0.0005	0.0005
N-bin for MVFH	36	36	none	none

Because the execution time and alignment accuracy are two major issues for a 3D image alignment system, these elements are taken as the evaluation conditions to examine the proposed alignment system.

(1) Alignment accuracy

To evaluate the alignment accuracy, the proposed TNFN-based coarse-to-fine system is compared with NNM [46] and ICP [45], two methods that use PCA for coarse alignment. Thus, based on the 1000 testing sets of point clouds, the alignment errors of the coarse and fine alignments are listed in Table 5.16 where RMSE indicates the root mean square error. From this table, the proposed system exhibits the lowest coarse and fine alignment errors among all systems. In addition, the proposed method improves the PCA coarse alignment, as shown in the table. Figure 5.18(a) and (b) presents a coarse alignment example of PCA and the proposed TNFN-based method, where the blue and red point clouds represent the testing and reference model data, respectively. From this figure, the proposed method exhibits less alignment error than PCA.

To compare RGLS with the pseudo inverse method, this paper uses the same 1000 testing sets of point clouds on the pseudo inverse method. The RMSE of the pseudo inverse method for the coarse phase is 0.2619, which is larger than RGLS (0.1042). Thus, in the 3D image alignment task, RGLS would be better than the pseudo inverse method. In short, from

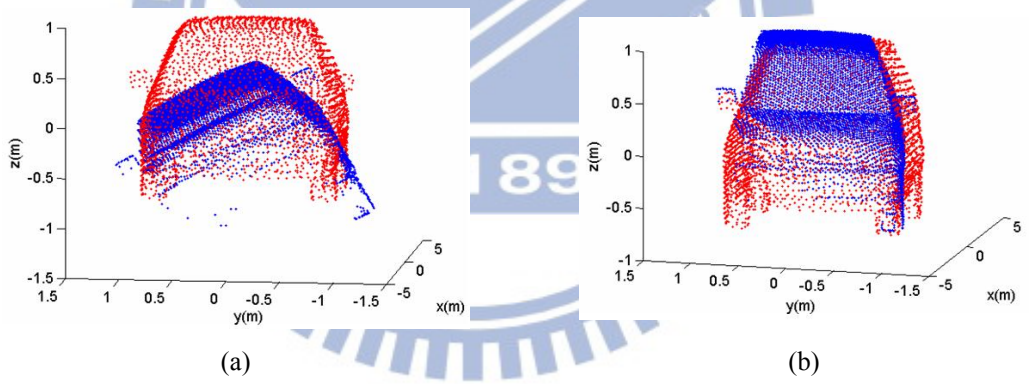
example 1 to example 3, we conclude that RGLS would be more suitable than the pseudo inverse method for constructing a TNTN.

(2) Alignment speed

In consideration of alignment speed, the average execution time for aligning 1000 testing sets of point clouds is calculated. The results of the alignment speed are also listed in Table 5.16. From the table, the execution time of the proposed system is shorter than those of NNM and ICP.

Table 5.16: Results of alignment accuracy and execution time.

Method	Average RMSE (m)		Average execution Time (sec)
	Coarse alignment error	Fine alignment error	
TNFN-based coarse-to-fine alignment	**0.1042m**	**0.0627m**	**3.29s**
PCA coarse alignment NNM fine alignment	0.2846m	0.1423m	4.53s
PCA coarse alignment ICP fine alignment	0.2846m	0.0688m	49.48s



(a) (b)

Figure 5-18: Examples of two coarse alignment methods: (a) PCA and (b) TNFN-based coarse alignment.

B. Validation of real 3D point cloud data alignment

Figure 5.19 presents a real case of 3D point cloud data scanned by a 3D imaging laser scanner. The image size of the scanned scene is 256×256 with 20 degree field of view. In the 3D scenery, the vehicle region is extracted by using the segmentation algorithm described in [64]. The extracted vehicle data is then used to validate the alignment performance of the proposed system, NNM and ICP. Figure 5.20 (a) and (b) show the coarse alignment results of PCA (used for NNM [46] and ICP [45] in coarse phase) and the proposed TNFN-based

method. In this figure, the coarse alignment errors of PCA and the proposed method are 0.262 and 0.106m, respectively. Thus, this result again proves that the proposed method is superior to PCA. In the case of fine alignment, Fig. 5.21(a)-(c) depicts the fine alignment results of proposed TNFN-based fine alignment system, NNM, and ICP. From this figure, the fine alignment errors of the proposed system, NNM, and ICP are 0.0558, 0.1121, and 0.0569m, respectively. These results indicate that the proposed TNFN-based method can achieve high accuracy in real 3D point cloud data. Furthermore, regarding the alignment speed, the execution time of the proposed system, NNM, and ICP are 1.71, 2.13, and 7.93s, respectively. Therefore, the proposed system demonstrates higher alignment speed compared to NNM and ICP. In short, the proposed TNFN-based coarse-to-fine 3D image alignment system can align 3D point cloud data with the reference model accurately at high speed.

