DESIGN AND CONSTRUCTION OF A SMA CONTROLLED ARTIFICIAL FACE

Thomas Kin Fong LEI

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF PHILOSOPHY

in

Department of Mechanical and Automation Engineering

July 2000

DESIGN AND CONSTRUCTION OF A SMA CONTROLLED ARTIFICIAL FACE

ABSTRACT

Emotion expression is the special characteristic of human being and animal. As such, people nowadays may prefer a robot which not only provide help but also equipped with emotional feedback, and hence the motivation of the present project. In this works, we design and construct a human-sized artificial face to mimic human facial expressions. In human face, actuation of the many muscles allows expression of different emotions. We aim here to conduct actuation of our artificial face by shape memory alloy (SMA) wire in substitute of muscles. However, because of its special feature control of the SMA wire is a difficult problem. Here, we attempt two methods, the model-based approach and the neural-fuzzy-based approach. In model-based approach, we identify the model of the SMA wire, based upon which the relationship of displacement and applied current can be computed. The performance of this approach is illustrated in a 1dimension position control experiment of a 2 SMA wires linkage system. The model-based approach, however, becomes too difficult to use when more SMA wires are involved. Hence, a neural-fuzzy-based approach is also pursued, and illustrated in the 2 dimension position control experiments of a 4 SMA wires linkage system. With these control algorithms at hand, we provided to design, construct and control an artificial face. In this project, the hardware artificial face is built using a plastic skull model. There are 6 control points and 3 mechanical structures on the artificial face. By actuating different combination of these control points and mechanical structures, the artificial face is made to express different facial expressions.

設計與建立一個利用形狀記憶金屬線驅動的人工面譜

摘要

表達感情是人類和動物的一種特有的性質，現時的人們不單希望能夠從機械人中得到幫助，更希望能夠與人們有感情的交流，這就是我們這個研究的動機。從這個研究，我們設計並且建立了一個真人大小的人工面譜，它能夠模仿人類面部的表情。人類的面部能夠表達出不同的面部表情是由於眾多的肌肉驅動而達到的，我們在這研究的目標是由形狀記憶金屬線取代了肌肉，我們的人工面譜也可以表達出不同的面部表情。可是，因爲形狀記憶金屬線的特殊性質，控制形狀記憶金屬線是一個困難的控制問題。在這個研究中，我們提出了兩個方法去解決這個問題，分別是模型近似法及類神經快思邏輯近似法。在模型近似法中，我們找出形狀記憶金屬線的數學模型，這樣，便能夠計算到它的收縮長度及提供電流大小的關係，這個模型近似法的效能由兩條形狀記憶金屬線連接的一維位置控制系統中表達出來。但是當我們將更多的形狀記憶金屬線連接在一起時，這個模型近似法變得不準確，因此，我們提出了類神經快思邏輯近似法，這個類神經快思邏輯近似法的效能由四條形狀記憶金屬線連接的二維位置控制系統中表達出來。由以上得知的控制理論，我們可設計、建立並控制這個人工面譜。這個研究中的人工面譜是建立在一個塑膠造的人頭骨上，在面譜上有六個控制點及三個機械結構，從這些控制點及機械結構的不同組合的推動，人工面譜便可以得以表達出不同的面部表情。

ACKNOWLEDGMENTS

I would like to acknowledge all the people who have assisted me during two years of my master study at Chinese University of Hong Kong. I am most grateful to my supervisor, Prof. Yeung Yam, for his professional and personal advice, guidance and help. He has provided me with numerous valuable ideas, insights, encouragement and comments. He has also been very understanding and supportive. I am very fortunate to have him as my supervisor. I am very grateful to Dr. Péter Baranyi for his numerous suggestions, discussions, ideas and personal help. I am also thankful for his cares when I was in Budapest.

My fellow students and colleagues in the Intelligent and Control Systems Laboratory, Dr. Stephen Yang and Mark Wong, have provided help and moral support throughout my stay here. I would like to thank our technical support team, Allan Mok and Philip Lee, for their professional advice.

I would like to thank all the members of my family, especially for my mother. Thanks to Andrea Ho-Asjoe for her support.

CONTENTS

LIST OF FIGURE

CHAPTER ONE

INTRODUCTION

Emotion expression is the special characteristic of human being and animal. In recent years, there has been interested in robot emotion feedback to human considerably. In the Science University of Tokyo, Japan, they have built a female robotic head that can both recognize and express fear, happiness, surprise, sadness, anger and disgust [1]. It was constructed by aluminum structure with 18 air-pressure-driven microactuators. In the Massachusetts Institute of Technology, USA, Research group had developed an interactive robot which can express meaningful social exchanges with humans [2]. Their approach is inspired by the way infants learn to communicate with adults. Specifically, the mode of social interaction is that of a caretaker-infant relationship where a human acts as the caretaker for the robot. We want to make the robot as human-like being as possible nowadays. We prefer a robot that can provide not only help but also emotional feedback.

In this project, the objective is to construct a human-sized artificial face to mimic human facial expressions. Of course, a human face is actuated by muscles to express its many facial expressions. With the absence of muscle in our artificial face, we need other type of actuation to effectuate the expressions. In this regards, many kinds of actuators are possible, such as motors, pneumatics, shape memory alloy (SMA), each of them come with certain advantages and disadvantages. Motors and pneumatics are powerful and easy to control, but they are large, noisy and dirty. SMA wire is clean and bio-friendly, and considering power pre area, the largest among the actuators, but the characteristics of SMA are complicated to express in mathematics. Controlling the SMA is hence a difficult problem. However, since the present project is to construct a human-sized face with clean and quiet moving parts, SMA wire is our preferred choice of actuation.

SMA has been received increasing attention in recent years, especially in the development of innovative engineering systems such as micro-actuators, micro-valves, vibration absorbers, etc. The use of these so called smart materials as actuators in robotic applications attempts to take advantage of their large capacities in motion and force transmission. To effectuate their control, various mathematical models have been proposed in the literatures to describe their nonlinear behavior with hysteresis. Some of these models are based on the analogy with the

phase transformation in the ferroelectric materials as described by the Landau-Devonshire theory [3][4], which is based on state equilibrium consideration set up by a minimized free energy function. Others describe the stress variations as a linear function of strain variations, the temperature and the martensitic fraction variations. Different mathematical expressions of such factors have been applied to approximate the typical stress-strain characteristic and hysteresis of the SMA [5][6][7]. In this work, we adopt the model by K.Ikuta [8][9][10] describing the relationship of stress, strain and temperature. It is shown that upon identifying the various parameters of the Ikuta model, the resulting 1dimensional position control performance of two linking SMA wires can be conducted satisfactorily.

Because of the increase in non-linearity as more SMA wires are involved, model-based approach becomes less effective in yielding reasonable performance. This motivation also our investigation in another approach: the neuro-fuzzy-based approach of controlling SMA. Research in the field of neural networks has attracted increasing attention in recent years. Since 1943, when Warren McCulloch and Walter Pitts [18] presented the first model of artificial neurons, more and more new and sophisticated proposals have been made from decade to decade. Mathematical analysis has solved some of the mysteries pose by the new models but has left many questions open for future investigations. The advantage of neural network is that it can approximate any functions without the need of accurate mathematical models. However, due to the lack of standardized process to determine the number of neurons, neuron-layers and connections, a neural network may contain redundant, weakly contributing components. In this work, we introduce two neuro-fuzzy algorithms. The first have a better approximation properly but the structure of the network is more complicated. The second has less performance in approximation but readily allows application of certain complexity reduction on its structure.

This thesis is organized as follows:

Chapter 2 proposes a model-based approach with parameters readily identified and measured through simple experiments. A one-dimension control experiment will be performed based on this model.

Chapter 3 introduces two neural-fuzzy algorithms for controlling the movement of four linking SMA wires in two-dimensional plane. The experiments yield quite satisfactory performance.

Chapter 4 describes the construction of both software and hardware parts of the artificial face system. Various expressions as implemented on the artificial face are shown.

Chapter 5 concludes what we have done on this thesis and provides some suggestions for future development.

CHAPTER TWO

MODEL-BASED CONTROL OF SMA WIRES

This chapter describes the modeling and model identification of shape memory alloy (SMA) wire [11][12]. Specifically, we take the model by Ikuta [8][9][10] and identifies experimentally the relevant parameters of the SMA wire. The identified single wire model is then extended to a system of two SMA wires joining together at their tips, based upon which open loop position control of the linkage is conducted. Experiments on a two-wires linking system with an overlooking video camera for on-line measurements yield quite satisfactory performance.

2.1 MODEL IDENTIFICATION OF SMA WIRES

Modeling of SMA is subdivided into two principal components, namely, the temperature-current relationship and the strain-temperature relationship, which capture, respectively, the dynamic characteristics and the hysteresis phenomenon of the SMA metal. Together, they give rise to the strain-current relationship that dictates the position control of the linking point in consideration.

2.1.1 Temperature-Current Relationship

Temperature-current relationship is derived by modeling the heat flow in the SMA wire. It is assumed that the temperature is spatially uniform throughout during the process, i.e., temperature gradients within the wire is negligible, and hence one can apply conventional dynamic conduction and heat transfer theory directly [13]. Let

ρ_w = density of wire material [$kg\ m^{-3}$]
c_w = specific heat of wire [$J\ kg^{-1}\ ℃^{-1}$]
V_w = volume of wire [m^3]
A_w = surface area of wire [m^2]

ρ_j = density of the joint material [$kg\ m^{-3}$]

c_j = specific heat of the joint [$J\ kg^{-1}\ ℃^{-1}$]

V_j = volume of the joint [m^3]

A_j = surface area of the joint [m^2]

T = temperature [℃]

t = time [s]

i = electrical current [A]

R = electrical resistance of wire [Ω]

h = convection heat transfer coefficient [$W\ m^{-2}\ ℃^{-1}$]

T_{air} = ambient temperature [℃]

The heat transfer equation for a length of SMA wire mounted on conducting joints is then

$$\rho_w c_w V_w \frac{dT(t)}{dt} + \rho_j c_j V_j \frac{dT(t)}{dt} = Ri^2(t) - hA_w(T(t) - T_{air}) - hA_j(T(t) - T_{air}). \quad (2.1)$$

The left hand side of (2.1) amounts to the rate of change of thermal energy in the SMA wire, which should equal to the rate of heat produced by the electrical current, plus the rate of heat lost to the surrounding air from the surfaces of the wire and the joints. Heat lost due to radiation is assumed negligible. With a constant current $i(t)=i_c$, the temperature at steady state can be expressed as

$$T_{ss} = \frac{Ri_c^2}{hA_w + hA_j} + T_{air}. \quad (2.2)$$

Figure 2.1 compares the steady state temperature/current relationship (T_{ss} vs i_c) obtained from experiments and from (2.2). A value of $1.15 \times 10^{-3} m^2$ is determined for A_j to yield a good match of the two. The value of V_j is estimated by comparing the measured transient of the temperature response with the predictions from (2.1). Other parameters are obtained as follows: ρ_w and c_w from the factory provided data sheet for the FLEXINOL-LT wire, A_w and V_w from the geometry of the wire, and R is measured directly as the resistance of the SMA wire. Any slight changes in these values which might be caused by temperature variations are neglected. The joints are made of iron, the values ρ_j and c_j are obtained from the corresponding table in [13]. And finally, h and T_{air} are measured from the environment.

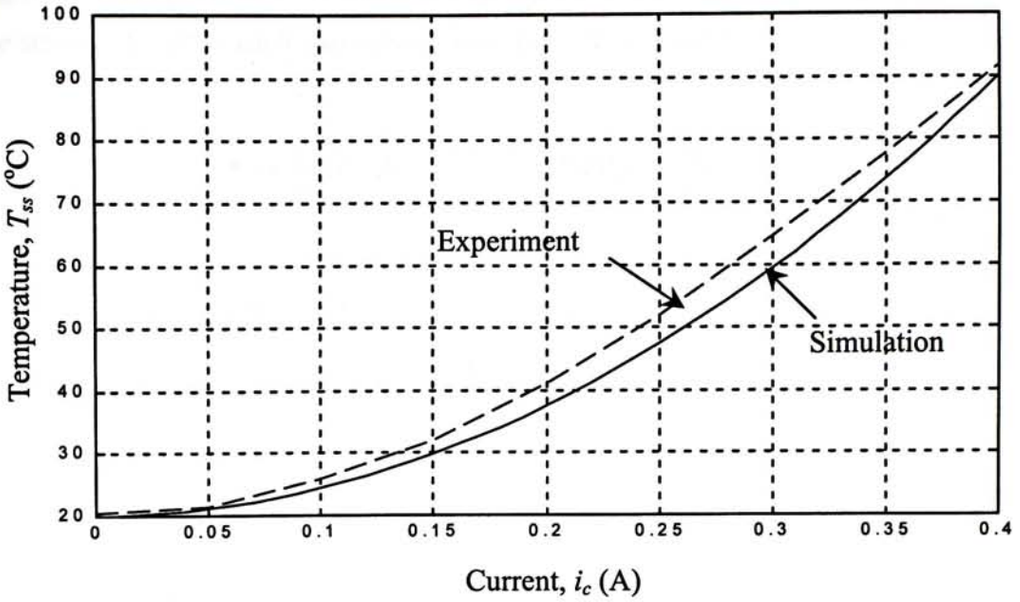


Figure 2.1. Steady state temperature-current diagram for a 100μm FLEXINOL-LT wire: Simulation (solid line) and Experiment (dotted line). (ρ_w = 6450kgm^{-3}, c_w = 320Jkg^{-1} ℃$^{-1}$, ρ_j = 7870kgm^{-3}, c_j = 447Jkg^{-1} ℃$^{-1}$, A_j = 1.15×10^{-3}m^2, V_j = 2×10^{-8}m^3, R=40Ω, h=70Wm^{-2} ℃$^{-1}$, T_{air}=20 ℃)

2.1.2 Stress-Strain Relationship

Here, we consider the SMA material in the biphased state of austenite phase (high temperature) or martensite phase (low temperature) only. The R-phase will not be considered. Let

ϵ = strain of wire

ϵ_A = strain in the austenite phase

ϵ_M = strain in the martensite phase

V_A = volume in the austenite phase [m^3]

V_M = volume in the martensite phase [m^3]

D_A = young modulus in the austenite phase [MPa]

D_M = young modulus in the martensite phase [MPa]

σ = applied stress [MPa]

R = martensite fraction [%]

Using the model proposed by K. Ikuta [8][9][10], the strain ϵ can be computed as the average of the strain value ϵ^* in each unit volume over the total volume V of the material, i.e.,

$$\varepsilon = <\varepsilon^*> = \frac{1}{V}\int_V \varepsilon^* dV = \frac{V_A}{V}\int_{V_A} \varepsilon_A^* dV_A + \frac{V_M}{V}\int_{V_M} \varepsilon_M^* dV_M$$

where ϵ_A^* and ϵ_M^* are the unit volume strain values corresponding to the austenite and martensite phases, respectively. With $R = \frac{V_M}{V}$,

$$\varepsilon = (1-R)<\varepsilon_A^*> + R<\varepsilon_M^*>$$

When the material is in a full austenite phase, R=0 and $\epsilon = <\epsilon_A^*>$. When it is in full martensite phase, R=1 and $\epsilon = <\epsilon_M^*>$. Similarly, the stress σ in the wire can be expressed as:

$$\sigma = <\sigma^* + \sigma_r> = <\sigma^*> + <\sigma_r>$$

where σ^* is the applied stress on an elementary volume, and σ_r represents the stress field to ensure cohesion of the elementary volumes. Since this field is self-equilibrate, $<\sigma_r> = 0$, and so $\sigma = <\sigma^*>$. By the Saint-Venant principle, one then has $\sigma = \sigma^*$.