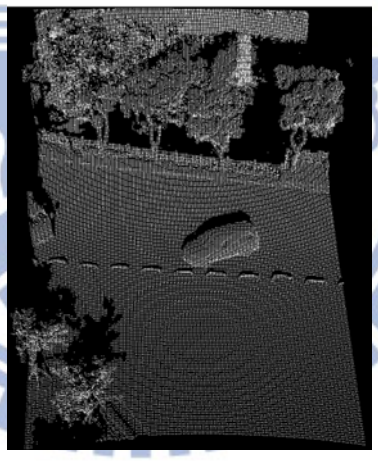
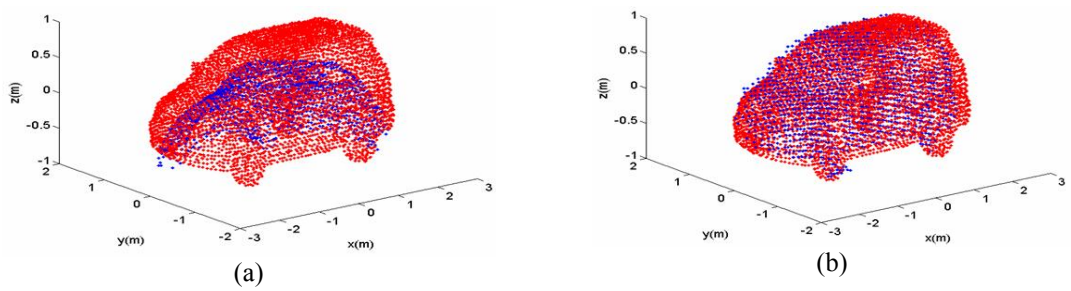
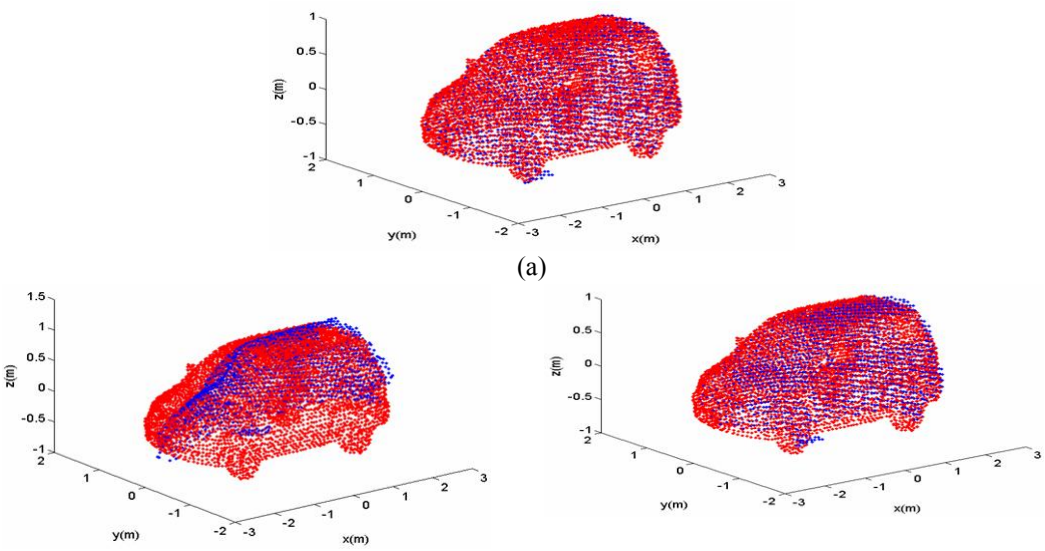


Figure 5-19: Real case of 3D point cloud data scanned by a 3D imaging laser scanner.



(a) (b)

Figure 5-20: Coarse alignment results: (a) PCA and (b) TNFN-based coarse alignment.



(a)

(b) (c)

Figure 5-21: Fine alignment results: (a) TNFN-based fine alignment, (b) NNM, and (c) ICP.

Chapter 6
Conclusions and Future Works

The purpose of this dissertation is to develop a methodology to automatically design TSK-type neural fuzzy networks (TNFNs) such that the developed networks can be applied to real world problems. To make TNFNs to be useful, the learning algorithm must be powerful to evolve networks in simulation that are robust enough to transfer to the real world. Toward this end, two components have been involved to achieve this goal: regularized least square based cooperative coevolutionary algorithm (RGLS-HCCA) and image alignment applications. The RGLS-HCCA model can evolve the structure and parameters of TNFN and the evolved TNFN can be taken to transfer the problem from simulation to the real world applications.

This chapter summarizes the conclusions of these two components in Section 6.1 and discusses future works to extend the proposed algorithm in Section 6.2.

6.1 Conclusions

This dissertation concludes two key components to the fields of evolutionary computation and its applications. Regarding the first component, the proposed RGLS-HCCA encodes an antecedent part of a TSK-type fuzzy rule into a chromosome and utilizes RGLS to estimate the consequent part of a TSK-type fuzzy rule. Such combination not only reduces the number of parameters that must be trained but also controls HCCA to adapt the network to more complex tasks. In HCCA, it proposes parameter level evolution (PLE) and structure level evolution (SLE) to solve the problem of the random group selection, preserve the good combinations of fuzzy rules, and make the parameters and structure of network be evolved locally and globally, respectively. In addition, this dissertation proposes VAC, VAM, and SRM such that the variable length of chromosomes can be evaluated and the number of fuzzy

rules can be self-adjusted. The experimental results show that by applying RGLS-HCCA to the prediction of Mackey-Glass time series, RGLS-HCCA would demonstrate faster the algorithm convergence rate and lower estimating error than those of other learning algorithms.

Regarding the second component, two image alignment applications, which are 2D and 3D image alignment problems, are used to demonstrate the applicability of RGLS-HCCA. For 2D image alignment application, RGLS-HCCA is used to construct a CNFN-based 2D image alignment system. The CNFN utilizes the multi-stage of TNFN to solve problems that one-stage neural network have difficulty in applying a large range of affine parameters. This evidence can be found in the experimental results of both synthesized and real-images cases. The results show that the performance of the proposed scheme is superior to the traditional neural network methods on accuracy and robustness. For 3D image alignment application, the use of RGLS-HCCA can benefit the training of the TNFN-based coarse-to-fine 3D image alignment system. In the coarse alignment procedure, utilizing RGLS-HCCA to train a TNFN to model the relationship between the input feature and output pose can solve the problem of the high alignment error caused by PCA. In fine alignment procedure, using RGLS-HCCA to train a TNFN to model the reference surface can improve the heavy computational cost caused by ICP. In addition, by combining the surface modeling with the downhill simplex optimization, the distance from the input image to the reference model can be reduced iteratively. The evidence can be found in the experimental results to demonstrate the superior performance of the proposed 3D image alignment system over existing systems.

In summary, the most contributions of this dissertation are the proposed RGLS-HCCA for solving the problems that current evolutionary algorithms suffer from and verify the applicability of RGLS-HCCA to real world problems.

6.2 Future Works

The future works of the proposed RGLS-HCCA and the image alignment applications

are discussed as follows:

To discuss the proposed RGLS-HCCA, the number of hierarchical level is only two to execute the training of structure and parameters of neural fuzzy networks. As the application problem become more complex, there is a need to increase the hierarchical level to match the complex problem. Thus, in the future work, the multi hierarchical level is taken into consideration of further investigation of how to cooperate these hierarchical levels to adapt the model to a complex problem.

For the image alignment applications, two tasks are considered: 2D image alignment and 3D image alignment. For the 2D image alignment task, although the proposed system can demonstrate high performance, it still has some limitations. Specifically, as the application problem becomes more complicated, the number of cooperative neural fuzzy networks would increase. Such condition leads the proposed model to suffer from the difficulty of choosing the suitable number of cooperative networks. If the unsuitable number of networks is chosen, the overall system will yield large estimated errors. Therefore, future works should identify a well-defined method to determine the number of cooperative neural fuzzy networks automatically.

For the 3D image alignment task, in spite of combing the surface modeling with the downhill simplex optimization can obtain good results in fine alignment phase, the downhill simplex optimization may suffer from getting in local minima. Toward this end, the on-line parallel search techniques may be the solution for preventing the local minima happened. The on-lien parallel search techniques should be fast and keep the proper accuracy for applying to the fine alignment task. Therefore, the future work would modify the proposed RGLS-HCCA model to satisfy the design of the fine alignment phase.

Bibliography

[1] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Systems Man Cybern.*, vol. 22, no. 6, pp. 1414-1427, 1992.