In the austenite phase, behavior is elastic and one has:

$$\varepsilon_A = \frac{\sigma}{D_A} \tag{2.3}$$

which is, simply, the Hooke's law. In the martensite phase, behavior is elastoplastic and one has:

$$\varepsilon_M = \frac{\sigma}{D_M} + \alpha(\sigma - \sigma_1) + \varepsilon_r \tag{2.4}$$

where α and ϵ_r account for the transformation plasticity. They vary depending on the previous values of the applied stress σ_1 as follows:

If $\sigma_{-1} \leq \sigma$ and if

$$0 \leq \sigma \leq \sigma_1 \qquad \alpha = 0; \quad \varepsilon_r = 0 \tag{2.5}$$

$$\sigma_1 \leq \sigma \leq \sigma_2 \qquad \alpha = \frac{\varepsilon_2 - \varepsilon_1}{\sigma_2 - \sigma_1}; \quad \varepsilon_r = 0 \tag{2.6}$$

$$\sigma_2 \leq \sigma \qquad \alpha = 0; \quad \varepsilon_r = \varepsilon_2 - \varepsilon_1 \tag{2.7}$$

else

$$\alpha = 0; \quad \varepsilon_r = \varepsilon_{-1} - \frac{\sigma_{-1}}{D_M} \tag{2.8}$$

Equations (2.5)-(2.8) allow us to determine the values σ_1, σ_2, ϵ_1 and ϵ_2 for the FLEXINOL-LT wire utilized in our setup, by comparing the simulated stress-strain plots with the experimentally determined ones. Figures 2.2 and 2.3 show the results for the temperatures 20°C and 25°C, respectively. The parameters σ_1, σ_2, ϵ_1 and ϵ_2 are obtained to yield a good match between the simulated and experimental data. The other parameters of D_A , D_M , A_s , A_f, M_s and M_f are read off directly from factory supplied specifications.

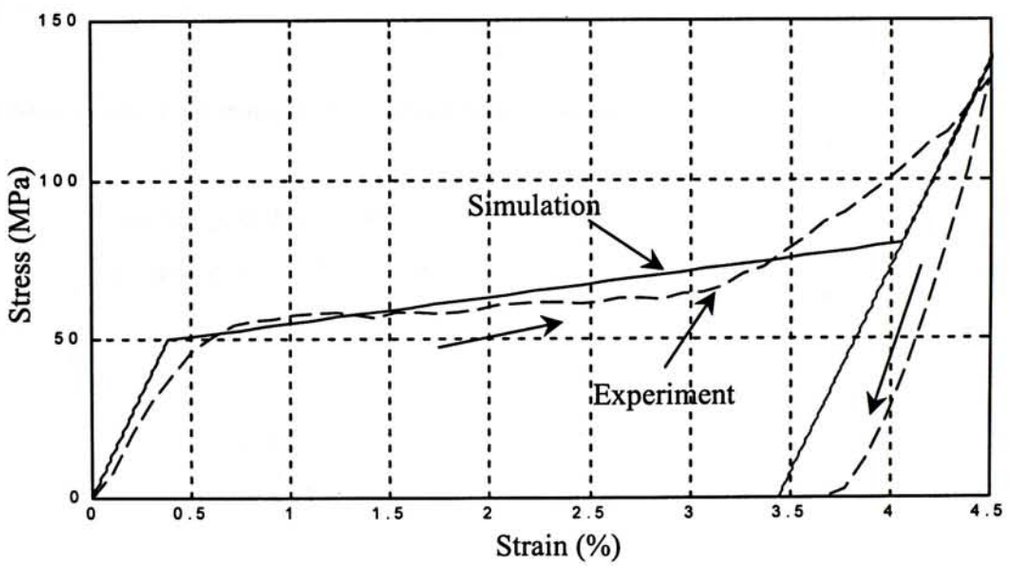


Figure 2.2. Stress-strain diagram for 100μm FLEXINOL-LT wire at 20°C: Simulation (solid line) and Experiment (dotted line). (D_A=50GPa, D_M=13GPa, A_s=28 ℃, A_f=32 ℃, M_s=26 ℃, M_f=21 ℃, σ_1=50MPa, σ_2=80MPa, ϵ_r=3.45%)

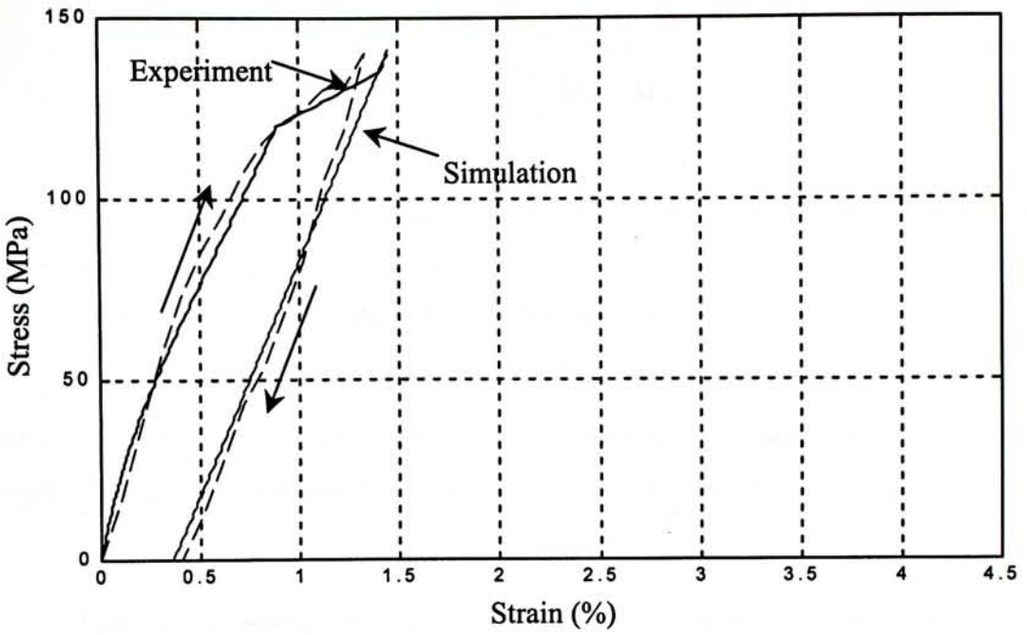


Figure 2.3. Stress-strain diagram for the 100μm FLEXINOL-LT wire at 25°C: Simulation (solid line) and Experiment (dotted line). (D_A=50GPa, D_M=13GPa, σ_1=120MPa, σ_2=135MPa, ϵ_r=0.40%)

2.1.3 Martensite Fraction-Temperature Relationship

This is the relationship that gives rise to the hysteresis effect. Let Ms, M_f, A_s, and A_f be the start and finish temperatures of martensite and austenite phase, respectively. The martensite fraction $R \in [0, 1]$ as a function of stress and temperature can then be modeled as [8]:

$$R(T,\sigma) = \frac{R_1}{1+\exp K(T - c\sigma - T_m)} + R_2 \tag{2.9}$$

where c is the Clausius-Clapeyron constant, and K and T_m are constants depending on the direction of transformation:

- for transformation $(A \rightarrow M)$, $(T\text{-}c\sigma)$ increases :

$$T_m = \frac{A_s + A_f}{2} \qquad K = \frac{4.4}{A_f - A_s}$$

- for reverse transformation $(M \rightarrow A)$, $(T\text{-}c\sigma)$ decreases :

$$T_m = \frac{M_s + M_f}{2} \qquad K = \frac{4.4}{M_s - M_f}$$

Moreover, the variables R_1 and R_2 take on values according to the change of direction of the transformation:

- when $(M \rightarrow A)$ reverses to $(A \rightarrow M)$: $R_1 = (1 - R_{-1})$ and $R_2 = R_{-1}$
- when $(A \rightarrow M)$ reverses to $(M \rightarrow A)$: $R_1 = R_{-1}$ and $R_2 = 0$

where R_{-1} denotes the value of R before the transformation reverses direction. Figure 2.4 depicts two simulation examples of the hysteresis effect for the 100μm FLEXINOL-LT wire.

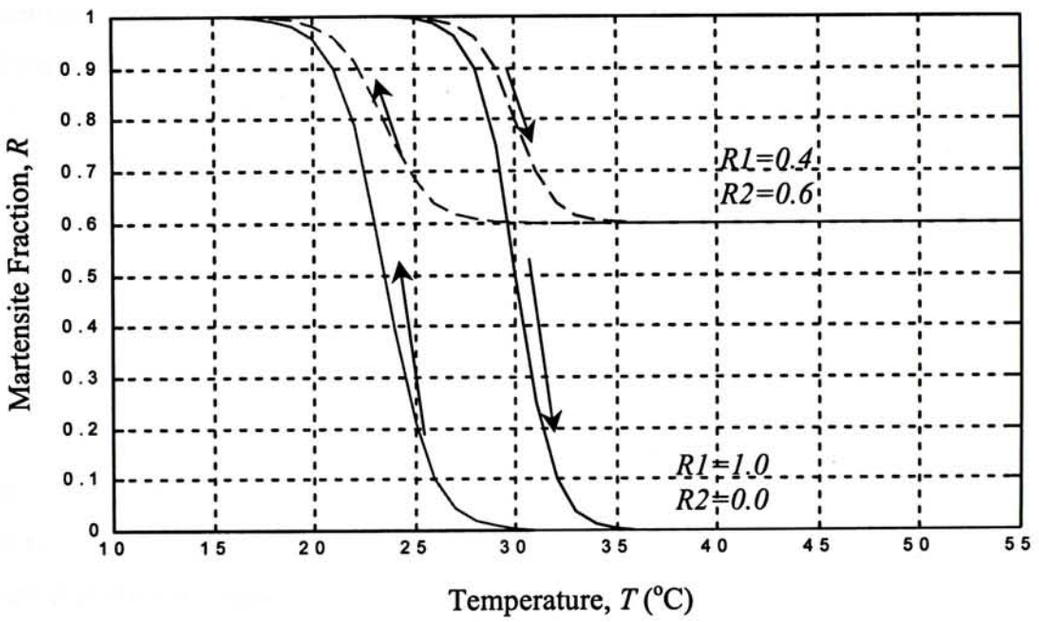


Figure 2.4. Simulated martensite fraction-temperature relationships for the 100μm FLEXINOL-LT wire

2.2 MODEL-BASED POSITION CONTROL OF TWO LINKING SMA WIRES

Figure 2.5 depicts the position control experimental setup consisting of two 100μm FLEXINOL LT wires. Each wire has one end connected to a stationary point, and the other end joining together and is electrically ground. The wires are designated as wire0 and wire1 as shown. The objective of the experiment is to control the linear movement of the linking point by changing the lengths of SMA wires through varying their temperatures. This, in turn, is achieved by passing currents through the wires and adjusting the current power

dissipation by a pulse width modulator. The system is equipped with an overlooking video system with CCD camera for on-line measurements.

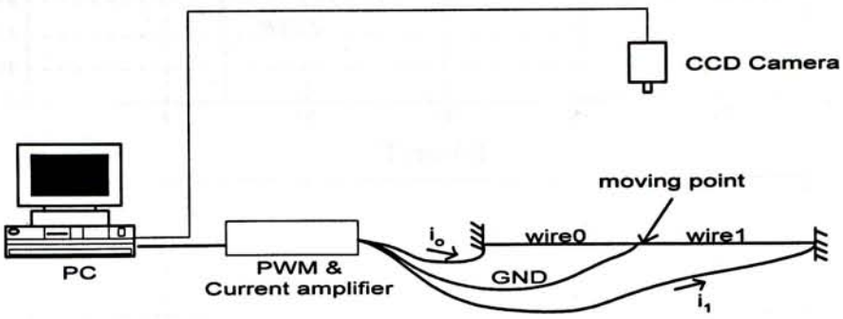


Figure 2.5. Position control of a two SMA wires linkage system.

Assuming that the wires are always in tension during the process, it can be stated that at equilibrium the total strains of the two wires must be constant and that the stress in the wires must be equal to each other. That is:

$$\begin{cases} \varepsilon_{wire\,0} = (1 - R_{wire\,0})\varepsilon_{A,wire\,0} + R_{wire\,0}\varepsilon_{M,wire\,0} \\ \varepsilon_{wire\,1} = (1 - R_{wire\,1})\varepsilon_{A,wire\,1} + R_{wire\,1}\varepsilon_{M,wire\,1} \\ \varepsilon_{total} = \varepsilon_{wire\,0} + \varepsilon_{wire\,1} \\ \sigma_{wire\,0} = \sigma_{wire\,1} \end{cases} \tag{2.10}$$

Here, ϵ_{total} is a constant and can be measured from the experimental setup. Then, using (2.10) with (2.3), (2.4), and (2.9), one can solve for ϵ_{wire0}, i.e., the position of the linkage point, as a function of the wire currents.

Figures 2.6-2.8 present the results of the above computation. Figure 2.6 shows the input currents to wire0 and wire1, Figure 2.7 shows the corresponding simulated temperature responses of the wires, and Figure 2.8 depicts the resulting strain ϵ_{wire0}. Experimental measurements of ϵ_{wire0} are also included in Figure 2.8 for comparison. The simulated and experimental curves compare reasonably well, indicating the performance of the model-based approach to position control.