[2]C. J. Lin and C. C. Peng, "Identification and prediction using neuro-fuzzy networks with symbiotic adaptive particle swarm optimization," *Informatica*, vol. 35, no. 1, pp. 113-122, 2011.

[3] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Systems*, vol. 5, no. 4, pp. 477-496, 1997.

[4] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Systems Man Cybern.*, vol. 15, pp. 116-132, 1985.

[5] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, Prentice-Hall, Englewood Cliffs, NJ, May 1996.

[6] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Systems*, vol. 6, no. 1, pp. 12-31, 1998.

[7] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Trans. Systems Man Cybern.*, vol. 32, no. 2, pp. 176-190, 2002.

[8] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, 1990.

[9] C. J. Lin and C. C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Systems Man Cybern. Part B*, vol. 34, no. 5, pp. 2144-2154, 2004.

[10] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Networks*, vol. 10, no. 4, pp. 828-845, 1999.

[11] D. E. Goldberg, *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley, 1989.

[12] C. J. Lin and Y. C. Hsu, "Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 4, pp. 729-745, 2007.

[13] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, pp. 11-32, 1996.

[14] F. J. Gomez, "Robust non-linear control through neuroevolution," Ph. D. Disseration, Dep. Computer Sciences, Univ. Texas of Austin, USA, 2003.

[15] Y. C. Hsu, S. F. Lin, and Y. C. Cheng, "Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5320-5330, 2010.

[16] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," *in Proc. Int. Conf. Genetic Algorithms*, San Diego, CA, USA, pp. 450-457, July 1991.

[17] C. T. Lin and C. P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Trans. Systems Man Cybern. Part B*, vol. 30, no. 2, pp. 276-289, 2000.

[18] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Systems Man, Cybern. Part B*, vol. 30, no. 2, pp. 290-302, 2000.

[19] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets and Systems*, vol. 80, no. 3, pp. 273-293, 1996.

[20] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "VGA-classifier: design and applications," *IEEE Trans. Systems Man and Cybern. Part B*, vol. 30, no.6 , pp. 890-895, 2000.

[21] K. S. Tang, "Genetic algorithms in modeling and optimization," Ph. D. Dissertation, Dep.

Electronic Engineering, City Univ. Hong Kong, Hong Kong, 1996.

[22] C. F. Juang, "Combination of online clustering and Q-value based GA for reinforcement fuzzy system design," *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 3, pp. 289–302, 2005.

[23] F. Gomez and J. Schmidhuber, "Co-evolving recurrent neurons learn deep memory POMDPs," *in Proc. Conf. Genetic and Evolutionary Computation*, Washington, DC, USA, pp. 491-498, June 25-29, 2005.

[24] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithm," Ph. D. Disseration, Dep. of Computer Sciences, University North Carolina Charlotte, USA, 1999.

[25] S. K. Tanbeer, C. F. Ahmed, and B. S. Jeong, "Parallel and distributed algorithm for frequent pattern mining in large database," *IETE Tech Rev*, vol. 26, no.1, pp. 55-65, 2009.

[26] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," *in Proc. Int .Conf. Very Large Data Base*s, Santiago, Chile, pp. 487-499, September 12-15, 1994.

[27] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *in Proc. Int. Conf. Management of Data,* Dallas, Texas, USA, pp.1-12, May 16-18, 2000.

[28] Y. T. Wu, Y. J. An, J. Geller, and Y. T. Wu, "A data mining based genetic algorithm," in *Proc. IEEE Int. Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Collaborative Computing*, *Integration*, *and Assurance*, Gyeongju, Korea, pp. 6, April 27-28, 2006.

[29] S. Shankar and T. Purusothaman, "Utility sentient frequent itemset mining and association rule mining: a literature survey and comparative study," *International Journal of Soft Computing Applications*, No. 4, pp. 81-95, 2009.

[30] X. J. Liu, J. Yang, and H. B. Shen, "Automatic image registration by local descriptors in remote sensing," *Optical Engineering*, vol. 47, no. 8, 087206, 2008.

[31] X. M. Peng, W. Chen, and Q. Ma, "Feature-based nonrigid image registration using

Hausdorff distance matching measure," *Optical Engineering*, vol. 46, no. 5, 057201, 2007.

[32] D. Skea, I. Barrodale, R. Kuwahara, and R. Poeckert, "A control point matching algorithm," *Pattern Recognition*, vol. 26, no 2, pp. 269-276, 1993.

[33] S. Manickam, S. D. Roth, and T. Bushman, "Intelligent and optimal normalized correlation for high-speed pattern matching," *NEPCON. WEST 2000*, Anaheim, California, USA, pp. 191-206, February 27-March 2, 2000.