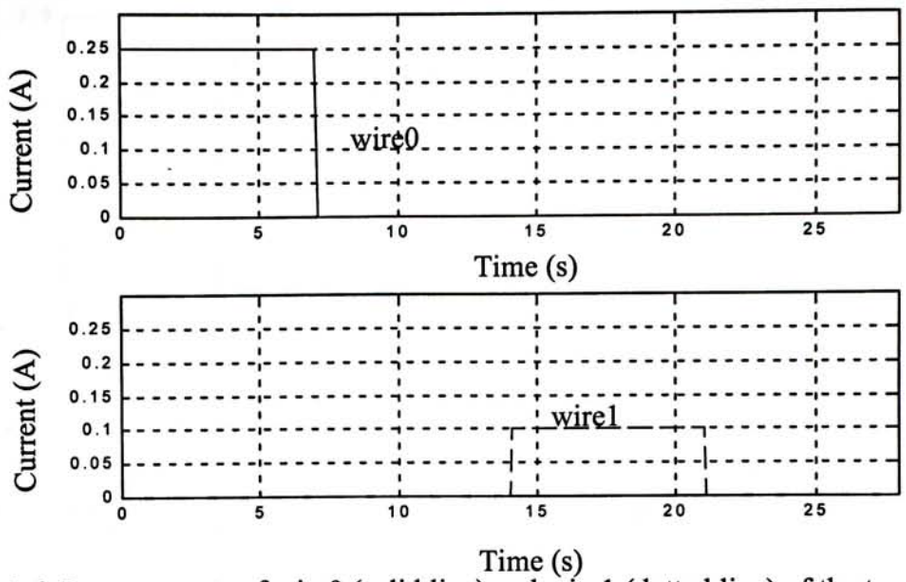


Figure 2.6. Input currents of wire0 (solid line) and wire1 (dotted line) of the two SMA wires linkage system.

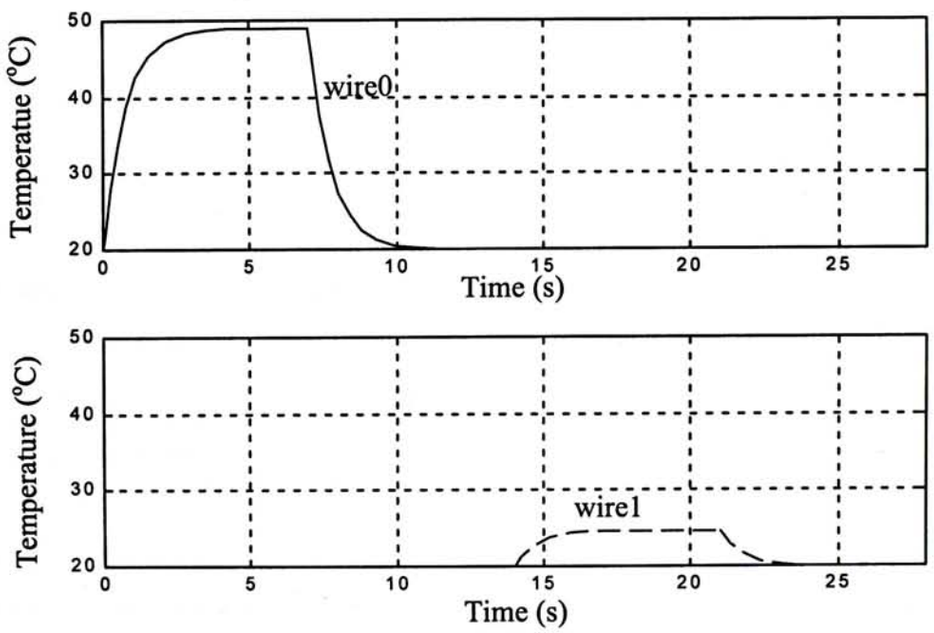


Figure 2.7. Simulated temperature response of wire0 (solid line) and wire1 (dotted line) subject to the input currents of figure 2.6.

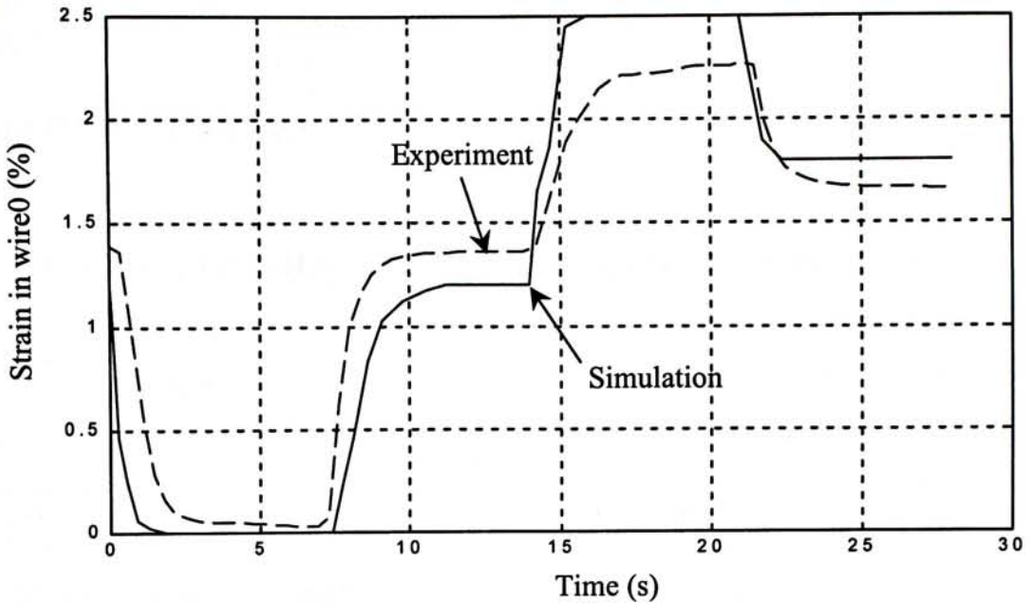


Figure 2.8. Strain ϵ_{wire0} of the two-SMA wires linkage system subject to the input currents of figure 2.6: Simulation (solid line) and Experimental (dotted line) result.

2.3 SUMMARY

Shape Memory Alloy (SMA) has generated a lot of expectation in recent years for potential use as actuators and sensors. The outstanding problem, however, remains in an accurate modeling and subsequent control of its performance. In this chapter, we reported the model-based approach to the position control of a two SMA wires linking system, whereby an existing model is adopted for the FLEXINOL-LT wires utilized in our setup. Necessary model parameters are then identified experimentally. Under equilibrium conditions, relationship between the currents to the wires and the ensuing position of the linking point can be computed. This approach yields acceptable results when applied to a two SMA wire case but becomes intractable when additional wire is included. The next chapter utilized two neuro-fuzzy approaches. These approaches are applied to a four SMA wires experiment with reasonable good performance.

CHAPTER THREE

NEURAL-FUZZY-BASED CONTROL OF SMA WIRES

In the last chapter, we identified the single SMA wire model and described the movement of two linking SMA wires in one-dimension. Because of the high non-linearity of the SMA wire, however, the model-based approach becomes less accurate when more SMA wires are involved. Moreover, the geometry of the artificial face and the connection of the SMA wires on the artificial face are complicated. These additional non-linear effects tend to render the model-based approach less effective in coming up with reasonable performance. Hence, we here pursue the approach to control the system using neural-fuzzy based methods. Particularly, we introduce two neural-fuzzy algorithms of different structures. Their performance will be illustrated in a controlling experiment of four linking SMA wires moving in two-dimensional plane.

3.1 ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM (ANFIS)

In this section, we introduce a neural network that is functionally equivalent to fuzzy inference system. The proposed architecture is referred to as adaptive neuro-fuzzy inference system (ANFIS) [14]. Figure 3.1 illustrates the corresponding equivalent ANFIS architecture.

3.1.1 ANFIS Architecture

The ANFIS structure can be divided into four principal components much like a fuzzy system: fuzzification interface, fuzzy rule base, fuzzy inference and defuzzification interface. Figure 3.1(a) depicts the architecture of the ANFIS for the present work. For simplicity, here we use a system with two inputs x_1 and x_2 and one output y for illustration. Each input is assumed to have three membership functions, with fuzzy if-then rules in the following form:

Rule 1: If x_1 is A_1 and x_2 is B_1, then $f_1 = p_1x_1+q_1x_2+r_1$,

Rule 2: If x_1 is A_1 and x_2 is B_2, then $f_2 = p_2x_1+q_2x_2+r_2$,

...

Figure 3.1(b) depicts the partition of the two-dimensional input space into nine overlapping fuzzy regions, each of which is governed by a fuzzy if-then rule. The following describes the ANFIS architecture in Figure 3.1(a) layer by layer to mark its functional equivalency to the fuzzy system of Figure 3.1(b). Denoting the output as y_i^l, $i=1..n_l$ in layer l, where n_l is the number of neurons, one has:

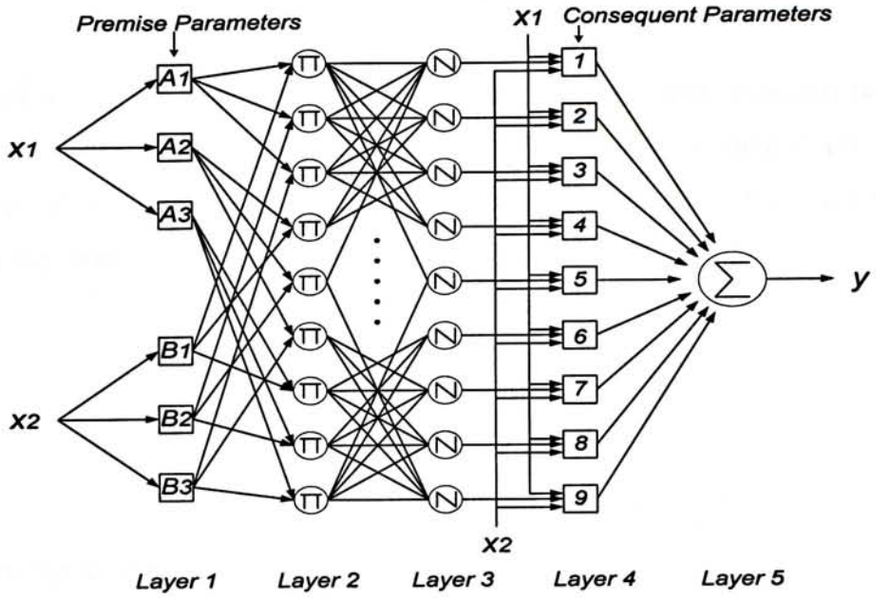


Figure 3.1(a). ANFIS architecture for a two-input fuzzy model with nine rules

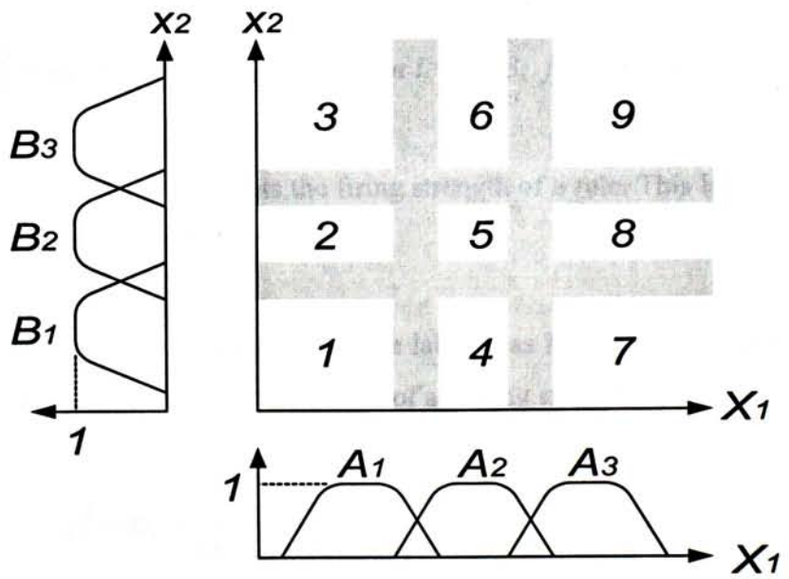


Figure 3.1(b). The input space that are partitioned into nine fuzzy regions

Layer 1 This layer acts as fuzzification interface. Every node i in this layer is an adaptive node with an associative node function:

$$y_i^1 = \mu_{Ai}(x_1) \qquad \text{for } i = 1,2,3, \text{ or}$$

$$y_i^1 = \mu_{Bi-3}(x_2) \qquad \text{for } i = 4,5,6$$

where x_1 (or x_2) is the input to node i and A_i (or B_{i-3}) is a node function (membership function). In effect, the membership function A_i (or B_{i-3}) maps the input x_1 (or x_2) into the membership value μ_{Ai} (or μ_{Bi-3}) as output. The present work adopt Gaussian bell-shaped function of the form

$$\mu A(x) = \frac{1}{1 + \left| \frac{x - ci}{ai} \right|^{2bi}}$$

for membership function A_i (or B_{i-3}), where $\{a_i, b_i, c_i\}$ is the parameter set to be determined by data fitting. These parameters are referred to as premise parameters.

Layer 2 This layer contains identical nodes labeled as Π, with output set to be the product of all the incoming signals:

$$y_k^2 = \omega_k = \mu_{Ai}(x_1)\mu_{Bj}(x_2) \qquad \text{for } i = 1,2,3;\ j = 1,2,3;\ k = 1,2...9$$

Physically, the node output represents the firing strength of a rule. This layer is equivalent to a fuzzy AND operator.

Layer 3 The layer also contains identical node labeled as N. The node i here calculates the ratio of the ith rule's firing strength to the sum of all firing strengths:

$$y_i^3 = \varpi_i = \frac{\omega_i}{\omega_1 + \omega_2 + ... + \omega_9} \qquad \text{for } i = 1,2...9$$

Outputs of this layer are termed normalized firing strengths.

Layer 4 Every node i in this layer is an adaptive node with a node function.

$$y_i^4 = \varpi_i f_i = \varpi_i (p_i x_1 + q_i x_2 + r_i) \qquad \text{for } i = 1,2...9$$

where input ϖ_i is the normalized firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is another parameter set to be determined by data fitting. These parameters are referred to as consequent parameters.

Layer 5 This layer contains a single fixed node, labeled as Σ. Output of this layer, the overall output of ANFIS, is the summation of all inputs to the node:

$$\text{Overall output} = y = y_1^5 = \sum_i \varpi_i f_i = \frac{\Sigma_i \omega_i f_i}{\Sigma_i \omega_i}$$

This layer hence acts as a defuzzification interface.