[34] R. J. Althof, M.G. J. Wind, and J. T. Dobbins, "A rapid and automatic image registration algorithm with subpixel accuracy," *IEEE Transactions on Medical Imaging*, vol. 16, no. 3, pp. 308-316, 1997.

[35] L. M. G. Fonseca and B. S. Manjunath, "Registration techniques for multisensor remotely sensed imagery," *Photogrammetric Engineering and Remote Sensing* , vol. 62, no. 9, pp. 1049-1056, 1996.

[36] S. Kaneko, I. Murase, and S. Igarashi, "Robust image registration by increment sign correlation," *Pattern Recognition* , vol. 35, no. 10, pp. 2223-2234, 2002.

[37] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858-1865, 2008.

[38] M. Amintoosi, M. Fathy, and N. Mozayani, "Precise image registration with structural similarity error measurement applied to superresolution," *EURASIP Journal on Advances in Signal Processing*, Article ID 305479, 7 pages, 2009.

[39] B. Zitova and J. Flusser, "Image registration methods: A survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977-1000, 2003.

[40] I. Elhanany, M. Sheinfeld, A. Beckl, Y. Kadmon, N. Tal, and D. Tirosh, "Robust image registration based on feedforward neural networks," *in Proceedings of IEEE International Conference on System, Man and Cybernetic*, NASHVILLE, TN, vol. 2, pp.

1507 – 1511, October 8-11, 2000.

[41] J. Wu and J. Xie, "Zernike moment-based image registration scheme utilizing feedforward neural networks," *in Proceedings of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, China, vol. 5, pp. 4046-4048, June 15-19, 2004.

[42] A. B. Xu and P. Guo, "Isomap and neural networks based image registration scheme," *Lecture Notes in Computer Science*, vol. 3972, pp. 486-491, 2006.

[43] A. B. Xu and P. Guo, "Image registration with regularized neural network," *Lecture Notes in Computer Science*, vol. 4233, pp. 286-293, 2006.

[44] H. Sarnel and Y. Senol, "Accurate and robust image registration based on radial basis neural networks," *Neural Comput. & Applic.*, vol. 20, no. 8, pp. 1255-1262, 2011.

[45] H. Liu, J. Yan, and D. Zhang, "Three-dimensional surface registration: A neural network strategy," *Neurocomputing*, vol. 70, pp. 597-602, 2006.

[46] J. Zhang, Y. Ge, S. H. Ong, C. K. Chui, S. H. Teoh, and C. H. Yan, "Rapid surface registration of 3D volumes using a neural network approach," *Image and Vision Computing*, vol. 26, pp. 201-210, 2008.

[47] A. Ghafoor, R.N. Iqbal, and S. Khan, "Robust image matching algorithm," *4th EURASIP Conference Focused on Video/Image Processing and Multimedia Communications*, Zagreb, Croatia, vol. 1, pp. 155-160, July 2-5, 2003.

[48] D. Chetverikov, D. Stepanov, and P. Kresk, "Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm," *Image and Vision Computing*, vol. 23, pp. 299-309, 2005.

[49] Y. H. Liu, "Improving ICP with easy implementation for free-form surface matching," *Pattern Recognition*, vol. 37, pp. 211-226, 2004.

[50] J. J. Jack and C. Roux, "Registration of 3-D images by genetic optimization," *Pattern Recognition Letters*, vol. 16, pp. 823-841, 1995.

[51] N. M. Alpert, J. F. Bradshaw, D. Kennedy, and J. A. Correia, "The principal axes

transformation – a method for image registration," *Journal of Nuclear Mdeicine*, vol. 31, pp. 1717-1722, 1990.

[52] P. Besl and N. Mckay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239-256, 1992.

[53] J. Peng, V. Strela, and D. Zorin, "A simple algorithm for surface denoising," *in Proceedings of IEEE Visualization*, San Diego, CA, USA, pp. 107-112, October 2001.

[54] S. Rusinkiewicz and M. Levoy, "Efficient variants of ICP algorithm," *in Prcoceeding s of Third International Conference on 3D Digital Imaging and Modeling*, Quebec City, Canada, pp. 145-152, May 28-June 1, 2001.

[55] G. Blais and M.Levine, "Registering multiview range data to create 3D computer objects," *IEEE on Trans. PAMI*, vol. 17, no. 8, pp. 820-824, 1995.

[56] C. C. Wu and S. F. Lin, "Efficient model detection in point cloud data based on bag of words classification," *Journal of Computational Information Systems*, vol. 7, no. 12, pp. 4170-4177, 2011.

[57] Y. J. Xu and C, J. Lin, "Efficient reinforcement learning through dynamic symbiotic evolution for TSK-type fuzzy controller design," *International Journal of General Systems*, Vol. 34, No. 5, pp. 559-578, 2005.

[58] Y. J. Xu and C. J. Lin, "A novel evolution learning for recurrent wavelet-based neuro-fuzzy networks," *Soft Comput.*, vol. 10, no.3, pp. 193-205, 2006.

[59] Y. J. Xu, C. J. Lin, and C. Y. Lee, "Supervised and reinforcement evolutionary learning for wavelet-based neuro-fuzzy networks," *J. Intell. Robot Syst.*, vol. 52, no. 2, pp. 285-312, 2008.

[60] C. J. Lin, C. H. Chen, and C. T. Lin, "Efficient self-evolving evolutionary learning for neurofuzzy inference systems," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1476-1490, 2008.

[61] A. N. Tikhonov, "On solving incorrectly posed problems and method of regularization,"

Doklady Akademii Nauk USSR, vol. 153, pp. 501-504, 1963.

[62] M. C. D. Almeida, A. V. Garcia, and E. N. Asada, "Regularized least squares power system state estimation," *IEEE Trans. Power Systems*, vol. 27, no.1, pp. 290-297, 2012.

[63] S. Shidong, "A Levenberg-Marquardt method for large-scale bound constrained nonlinear least-squares," Master Thesis, University of British Columbia, Canada, 2008.

[64] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Trans. Control Systems Technology*, vol. 2, no. 2, pp. 123-134, 1994.

[65] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Systems Man Cybern*, vol. 16, no. 1, pp. 93-101, 1986.

[66] O. Sigaud, C. Salaun, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: A survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115-1129, 2011.

[67] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its applications to prediction of a complex system," *Fuzzy Sets Syst.*, vol. 42, no. 3, pp. 315–334, 1991.

[68] T. Rabbani, F. A. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *in Proc. of ISPRS*, Dresden, Germany, pp. 248-253, Sep. 25-27, 2006.

[69] X. Wu, C. Zhang, and S. Zhang, "Mining both positive and negative association rules," *Proceedings of the 19th International Conference on Machine Learning*, Sydney, Australia, pp. 658–665, July 8-12, 2002.

[70] P. L. Hsu, R. Lai, C. C. Chiu, and C. I. Hsu, "The hybrid of association rule algorithm and genetic algorithms for tree induction: an example of predicting the student course performance," *Expert Systems with Applications*, vol. 25, no. 1, pp. 51-62, 2003.

[71] X. Yan, C. Zhang, and S. Zhang, "Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3066-3076, 2009.

[72] C. Chai and B. Li , "A novel association rules method based on genetic algorithm and fuzzy set strategy for web mining," *Journal of Computers*, vol. 5, no. 9, pp. 1448-1455, 2010.

[73] R. Surendiran, K. P. Rajan, and M. S. Kumar, "Study on the customer targeting using association rule mining," *International Journal on Computer Science and Engineering*, vol. 2, no. 7, pp. 2483-2485, 2010.

[74] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases, Advances in Fuzzy Systems-Applications and Theory*, vol.19, World Scientific Publishing, NJ, USA, 2001.

[75] Y. P. Zou, Z. K. Mi, and M. H. Xu, "Dynamic load balancing based on roulette wheel selection," *in Proc. IEEE Int. Conf. Communications, Circuits and Systems*, Guilin, China, vol. 3, pp.1732-1734, June 25-28, 2006.

[76] G. Lin and X. Yao, "Analysing crossover operators by search step size," *in Proc. IEEE Int. Conf. Evolutionary Computation*, Indianapolis, USA, pp. 107-110, April 13-16, 1997.

[77] E. Cox, *Fuzzy modeling and genetic algorithms for data mining and exploration*, Morgan Kaufman Publications, San Francisco, USA, 1st edition, 2005.

[78] S. Abedi and R. Tafazolli, "Genetically modified multiuser detection for code division multiple access systems," *IEEE Journal on Selected Areas*, pp. 1884-1887, 2008.

[79] I. Dempsey, "Constant generation for the financial domain using grammatical evolution," *Genetic and Evolutionary Computation Conference workshop program*, Washington, D.C., USA, pp. 350-353, June 25-29, 2005.

[80] R. T. Chin, "Automatic visual inspection: 1981 to 1987," *Computer Vision, Graphics,*

and Image Processing , vol. 41, no. 3, pp. 346 – 381, 1988.

[81] T. S. Newman and A. K. Jain, "A survey of automatic visual inspection," *Computer Vision and Image Understanding*, vol. 61, pp. 231 – 262, 1995.