Altogether, the 5 layers produce an adaptive neural network that is functionally equivalent to a product-sum-gravity-based fuzzy inference system with A_i and B_{i-3} as membership functions.

3.1.2 Hybrid Learning Algorithm

The ANFIS architecture can be trained to approximate any nonlinear function on a compact set [14]. In this work we adopt a particularly effective technique called Hybrid learning algorithm to train the premise and consequent parameters for our application. The learning algorithm is divided into two passes: forward pass and backward pass. In the forward pass, nodal operations are carried out to until layer 4 whereby the consequent parameters are identified by least-squares method. Then, the error signals are propagated backward whereby the premise parameters are updated by gradient descent method. Figure 3.2 summarizes the activities in each pass.

	Forward pass	Backward pass
Premise parameters	Fixed	Gradient descent
Consequent parameters	Least-squares estimator	Fixed
Signals	Node outputs	Error signals

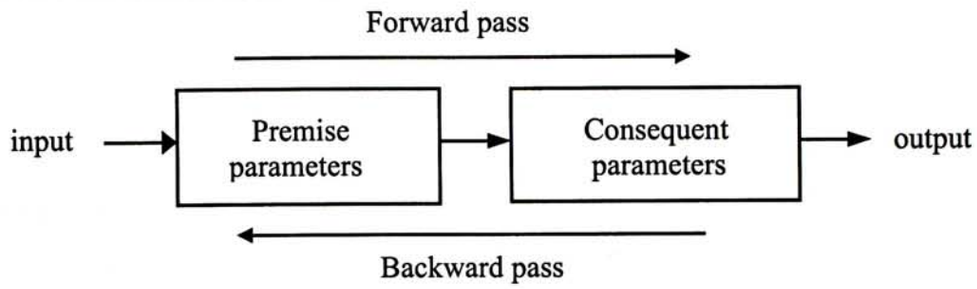


Figure 3.2. Two passes in the hybrid learning procedure for ANFIS.

Now we can combine gradient descent and the least-squares estimator to update the parameters in an adaptive network. For the hybrid learning algorithm, each epoch is composed of a forward pass and a backward pass. In the forward pass, we observe that it is a summation of all incoming signals in layer 5. If the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters.

$$y = \varpi_1(p_1x_1 + q_1x_2 + r_1) + \varpi_2(p_2x_1 + q_2x_2 + r_2) + \ldots + \varpi_9(p_9x_1 + q_9x_2 + r_9) \quad (3.1)$$
$$= (\varpi_1x_1)p_1 + (\varpi_1x_2)q_1 + (\varpi_1)r_1 + \ldots + (\varpi_9x_1)p_9 + (\varpi_9x_2)q_9 + (\varpi_9)r_9$$

We can rewrite equation (3.1):

$$y = \theta_1 f_1(\mathbf{u}) + \theta_2 f_2(\mathbf{u}) + \ldots + \theta_{27} f_{27}(\mathbf{u}) \quad (3.2)$$

where $\mathbf{u} = [x_1, x_2]^T$ is the input vector, $f_1, f_2 \ldots f_{27}$ are known functions of $\mathbf{u}$ if the premise parameters are fixed, and $\theta_1, \theta_2 \ldots \theta_{27}$ are unknown consequent parameter to be estimated. To find out the unknown parameter, we do the experiments to obtain the training data set composed of data pairs $\{(u_i; y_i), i = 1,2...N\}$. They represent desired input-output pairs of the target system to be modeled. Substituting each data pair into equation (3.2) yields a set of N linear equations:

$$\begin{cases} f_1(\mathbf{u_1})\theta_1 + f_2(\mathbf{u_1})\theta_2 + \ldots + f_{27}(\mathbf{u_1})\theta_{27} = y_1, \\ f_1(\mathbf{u_2})\theta_1 + f_2(\mathbf{u_2})\theta_2 + \ldots + f_{27}(\mathbf{u_2})\theta_{27} = y_2, \\ \ldots \\ f_1(\mathbf{u_N})\theta_1 + f_2(\mathbf{u_N})\theta_2 + \ldots + f_{27}(\mathbf{u_N})\theta_{27} = y_N \end{cases} \tag{3.3}$$

We can rewrite equation (3.3) into matrix form:

$$\mathbf{A\theta} = \mathbf{y}$$

where $\mathbf{A}$ is an $N \times 27$ matrix

$$\mathbf{A} = \begin{bmatrix} f_1(\mathbf{u_1}) \ldots f_{27}(\mathbf{u_1}) \\ \vdots \quad \vdots \quad \vdots \\ f_1(\mathbf{u_N}) \cdots f_{27}(\mathbf{u_N}) \end{bmatrix}$$

θ is a 27×1 unknown consequent parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{27} \end{bmatrix}$$

and $\mathbf{z}$ is an $N \times 1$ output vector:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Obviously, this is a standard linear least-squares problem, and the best solution for $\boldsymbol{\theta}$, which minimizes $\|\mathbf{A\theta} - \mathbf{y}\|^2$, is the least-squares estimator (LSE) $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$ and $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the pseudoinverse of $\mathbf{A}$ if $\mathbf{A}^T\mathbf{A}$ is nonsingular. The unknown consequent parameters can be identified optimally under the

condition that the premise parameters are fixed. After the consequent parameters are identified, we can compute the error measure for each training data pair. In the backward pass, the premise parameters are updated by backpropagation method. We can define the error measure for the nth ($1 \le n \le N$) entry of training data as the sum of squared errors:

$$E_n = (T_n - y_n)^2 \tag{3.4}$$

where T_n is the target output and y_n is the actual output of the nth input of training patterns. Hence, the overall error measure is the summation of equation (3.4):

$$E = \sum_{n=1}^{N} E_n$$

In order to develop a learning procedure that implements gradient descent in E over the parameter space, first we have to calculate the error rate $\partial E_p / \partial y$ for the nth training data and for each node output y. The error rate for the output layer, layer 5, can be calculated readily from equation (3.4):

$$\frac{\partial E_n}{\partial y_{1,n}^5} = -2(T_n - y_n) \tag{3.5}$$

For the internal node (k, i), the error rate can be derived by the chain rule:

$$\frac{\partial E_n}{\partial y_{i,n}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_n}{\partial y_{m,n}^{k+1}} \frac{\partial y_{m,n}^{k+1}}{\partial y_{i,n}^k} \tag{3.6}$$

where $1 \le k \le 4$. That is, the error rate of an internal node can be expressed as a linear combination of the error rates of the nodes in the next layer. Therefore for all $1 \le k \le 4$ and $1 \le i \le \#(k)$, we can find $\partial E_n / \partial y_{i,n}^k$ by equation (3.5) and (3.6).

Now if α (a_i, b_i or c_i) is a parameter which we want to find, we have

$$\frac{\partial E_n}{\partial \alpha} = \sum_{y^* \in S} \frac{\partial E_n}{\partial y^*} \frac{\partial y^*}{\partial \alpha}$$

where S is the set of nodes whose outputs depend on α. Then the derivative of the overall error measure E with respect to α is

$$\frac{\partial E}{\partial \alpha} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial \alpha}$$

Accordingly, the update formula for the generic parameter α is

$$\Delta \alpha = -\eta \frac{\partial E}{\partial \alpha}$$

where η is a learning rate. At the end of the backward pass for all training data, the premise parameters can be identified.

With Hybrid Learning Algorithm, ANFIS can be trained for any function [15].

3.2 GENERALIZED NEURAL NETWORK (GNN)

Because of the complexity of ANFIS, the generalized neural network (GNN) is presented. We need a simpler structure and reasonable performance network to be the control algorithm to implement our artificial face. Singular value-based complexity reduction [16][17] can apply to the GNN algorithm. The reduced GNN algorithm is capable of filtering out weakly contributing weighting connections. This section introduces the GNN architecture and its complexity reduction method.

3.2.1 GNN Architecture

This paragraph defines a generalized type neural network chosen from [18][19].

Let us focus on two neighbour layers l and l+1 of a forward model. Let the neurons be denoted as N_i^l, $i = 1..n_l$ in layer l, where n_l is the number of neurons. Further, let input values of N_i^l be $x_{i,k}^l$, $k = 1..n_{l-1}$ and its output y_i^l. The connection between two layers (l

and l+1) can be defined by matrix, which consists of weighting functions $f_{j,i}^{l}(y_i^l)$, where $j=1..n_{l+1}$. Thus

$$x_{j,i}^{l+1} = f_{j,i}^{l}(y_i^l) \tag{3.7}$$

Neurons apply sum operation to input values, but have no transfer function. For instance, the output of N_j^{l+1} is:

$$y_j^{l+1} = \sum_{i=1}^{n_l} x_{j,i}^{l+1}$$

Therefore, from (3.7) neuron N_j^{l+1} yields

$$y_j^{l+1} = \sum_{i=1}^{n_l} f_{j,i}^{l}(y_i^l) \tag{3.8}$$

We use a system with three inputs x_1, x_2, x_3 and two outputs y_1, y_2 for illustration. Here, $n_l=3$, $n_{l+1}=2$, $i=1..3$ and $j=1..2$. Figure 3.3 shows the architecture of three inputs and two outputs generalized neural network.

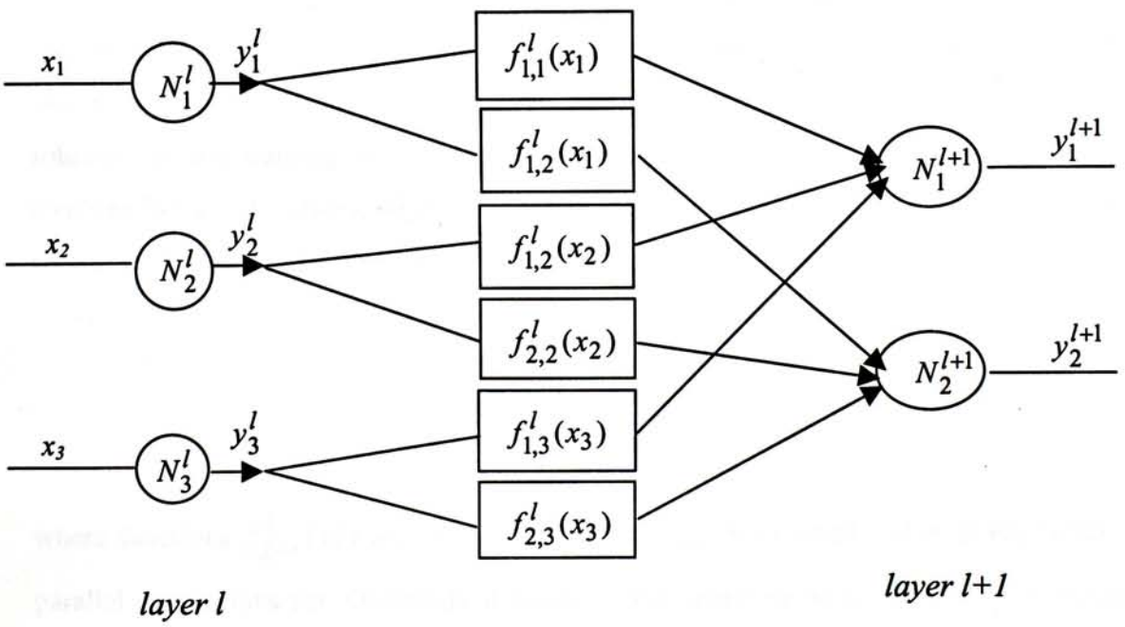


Figure 3.3. Generalized neural network of three inputs and two outputs case.

If the same type of weighting functions are connected to neurons N_i^l as:

$$f_{j,i}^l(y_i^l) = w_{j,i}^l f(y_i^l)$$

then the standard neural network is obtained where the connections are weighted by constant values $w_{j,i}^l$ and the neurons have transfer functions $y = f(x)$. An extended version of the standard network contains neurons that may have different types of functions. This can also be defined as a special case of the generalized type network, where the transfer functions of the neurons are transformed into the weighting functions of the connections as

$$f_{j,i}^l(y_i^l) = w_{j,i}^l f_i^l(y_i^l)$$

The introduced generalized neural network has advantageous and also flexible approximation properties over the standard type [18][19].

3.2.2 Approximation of the GNN

In spite of the mentioned advantageous approximation property, having various types of weighting function in the generalized network needs considerable computational effort in contrast to the standard type. Furthermore, the use of the generalized type is strongly restricted by the fact that its training, namely, the searching of the unknown weighting functions may lead to a complicated or unsolvable mathematical problem. One natural solution of the training is to replace the unknown weighting functions with linearly combined known functions, where only the linear combination must be trained. It practically means that some parallel layer is added to the output. From (3.8):

$$y_j^{l+1} = \sum_{t=1}^{m} \sum_{i=1}^{n_l} f_{j,i,t}^l(y_i^l) b_{j,i,t}^l$$

where functions $f_{j,i,t}^l(y_i^l)$ are known and values $b_{j,i,t}^l$ are trained and m is the number of parallel connections net. Obviously it results in the approximation of the original network. The key idea of this technique comes from neuro-fuzzy algorithms. Let us have a brief introduction, in order to have an easy understanding. Neuro-fuzzy algorithms have recently

emerged as a new topic of both the fuzzy and the neural network theory. In these techniques, the weighting functions in the generalized form are approximated by a one variable fuzzy logic algorithm. In order to save the calculation effort let us apply the widely adopted product-sum-gravity method [20] that leads to the simple implementation of the generalized network even in case of training. The product-sum-gravity method is referred to appendix 1. In order to approximate the contribution of each neuron to the neurons of the next layer by the PSG technique, (3.8) can be written in (3.9). Let us define a more general form, where all antecedent universes may have different number of antecedent sets:

$$y_j^{l+1} = \sum_{i=1}^{n_l} f_{j,i}^l(y_i^l) = \sum_{i=1}^{n_l}\sum_{t=1}^{m_i^l} \mu_{A_{i,t}^l}(y_i^l) b_{j,i,t}^l \tag{3.9}$$

where m_i^l is the number of antecedent sets on universe connected to the output of i-th neuron in layer l. Figure 3.4 depicts a neuro-fuzzy network that represents (3.9). We use a system with three inputs x_1, x_2, x_3 and two outputs y_1, y_2 for illustration. Here, $n_l = 3$, $n_{l+1} = 2$, $i = 1..3$, $j = 1..2$ and $m_1^l = m_2^l = m_3^l = 4$.

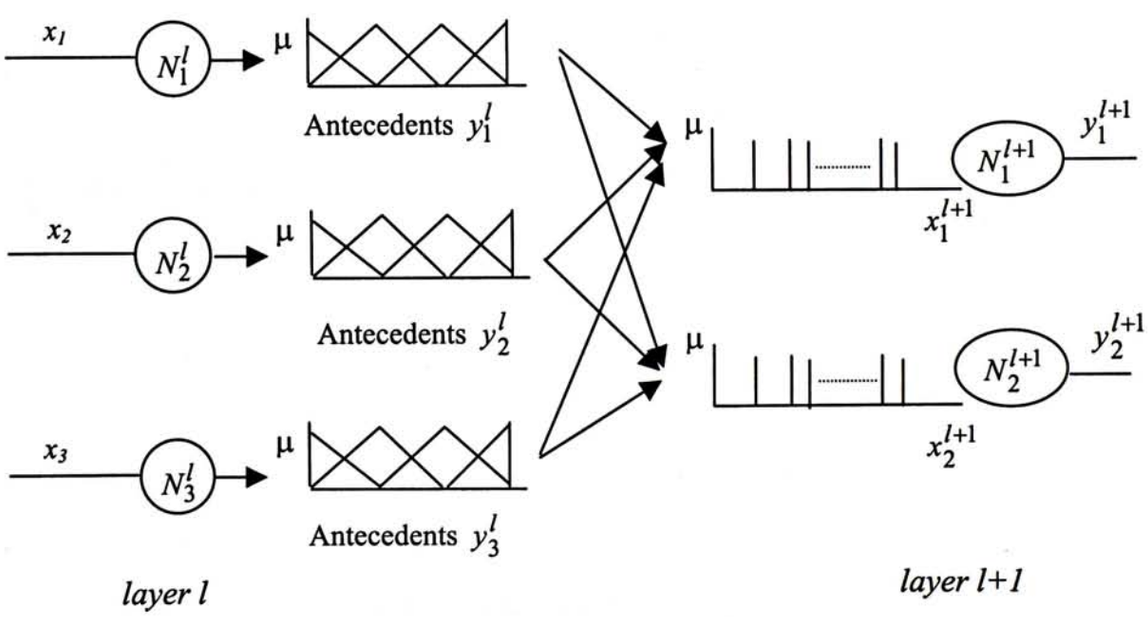


Figure 3.4. Approximation of the GNN

As a matter of fact, if $\forall i : m_i^l = m^l$ then this algorithm is the same as summing up the outputs of m generalized network connected parallel:

$$y_j^{l+1} = \sum_{i=1}^{n_l}\sum_{t=1}^{m_i^l} \mu_{A_{i,t}^l}(y_i^l) b_{j,i,t}^l = \sum_{t=1}^{m^l}\sum_{i=1}^{n_l} \mu_{A_{i,t}^l}(y_i^l) b_{j,i,t}^l = \sum_{t=1}^{m^l}\sum_{i=1}^{n_l} f_{j,i}^l(y_i^l)$$

The benefit is that the arbitrary type weighting functions, which are unknown before the training, are replaced with rather simple known functions and only their linear weighting must be trained.

3.2.3 Backpropagation Training Algorithm

In order to complete the generalized network algorithm let us briefly discuss the key step of a simple training method, that does not tune the antecedent sets, but the position of the consequents, namely values $b_{j,i,t}^l$. Let the k-th training pattern contain input values $x_i^l(k)$ and the desired output value $d_j^{l+1}(k)$. Based on the key idea of Least Square Method [21] used in backpropagation training algorithm, let the error criteria of the algorithm be the square instantaneous error as:

$$\left(\varepsilon_j^{l+1}(k)\right)^2 = \left(d_j^{l+1}(k) - y_j^{l+1}(k)\right)^2 = \left(d_j^{l+1}(k) - \sum_{i=1}^{n_l}\sum_{t=1}^{m_i^l} \mu_{A_{i,t}^l}(x_i^l(k)) b_{j,i,t}^l(k)\right)^2$$

Therefore instantaneous gradient:

$$\hat{\nabla}(k) = \frac{\partial \bar{\varepsilon}(k)^2}{\partial b_{j,i,t}^l} = -2\varepsilon_j^{l+1}(k)\mu_{A_{i,t}^l}(x_i^l(k))$$

In order to tune values $b_{j,i,t}^l$, the gradient descent method is applied as:

$$\Delta b_{j,i,t}^l(k) = -\hat{\nabla}(k) = 2p'\varepsilon_j^{l+1}(k)\mu_{A_{i,t}^l}(x_i^l(k))$$

$$b_{j,i,t}^l(k+1) = b_{j,i,t}^l(k) + p\mu_{A_{i,t}^l}(x_i^l(k))\varepsilon_j^{l+1}(k)$$

where p is the learning parameter. Note that the widely adopted simple back-propagation algorithm is obtained [18][19][21]. Using more layers and outputs the typical error-back-propagation can be used [18][19][21].

3.2.4 Complexity reduction of the GNN

One of the main problems of applying fuzzy or neural technique is the calculation complexity. Engineers have to face this problem in complex systems or especially in the field of information retrieval where the extremely large information maps of whole libraries or internet have to be processed at each user's request. These applications often use the GNN algorithm.

The main objective of this section is to propose the singular value-based complexity reduction [16][17][22][23][24][25][26], which is capable of filtering out common linear combinations and reduce the number of antecedent sets based on the transformation of the weighting functions. One of the main advantages of the proposed method is that the effectiveness of the compression is controlled by the help of given error threshold. First, the method is presented for exact reduction, then finally it will be modified to have reduction error controllable property.

Using singular value-based complexity reduction method, equation (3.9) can always be transformed into the following form:

$$y_j^{l+1} = \sum_{z=1}^{n_{l+1}^{\ r}} a_{j,z}^{l} \sum_{i=1}^{n_l} \sum_{t=1}^{m_i^{l\,r}} \mu_{A_{i,t}^{l\ r}}(y_i^l) b_{z,i,t}^{l\ r} \tag{3.10}$$

where "r" denotes "reduced", further $n_{l+1}^{\ r} \le n_{l+1}$ and $\forall i : m_i^{l\,r} \le m_{l,i}$.

The reduced form is represented as neural network on figure 3.5. We use a system with three inputs x_1, x_2, x_3 and two outputs y_1, y_2 for illustration. Here, $n_l = 3$, $n_{l+1} = 2$, $i = 1..3$, $j = 1..2$, $m_1^{l\,r} = m_2^{l\,r} = m_3^{l\,r} = 2$ and $n_{l+1}^{\ r} = 2$.

The definition of appendix 2 briefly introduces singular value-based reduction (SVDR) which is the basic concept of the following calculation.

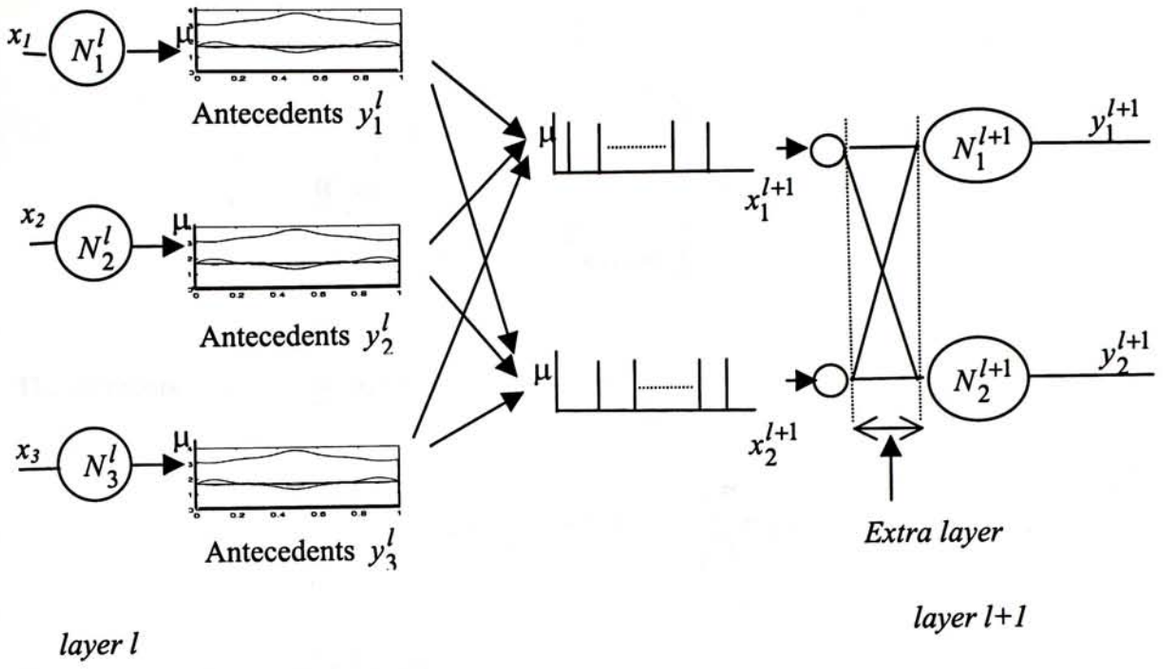


Figure 3.5. Reduced GNN

The parameters of (3.14) can be introduced in the followings. For convenience, we divide the reduction method into 2 steps. We define:

$$\underline{\underline{\mathbf{A}}}^l = [a^l_{j,z}],\ \underline{\underline{\mathbf{B}}}_i^{l^r} = \begin{bmatrix} b^l_{1,i,1}{}^r & \cdots & b^l_{n^r_{l+1},i,1}{}^r \\ \vdots & \ddots & \vdots \\ b^l_{1,i,m_i^{l^r}}{}^r & \cdots & b^l_{n^r_{l+1},i,m_i^{l^r}}{}^r \end{bmatrix} \text{ and}$$

$$\forall i: \left[\mu_{A^l_{i,t=1}}{}_r(y_i^l) \cdots \mu_{A^l_{i,t=m_i^{l^r}}}{}_r(y_i^l) \right]^{\mathrm{T}} = \underline{\underline{\mathbf{T}}}_i^{l^{\mathrm{T}}} \left[\mu_{A^l_{i,t=1}}(y_i^l) \cdots \mu_{A^l_{i,t=m_i^l}}(y_i^l) \right]^{\mathrm{T}}.$$

Step1:

Let matrix

$$\underline{\underline{\mathbf{S}}}^l = \left[\underline{\underline{\mathbf{B}}}_1^l \cdots \underline{\underline{\mathbf{B}}}_{n_l}^l \right] = [s^l_{v,z}] \tag{3.11}$$

whose size is $n_{l+1} \times \sum_{i=1}^{n_l} m_i^l$ and where

$$\underline{\underline{\mathbf{B}}}_i^l = \begin{bmatrix} b_{1,i,1}^l & \cdots & b_{1,i,m_i^l}^l \\ \vdots & \ddots & \vdots \\ b_{n_{l+1},i,1}^l & \cdots & b_{n_{l+1},i,m_i^l}^l \end{bmatrix} \text{ and } i = 1..n_l .$$

The elements of matrix $\underline{\underline{\mathbf{S}}}^l$ are hence:

$$s_{j,o+t}^l = b_{j,i,t}^l, \text{ where } o = \sum_{p=1}^{i-1} m_p^l$$

Then applying SVDR to $\underline{\underline{\mathbf{S}}}^l$ yields:

$$\underline{\underline{\mathbf{S}}}^l = \underline{\underline{\mathbf{A}}}^l \underline{\underline{\mathbf{D}}}^l \underline{\underline{\mathbf{V}}}^l = \underline{\underline{\mathbf{A}}}^l \underline{\underline{\mathbf{S}}}^{l\prime}$$

With $\underline{\underline{\mathbf{A}}}^l = [a_{j,z}^l]$ whose size is $n_{l+1} \times n_{l+1}^r$ and with $\underline{\underline{\mathbf{S}}}^{l\prime} = \left[s_{v,z}^{l\prime}\right]$ whose size is $n_{l+1}^r \times \sum_{i=1}^{n} m_i^l$ and partitioned as $\underline{\underline{\mathbf{S}}}^{l\prime} = \left[\underline{\underline{\mathbf{B}}}_1^{l\prime} \quad \cdots \quad \underline{\underline{\mathbf{B}}}_{n_l}^{l\prime}\right]$, one obtains $\underline{\underline{\mathbf{B}}}_t^{l\prime} = [b_{j,i,t}^{l\prime}]$ as:

$$b_{j,i,t}^{l\prime} = s_{j,o+t}^{l\prime}, \text{ where } o = \sum_{p=1}^{i-1} m_p^l \text{ and } j = 1..n_{l+1}^r$$

Here, one of the parameters $\underline{\underline{\mathbf{A}}}^l = [a_{j,z}^l]$ of (3.10) can be obtained.

Step2:

Let $\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}} = \begin{bmatrix} b_{1,i,1}^{l\prime} & \cdots & b_{n_{l+1}^r,i,1}^{l\prime} \\ \vdots & \ddots & \vdots \\ b_{1,i,m_i^l}^{l\prime} & \cdots & b_{n_{l+1}^r,i,m_i^l}^{l\prime} \end{bmatrix}$, where $i = 1..n_l$.

Applying SVDR to $\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}}$, the following equation obtained:

$$\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}} = \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{D}}}_i^l \underline{\underline{\mathbf{V}}}_i^l = \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{B}}}_i^{l^r}, \ i = 1..n_l .$$

The size of $\underline{\underline{\mathbf{T}}}_i^l$ is $m_l \times m_i^{l^r}$ and $\underline{\underline{\mathbf{B}}}_i^{l^r}$ is $m_i^{l^r} \times n_{l+1}^r$. The elements of matrix $\underline{\underline{\mathbf{B}}}_i^{l^r}$ are defined in the following way:

$$\underline{\underline{\mathbf{B}}}_i^{l^r} = \begin{bmatrix} b_{1,i,1}^{l\ r} & \cdots & b_{n_{l+1}^r,i,1}^{l\ r} \\ \vdots & \ddots & \vdots \\ b_{1,i,m_i^{l^r}}^{l\ r} & \cdots & b_{n_{l+1}^r,i,m_i^{l^r}}^{l\ r} \end{bmatrix}$$

$\mu_{A_{i,t}^l\ r}(y)$ can be obtained by $\underline{\underline{\mathbf{T}}}_i^l$ as the following way:

$$\forall i: \left[\mu_{A_{i,t=1}^l\ r}(y_i^l) \cdots \mu_{A_{i,t=m_i^{l^r}}^l\ r}(y_i^l) \right]^{\mathrm{T}}$$

$$= \underline{\underline{\mathbf{T}}}_i^{l\,\mathrm{T}} \left[\mu_{A_{i,t=1}^l}(y_{l,i}) \cdots \mu_{A_{i,t=m_i^l}^l}(y_i^l) \right]^{\mathrm{T}}$$

Consequently, the parameters of (3.10) are obtained.