[82] M. Moganti, F. Ercal, C. H. Dagli, and S. Tsunekawa, "Automatic PCB inspection algorithms: a survey," *Computer Vision and Image Understanding*, vol. 63, pp. 287 – 313, 1996.

[83] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[84] L. W. Peng and S. M. Shamsuddin, "3D object reconstruction and representation using neural networks," *in Proceedings GRA-PHITE 2004*, Singapore, pp. 139-147, 15-18 June, 2004.

[85] C. R. Maurer, R. J. Maciunas, and J. M. Fitzpatrick, "Registration of head CT Images to physical space using a weighted combination of points and surfaces," *IEEE Trans. on medical imaging*, vol. 17, no. 5, pp. 753-761, 1998.

[86] M. Hofmeister, P. Vorst, and A. Zell, "A comparison of efficient global image features for localizing small mobile robots," *in Proceedings of ISR/ROBOTIK*, Munich, Germany, pp. 143-150, June 7-9, 2010.

[87] A. B. Abche, F. Yaacoub, A. Maalouf, and E. Karam, "Image registration based on neural network and Fourier transform," *in Proceedings of the 28th IEEE EMBS annual international conference*, New York, USA, pp. 1460-1463, , Aug.30-Sep. 3, 2006.

[88] A. Xu, X. Jin, P. Guo, and R. Bie "KICA feature extraction in application to FNN based image registration," *International Joint Conference on Neural Networks*, Vancouver, BC, Canada , pp. 3602-3608, July 16-21, 2006.

[89] S. Wei and S. Lai, "Robust and efficient image alignment based on relative gradient

matching," *IEEE Trans. on Image Processing* , vol. 15, no. 10, pp. 2936-2943, 2006.

[90] C. Y. Hsu, Y. C. Hsu, and S. F. Lin, "A hybrid learning neural network based image alignment system using global feature selection approach," *Advances in Computer Science and Engineering*, vol. 6, no. 2, pp. 129-157, 2011.

[91] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[92] D. M. Bradley, R. Patel, N. Vandapel, and S. M. Thayer, "Real-time image-based topological localization in large outdoor environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, pp. 3670-3677, August 2005.

[93] P. Moreno, A. Bernardion, and J. S. Victor, "Improving the SIFT descriptor with smooth derivative filters," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 18-26, 2009.

[94] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," *IEEE international Conference on Intelligent Robots and Systems*, Taipei, Taiwan, pp. 2155-2162, Oct. 18-22, 2010.

[95] R. B. Rusu, N. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3D point clouds," *in Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, Baden-Baden, Germany, pp. 119-128, July 24, 2008.

[96] W. Spendley, G. R. Hext, and F. R. Himsworth, "Sequential application of simplex designs in optimization and evolutionary operation," *Technometrics* , vol. 4, pp. 441-461, 1962.

[97] D. Lee and M. Wiswall, "A parallel implementation of the simplex function minimization routine," *Computational Economics*, vol. 30, no. 2, pp. 171-187, 2007.

[98] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph. D. Dissertation, Dep. Computer and Communication Sciences, Univ. Michigan, Ann Arbor, MI, 1975.

[99] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Trans*. *Syst*., *Man*, *Cybern*., vo1. 6, no. 1, pp. 122-128, 1986.

[100] T. Pahikkala and J. Boberg, “Fast n-fold cross-validation for regularized least-squares,” In Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence, Espoo, Finland, pp. 83-90, October 2006.

[101] A. S. Lapedes and R. Farber, “Nonlinear signal processing using neural networks: prediction and system modeling,” *Tech. Rep. LA-UR-87- 2662*, Los Alamos Nat. Lab., Los Alamos, NM, 1987.