Remark and further characterisation:

The functions of the transformed antecedent sets may not hold the *Ruspini* partition, hence, the PSG algorithm must be theoretically modified. It simply means that the reduced algorithm is applicable to implement the original approximation independently on the *Ruspini* partition.

If the reduced form is not only for executing the reduced algorithm in a real application, but it is for further studies in neuro-fuzzy operation, then the reduced form should accommodate additional characterisation pertaining to specific operation. This may require further

transformations. For instance, to obtain matrices $\underline{\underline{\mathbf{T}}}_i^l$ in such a way that the transformed functions are bounded by [0,1]. Some tools, such as *non-negativeness*, *sum-normalisation* and *set normalisation*, have been developed for these purposes [16][17][23]. To maintain the *Ruspini* partition in the case of piece-wise linear sets, the function SVDR is extended (SVDE) with *non-negativeness* and *sum-normalization* [16], which use in matrices $\underline{\underline{\mathbf{T}}}_i^l$, where all elements are in interval [0,1] and the sum of the elements in each row is one. Consequently, if SVDE is used instead of SVDR, the obtained reduced form (3.10) describes a fuzzy rule base, where the antecedent fuzzy sets are defined in *Ruspini* partition. In this case the use of (3.10) is theoretically in full accordance with product-sum-gravity inference. Appendix 3 briefly discusses two transformations and extends SVDR to SVDE.

3.2.5 Error bound of in-exact reduction of the GNN

This section introduces the error bound of in-exact reduction of GNN [27]. If nonzero singular values are discarded, the reduction brings error. The error can be estimated based on the discarded singular values since the error by SVDR (SVDE) is the sum of the discarded singular values [16][17][23]. The error of the reduced network also depends on the used antecedent function type. Various estimation is given in [16] according to the special functions.

Step1:

Let vector

$$\underline{\boldsymbol{\mu}}_{=0} = \left[\mu_{1,1}^l(y_1^l) \;\cdots\; \mu_{1,m_1^l}^l(y_1^l) \;\cdots\; \mu_{n_l,m_{n_l}^l}^l(y_{n_l}^l) \right]^{\mathrm{T}}$$

whose length is $\sum_{i=1}^{n_l} m_i^l$ and matrix (3.11) $\underline{\underline{\mathbf{S}}}^l = \left[\underline{\underline{\mathbf{B}}}_1^l \cdots \underline{\underline{\mathbf{B}}}_{n_l}^l \right]$. With this notation:

$$\underline{\mathbf{y}}^{l+1} = \underline{\underline{\mathbf{S}}}^l \underline{\boldsymbol{\mu}}_{=0} \tag{3.12}$$

Let apply the above described singular value-based reduction to the $\underline{\underline{\mathbf{S}}}^l$ matrix and discard the non-zero singular values:

$$\underline{\underline{\mathbf{S}}}^{l} \approx \underline{\underline{\mathbf{A}}}^{l}\underline{\underline{\mathbf{D}}}^{l}\underline{\underline{\mathbf{V}}}^{l} = \underline{\underline{\mathbf{A}}}^{l}\underline{\underline{\mathbf{S}}}^{l\prime}$$

The error of the reduction will be the sum of the discarded singular values, $\sum_{i=n_r+1}^{n_{SVD}} \lambda_i = E_1$ [16].

$$\left|\underline{\underline{\mathbf{S}}}^{l} - \underline{\underline{\mathbf{A}}}^{l}\underline{\underline{\mathbf{S}}}^{l\prime}\right| \le \left(\sum_{i=n_r+1}^{n_{SVD}} \lambda_i\right)\underline{\underline{\mathbf{1}}}_{(n_1\times n_2)} \le E_1\underline{\underline{\mathbf{1}}}_{(n_1\times n_2)} \tag{3.13}$$

The difference between the outputs of the original and the reduced network from equation (3.12) and (3.13):

$$\underline{\mathbf{y}}^{l+1\prime} = \underline{\underline{\mathbf{A}}}^{l}\underline{\underline{\mathbf{S}}}^{l\prime}\underline{\boldsymbol{\mu}}_{=0}$$

$$\left|\underline{\mathbf{y}}^{l+1} - \underline{\mathbf{y}}^{l+1\prime}\right| = \left|(\underline{\underline{\mathbf{S}}}^{l} - \underline{\underline{\mathbf{A}}}^{l}\underline{\underline{\mathbf{S}}}^{l\prime})\underline{\boldsymbol{\mu}}_{=0}\right| \le \left|\sum_{i=1}^{n_l}\sum_{t=1}^{m_i^l}\mu_{i,t}^{l}(y_i^l)E_1\right| = \left|\sum_{i=1}^{n_l}E_1\right| = n_l E_1 \tag{3.14}$$

Step2:

Let define the following vector:

$$\underline{\boldsymbol{\mu}}_{=i}(y_i^l) = \left[\mu_{i,1}^{l}(y_1^l) \quad \cdots \quad \mu_{i,m_{l,i}}^{l}(y_i^l)\right]$$

And $\underline{\underline{\mathbf{S}}}^{l\prime}$ can be partitioned as $\underline{\underline{\mathbf{S}}}^{l\prime} = \left[\underline{\underline{\mathbf{B}}}_{=1}^{l\prime} \quad \cdots \quad \underline{\underline{\mathbf{B}}}_{=n_l}^{l\prime}\right]$. With this notification:

$$\underline{\mathbf{y}}^{l+1,T} = \sum_{i=1=i}^{n_l}\underline{\boldsymbol{\mu}}_{=i}(y_i^l)\underline{\underline{\mathbf{B}}}_{=i}^{l,\mathrm{T}}\underline{\underline{\mathbf{A}}}^{l^{\mathrm{T}}} \tag{3.15}$$

Let apply the above described singular value-based reduction to the $\underline{\underline{\mathbf{B}}}_{=i}^{l,\mathrm{T}}$ matrices and discard the non-zero singular values:

$$\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}} \approx \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{D}}}_i^l \underline{\underline{\mathbf{V}}}_i^l = \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{B}}}_i^{l^r} .$$

The error of this reduction $\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}} - \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{B}}}_i^{l^r}$ will be the sum of the discarded singular values, $E_{2,i}$. The output of the reduced neural network will be

$$\underline{\mathbf{y}}^{l+1,T} = \sum_{i=1}^{n_l} \underline{\underline{\boldsymbol{\mu}}}_i^{\,r}(y_i^l) \underline{\underline{\mathbf{B}}}_i^{l^r} \underline{\underline{\mathbf{A}}}^{l^T} , \tag{3.16}$$

where

$$\underline{\underline{\boldsymbol{\mu}}}_i^{\,r}(y_i^l) = \left[\mu_{i,1}^{l\ r}(y_i^l) \quad \cdots \quad \mu_{i,m_i^l}^{l\ r}(y_i^l) \right] = \underline{\underline{\boldsymbol{\mu}}}_i(y_i^l) \underline{\underline{\mathbf{T}}}_i^l$$

The error $E_{2,i}$ from equation (3.15) and (3.16) is:

$$\left| \underline{\mathbf{y}}^{l+1,T} - \underline{\mathbf{y}}^{l+1,,T} \right| = \left| \sum_{i=1}^{n_l} \underline{\underline{\boldsymbol{\mu}}}_i(y_i^l) (\underline{\underline{\mathbf{B}}}_i^{l,\mathrm{T}} - \underline{\underline{\mathbf{T}}}_i^l \underline{\underline{\mathbf{B}}}_i^{l^r}) \underline{\underline{\mathbf{A}}}^{l^{\mathrm{T}}} \right| \le \left| \sum_{i=1}^{n_l} \underline{\underline{\boldsymbol{\mu}}}_i(y_i^l) \underline{\underline{\mathbf{1}}}_{(m_i^l \times n_{l+1}^r)} E_{2,i} \underline{\underline{\mathbf{A}}}^{l^T} \right|$$

$$\le \left| \sum_{i=1}^{n_l} \sum_{t=1}^{m_i^l} \sum_{z=1}^{n_{l+1}^r} \mu_{i,t}^l(y_i^l) a_{j,z}^l E_{2,i} \right| \le \left| \sum_{i=1}^{n_l} E_{2,i} \right| \tag{3.17}$$

The last step is valid, only if the sum of the rows of $\underline{\underline{\mathbf{A}}}^l$ was less or equal than one. This could be easily made, if in the first step a further transformation is made.

So the error in the reduction altogether is from equation (3.14) and (3.17):

$$E \le \sum_{i=1}^{n_l} E_{2,i} + n_l E_1 .$$

3.3 NEURAL-FUZZY-BASED POSITION CONTROL OF FOUR LINKING SMA WIRES

The experimental setup is expanded for position controlling the linking point of 4 SMA wires. As depicted in Figure 3.6, the linking point is now free to move in a two-dimensional plane, instead of the one-dimensional previously. In this case, the equilibrium condition requires that the total strain of the wires must be constant and that their stresses must be balanced along each of the x and y direction. Though technically the model-based method can still be applied for this situation, it will be a very tedious endeavor. Moreover, there are now more nonlinear effects to render the model-based approach less effective in coming up with reasonable performance. Hence, we opt here to control the system using a neural-fuzzy based approach. First, we propose using ANFIS to control this system and then GNN.

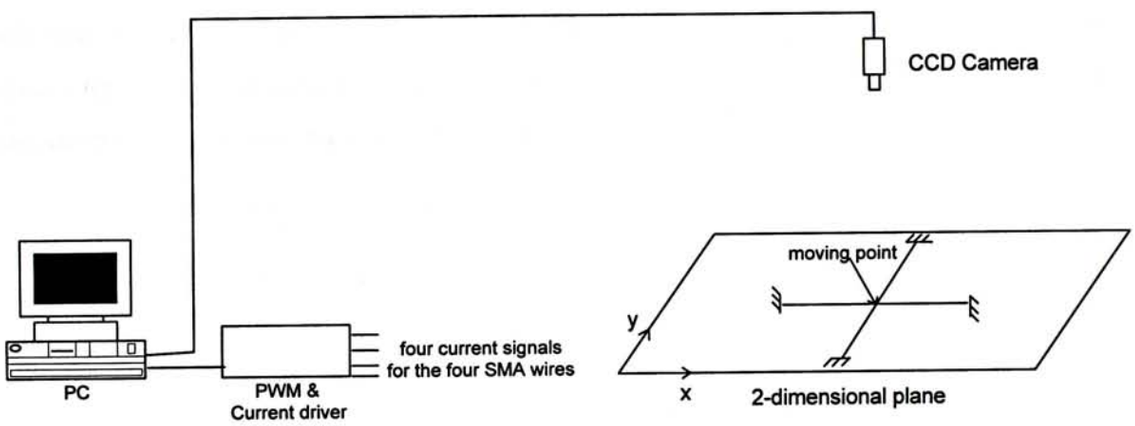


Figure 3.6. Position control system of 4 SMA wires in 2-dimensional plane.

3.3.1 ANFIS-based Position Control of Four Linking SMA Wire

Experimentation starts with generating 1000 data sets for training. Each of these data sets is obtained by first marking the current position (x_c,y_c) of the linkage point of the SMA wires, and then by commanding currents (i_0, i_1, i_2, i_3) through the SMA wires and measuring the actual position (x_a,y_a) of the linkage as resulted. Current inputs to the wires are randomly assigned except that to avoid damaging the SMA wires caused by contradicting command, only one of the neighboring pairs (see Figure 3.6) of the SMA wires is activated for each data set. The data sets are then utilized to train four 4-inputs 1-output ANFISs, one for each wire. The training objective is using the current position (x_c, y_c) and actual position (x_a, y_a) as input, calculate the output applied current ($i_{s,0}$, $i_{s,1}$, $i_{s,2}$, $i_{s,3}$). But it must have an error between

i_0, i_1, i_2, i_3 and $i_{s,0}$, $i_{s,1}$, $i_{s,2}$, $i_{s,3}$. We want to minimize the error and get the trained ANFIS for controlling the system.

Inputs to the ANFISs are the *(x,y)* coordinates of the current position and the to-be-resulted final position of the linking point, and the outputs are the currents for the wires. The ANFISs here each contains 256 rules, with four membership functions assigned to each input variable. The membership functions are all in Gaussian bell shape and all the rule consequence are singleton, meaning that $p_i = 0$, $q_i = 0$. The number of fitting parameters is hence 304, including 48 premise parameters and 256 consequent parameters, for each ANFIS. Convergence of training in this case is fast, shown in figure 3.7, reaching steady state values in around 100 epochs or so. Figure 3.8 and 3.9 show the resulting memberships and singleton rule consequents, respectively, for the ANFIS obtained for wire0.

Upon training of the ANFISs, corresponding currents to the wires are then generated to command the linking point to move around a circular path. Figure 3.10 compares the circular path with the actual trajectory of the linking point obtained through activation of the SMA wires with the ANFIS-generated currents. The performance in this case is quite reasonable considering, in particular, the simplicity of the present approach.

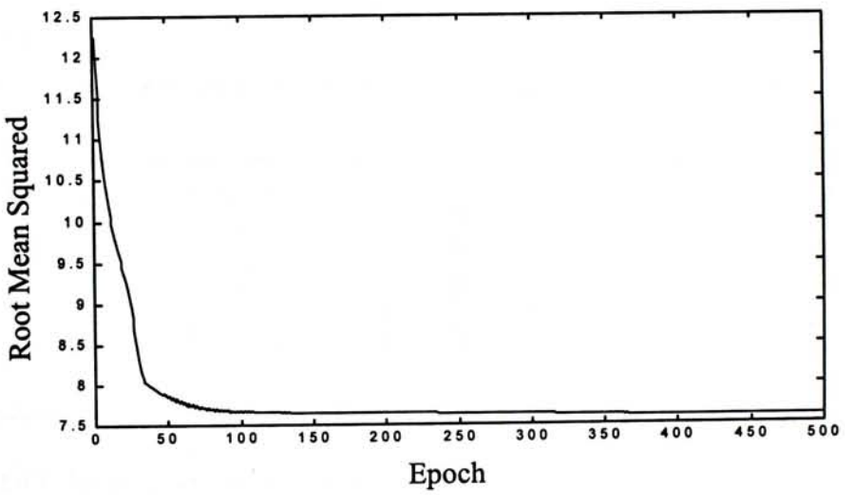


Figure 3.7. Root mean squared error for 1000 training patterns in 500 epochs.

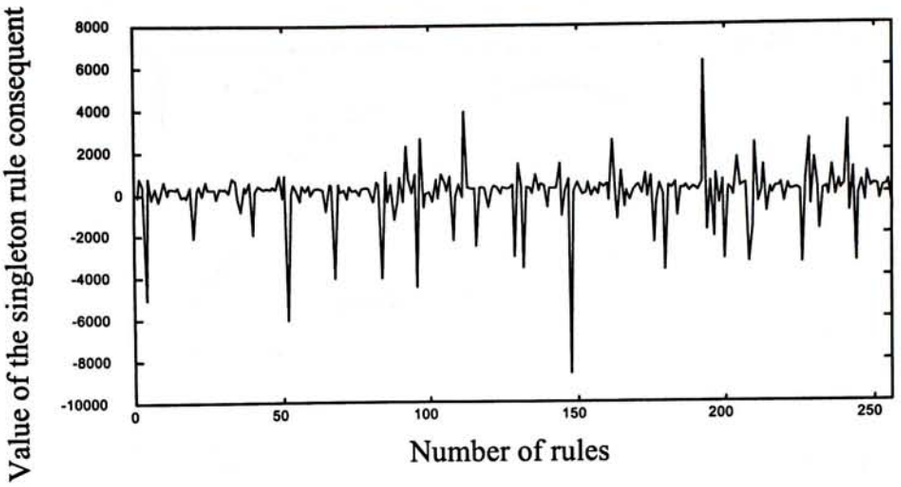


Figure 3.8. Singleton rule consequent of the trained ANFIS for wire0.

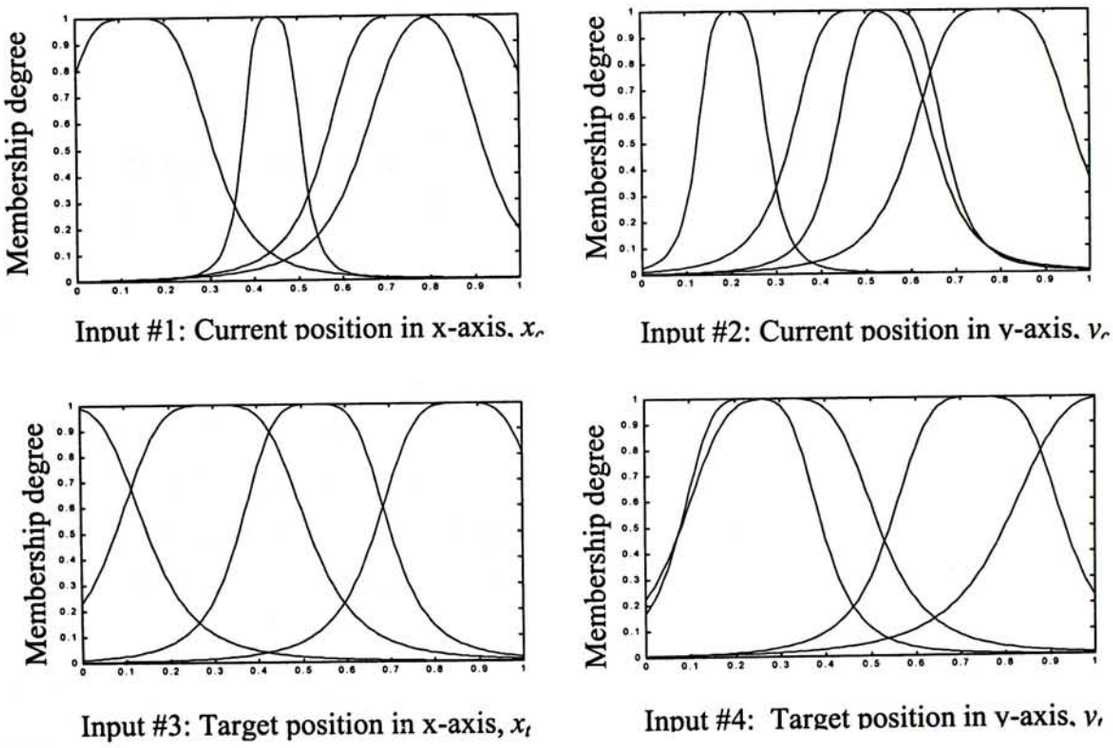


Figure 3.9. Input membership functions of the trained ANFIS for wire0.

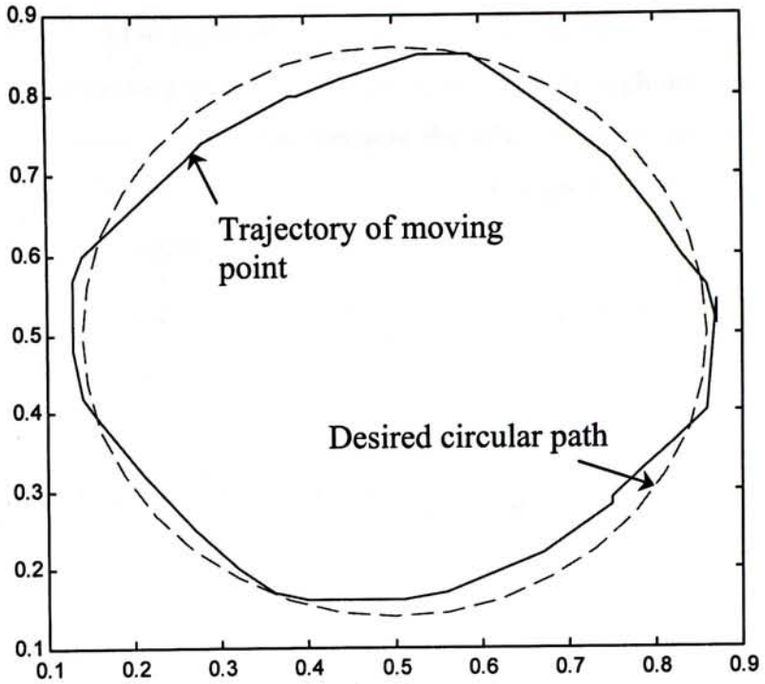


Figure 3.10. Comparison of commanded circular path and experimental trajectory for the 4 SMA wire linkage system using ANFIS algorithm.

3.3.2 GNN-based Position Control of Four Linking SMA Wire

In this experimentation, we also generated 1000 data sets for training. Each of these data sets is obtained by the last four moving currents (c_0, c_1, c_2, c_3) of the SMA wires, and then by commanding currents (i_0, i_1, i_2, i_3) through the SMA wires and measuring the actual positions (x_a,y_a) of the linkage as resulted. Current inputs to the wires are randomly assigned except that to avoid damaging the SMA wires caused by contradicting command, only one of the neighboring pairs (see Figure 3.6) of the SMA wires is activated for each data set. The data sets are then utilized to train a 6-inputs 4-output GNN. The training objective is using the last four moving currents (c_0, c_1, c_2, c_3) and actual position (x_a, y_a) as input, calculate the output applied current ($i_{s,0}$, $i_{s,1}$, $i_{s,2}$, $i_{s,3}$). But it must have an error between i_0, i_1, i_2, i_3 and $i_{s,0}$, $i_{s,1}$, $i_{s,2}$, $i_{s,3}$. We want to minimize the error and get the trained GNN for controlling the system.

Inputs to the GNN are the last moving currents of the current position and the to-be-resulted final position of the linking point, and the outputs are the currents for the wires. The GNN here contains 504 rules, with 21 membership functions assigned to each input variable. The membership functions are all in triangular shape, shown in figure 3.11 and all the rule consequence are singleton, shown in figure 3.12 for wire0.

Upon training of the GNN, corresponding currents to the wires are then generated to command the linking point to move around a circular path. Figure 3.13 compares the circular path with the actual trajectory of the linking point obtained through activation of the SMA wires with the GNN-generated currents. Because the GNN based controller is more simple than the ANFIS one, the performance in this case is not good than the last one. For the algorithm implementation, appendix 4 is presented.

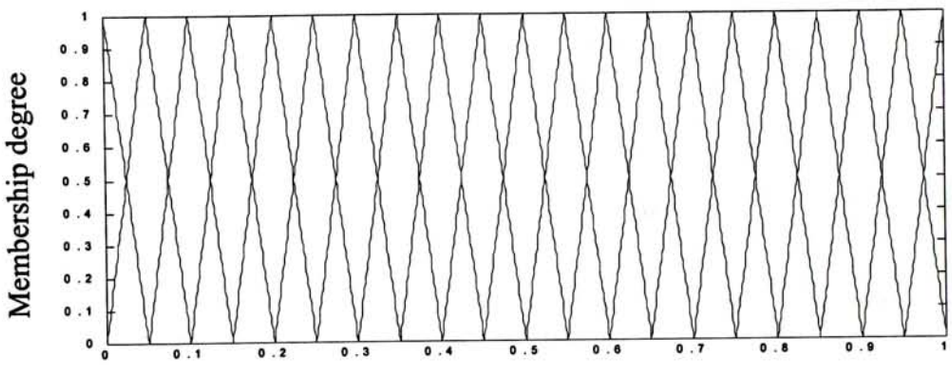


Figure 3.11. Input membership functions of GNN.

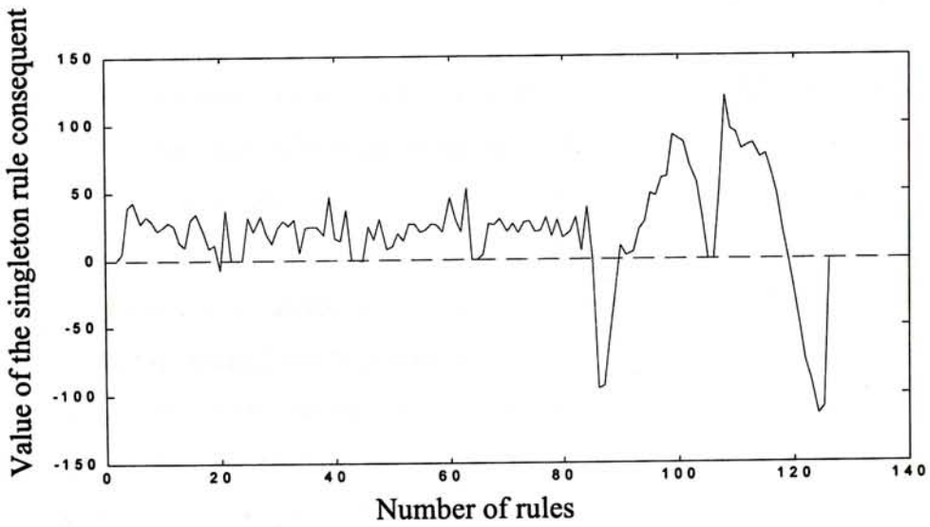


Figure 3.12. Singleton rule consequent of the trained GNN for wire0.

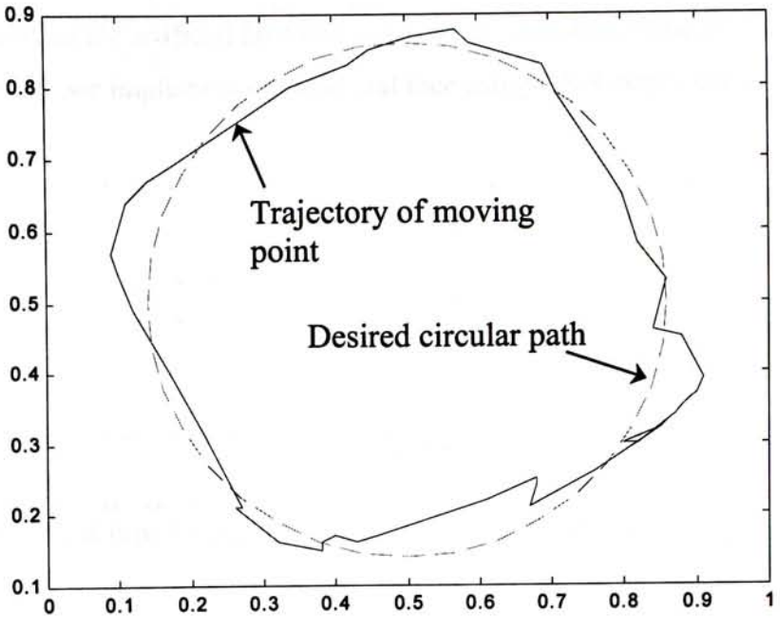


Figure 3.13. Comparison of commanded circular path and experimental trajectory for the 4 SMA wire linkage system using GNN algorithm with 504 rules.

In section 3.2.4, we introduced the complexity reduction of the GNN algorithm. The above GNN-based controller can be had an exact reduction according to the method of section 3.2.4. Figure 3.14 shows the input membership functions of reduced GNN. Each input remains 4 membership functions and the GNN remains 96 rules. Figure 3.15 compares the circular path with the actual trajectory of the linking point obtained through activation of the SMA wires with the reduced GNN-generated currents

From figure 3.13 and 3.15, the trajectory of the moving point is similar to a square with certain degree of rotation. This is because of the construction of the four actuating SMA wires, the direction of which are along the sides of the rotated square, and the present control algorithm is such that one of the wires always dominates over the others.

3.3.3 Performance Comparison of ANFIS and GNN Algorithms

In section 3.3.1 and 3.3.2, we showed the experimental results of ANFIS and GNN algorithms as controller to control the position control system of 4 SMA wires in 2-dimensional plane. Figure 3.10 and 3.13 show the trajectories of the moving points. We can see that the trajectory of figure 3.10 is better than figure 3.13. That means the performance of ANFIS-based controller is better than GNN-based controller. But the number of rules of ANFIS-based controller is 1024 and reduced GNN-based controller is 64. The different of

the number of rules is steep. In this project, we do not need to control the exact position of each control point on the artificial face and we want the calculation speed of the algorithm as fast as possible. So, we implement our artificial face using GNN-based controller.