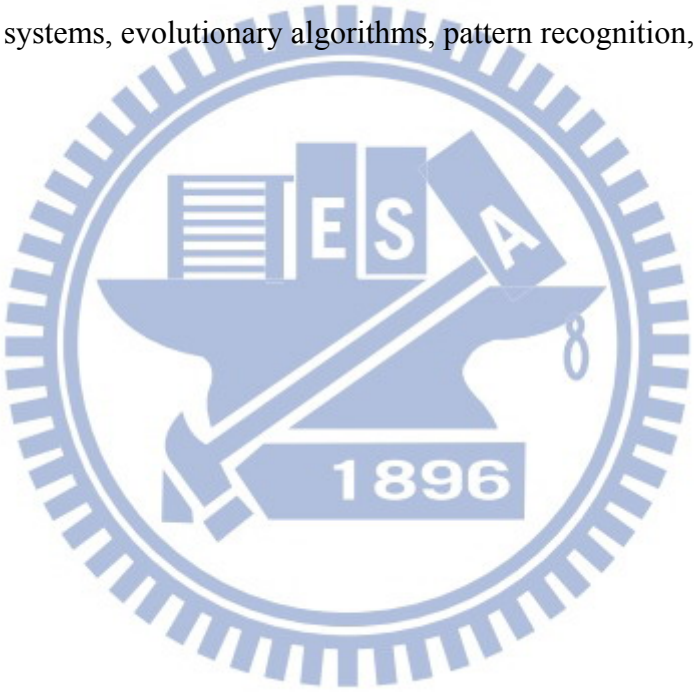
[102] R. S. Crowder, “Predicting the Mackey-Glass time series with cascade correlation learning,” in *Proc. 1990 Connectionist Models Summer School,* D. Touretzky, G. Hinton, and T. Sejnowski, Eds., Carnegie Mellon Univ., pp. 117-123, 1990.

[103] R. Jang, “ANFIS: Adaptive-network-based fuzzy inference system,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.

[104] J. Moody, “Fast learning in multi-resolution hierarchies,” in *Advances in Neural Information Processing Systems I*, D. *S.* Touretzky, Ed. San Mateo, CA Morgan Kaufman, pp. 29-39, 1989.

Vita

Chi-Yao Hsu has received his B.S. degree in department of electrical engineering from National Taiwan Ocean University, Taiwan, in 2001 and his M.S. degree in department of electrical engineering from National Central University, Taiwan, in 2003. He proposed the Ph. D. oral exam at institute of electrical control engineering from the National Chiao Tung University, Taiwan, R.O.C. in May, 2012. His research interests lie in the areas of neural networks, fuzzy systems, evolutionary algorithms, pattern recognition, and computer vision.



Publication List

Accepted Journal Papers:

1. Chi-Yao Hsu, Yung-Chi Hsu, and Sheng-Fuu Lin, "Reinforcement evolutionary learning using data mining algorithm with TSK-type fuzzy controllers," *Applied Soft Computing*, vol. 11, no. 3, pp. 3247-3259, 2011. (SCI/EI)
2. Sheng-Fuu Lin, Chin-Chia Wu, Chi-Yao Hsu, and Dou-Chih Hsu, "An efficient 3D model retrieval based on principal axes analysis and feature integration, " *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 4, pp. 583-604, 2011. (SCI/EI)
3. Chi-Yao Hsu, Yung-Chi Hsu, and Sheng-Fuu Lin, "A hybrid learning neural network based image alignment system using global feature selection approach," *Advances in Computer Science and Engineering*, vol. 6, no. 2, pp.129-157, 2011.
4. Chi-Yao Hsu, Yi-Chang Cheng, and Sheng-Fuu Lin, "Efficient and accurate image alignment using TSK-type neuro-fuzzy network with data-mining based Evolutionary Learning Algorithm" *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 96, pp. 1-22, 2011. (SCI/EI)
5. Yi-Chang Cheng, Sheng-Fuu Lin, and Chi-Yao Hsu, "Q-Value based particle swarm optimization for reinforcement neuro-fuzzy system design," *International Journal on Computer Science and Engineering*, vol. 3, no. 10, pp. 3477-3489, 2011.
6. Chi-Yao Hsu, Yi-Chang Cheng, and Sheng-Fuu Lin, "Precise image alignment using cooperative neural-fuzzy networks with association rule mining based evolutionary learning algorithm," *Optical Engineering*, vol. 51, no. 2, pp. 027006:1-15, 2012. (SCI/EI)
7. Chi-Yao Hsu, Sheng-Fuu Lin, and Jyun-Wei Chang, "Data mining-based hierarchical cooperative coevolutionary algorithm for TSK-type neuro-fuzzy networks design," accepted to appear in *Neural Computing and Applications*, 2012. (SCI/EI)
8. Jyun-Wei Chang, Sheng-Fuu Lin, and Chi-Yao Hsu, "Accurate and rapid alignment of laser scanned 3D surface using TSK-type neural-fuzzy network-based coarse-to-fine strategy," accepted to appear in *Optics and Lasers in Engineering*, 2012. (SCI/EI)

Conference Paper:

1. Chi-Yao Hsu and Sheng-Fuu Lin, "Design of image alignment system using TSK-type neuro-fuzzy network with two-stage data-mining based evolutionary learning algorithm," in *Proc*. *Computer Graphics Workshop 2011*, Taipei, Taiwan, July 2011.