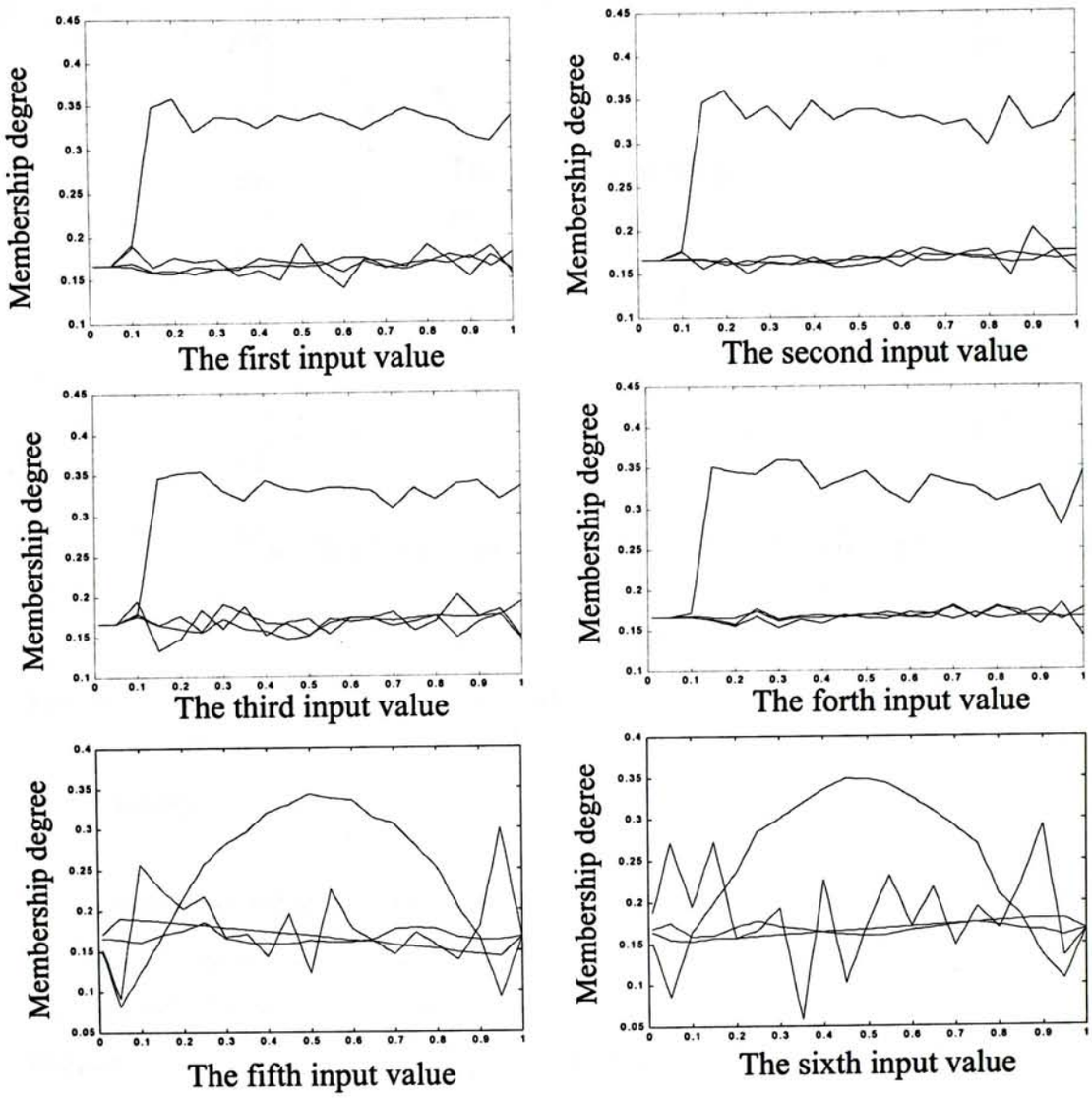


Figure 3.14. The input membership functions of reduced GNN.

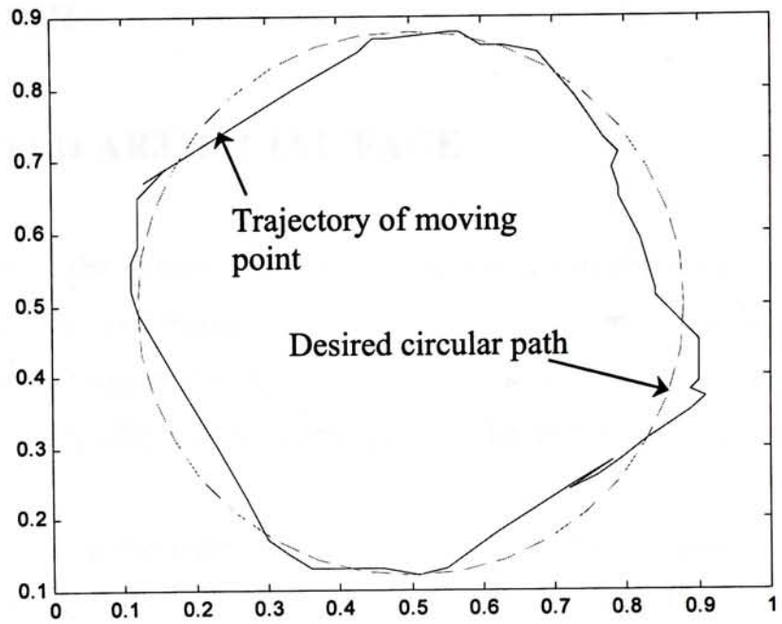


Figure 3.15. Comparison of commanded circular path and experimental trajectory for the 4 SMA wire linkage system using the reduced GNN algorithm with 96 rules.

3.4 Summary

This chapter presented two neural-fuzzy algorithms: ANFIS and GNN. The ANFIS is functionally equivalent to fuzzy inference system. However, its structure is more complicated. On the other hand, GNN harbors reasonable performance and a simpler structure, one that readily allow SVD complexity reduction to be applied. Respective performance of the ANFIS, GNN and reduced GNN in controlling the 2D position of a 4 linking SMA wire system are also included in the chapter. Since the present project involves the construction a SMA actuated artificial face, we desire a simple and fast calculation algorithm to the controller. For this reason, the reduced GNN is chosen for implementation on the artificial face.

CHAPTER FOUR

SMA ACTUATED ARTIFICIAL FACE

This chapter describes the construction of a SMA-actuated artificial face for mimicking human facial expressions. In a human face, numerous muscles are pulled in order to make the facial expressions. The same concept is used in our artificial face but instead of muscles, we use the shape memory alloy (SMA) wire to pull the skin of our artificial face to effectuate facial expressions.

In this project, we link up the software facial model of [28] to our hardware construction. The end result is by changing the facial expressions of the facial model in the computer, the artificial face will be actuated accordingly.

4.1 MUSCLES OF THE HUMAN FACE

In the general sense, muscles are the organs of motion [28]. The muscles of the face are commonly known as the muscles of facial expression. By their contractions, various facial expressions can be realized. Some facial muscles also perform other important functions, such as moving the cheeks and the lips, or closing and opening of the eyelids. The muscles used for facial expression are more on the surface. Some of the muscles attach to skin at both the origin and the insertion such as the obicularis oris, shown in figure 4.1. When the muscles are relaxed, the fatty tissues fill the hollows and smooth the angular transitions so as to allow the general shape of the skull to be seen. The illustration in figure 4.1 illustrates the surface muscles of the face.

The muscles of facial expression work collectively and not independently. The group functions as a well-organized and coordinated team, each member has specified functions, one of which is primary. These muscles mix with one another. It is difficult to separate the boundaries between the various muscles. The terminal ends of these muscles are interlaced with each other.

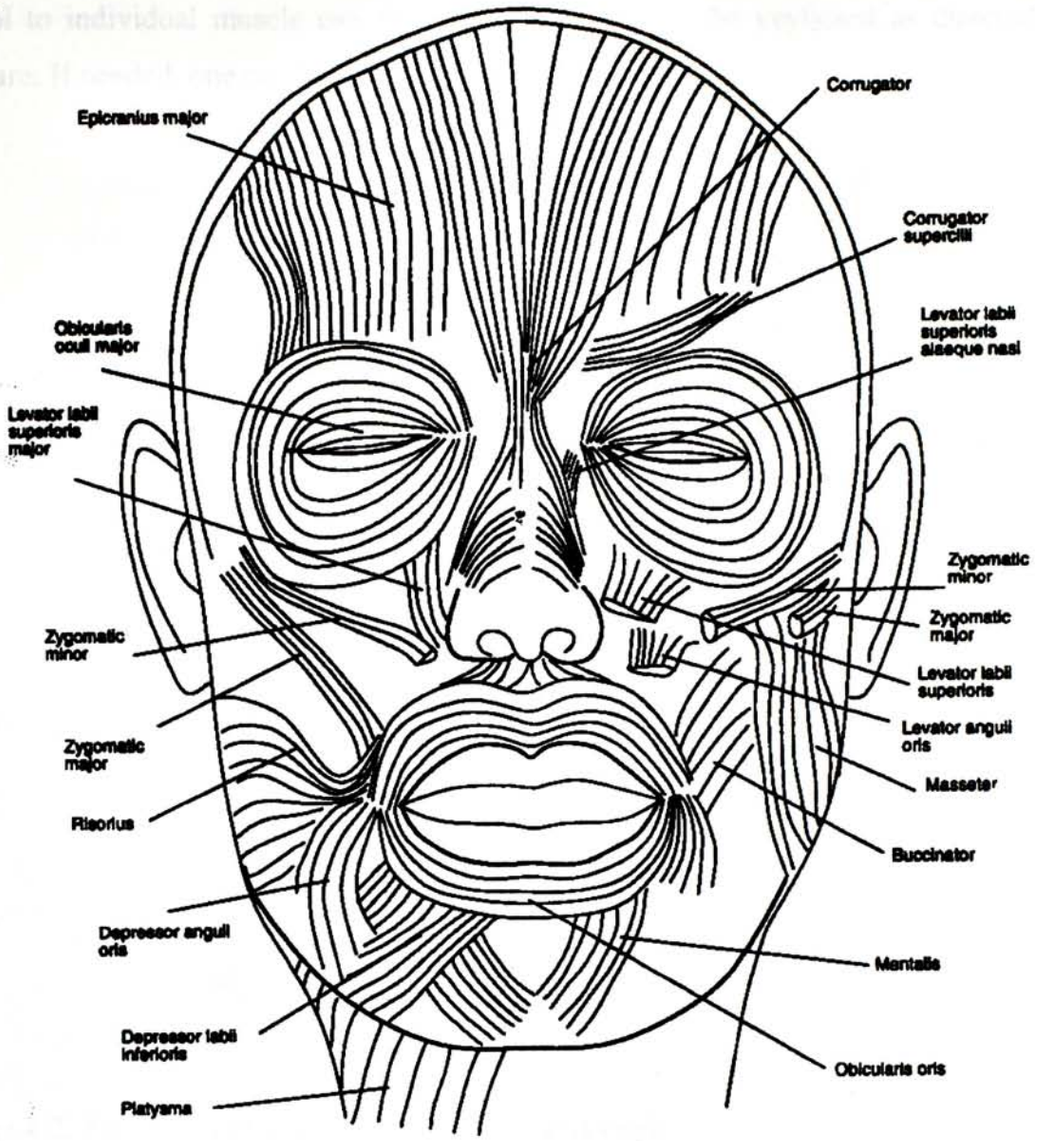


Figure 4.1. The frontal view of facial muscles [28].

4.2 THE SOFTWARE PART: facial model

The work of [28] presents a facial model which can display some facial expressions in the computer. We adapt the facial model of [28], written in C language, to the one as shown in figure 4.2. Corresponding to the hardware designs, there are 6 muscles on the facial model. These muscles can be contracted or released one-by-one by pressing the keyboard. For convenience, some preset facial expressions are also available in the face model. Once we decide on the facial expression in the software facial model, output commands will then be directed to the artificial face to realize the chosen expression.

Our computer setup is Pentium 233MHz, 64MB ram and the operating system is Microsoft Windows 98. At the start of the program, figure 4.3 will be shown. It will display two windows, the first one is the face model and the second one displays the message. Input

control to individual muscle can then be carried out via the keyboard as directed by the software. If needed, one can press "h" to get the help menu.

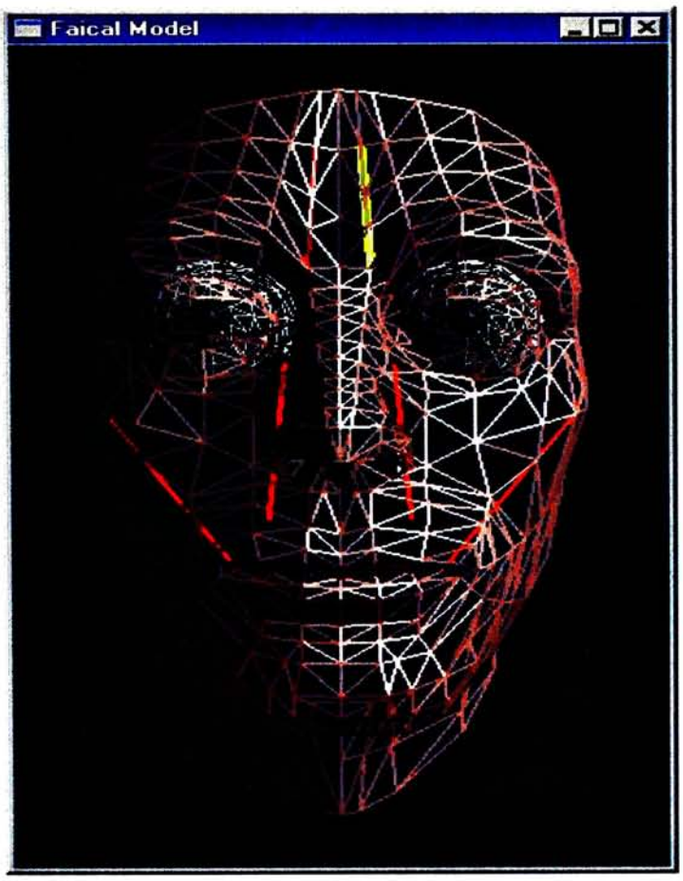


Figure 4.2. The geometry and muscles of the facial model.

Figure 4.3. The facial model and the command window.

4.3 THE HARDWARE PART: artificial face and peripheral interface

Figure 4.4 shows the setup of the whole system. It includes the artificial face and the electrical circuit board as the interface between the artificial face and the computer display. In the following, we will introduce the structure of the artificial face and the interface circuit.

Figure 4.4. The experimental setup of the artificial face system.

4.3.1 SMA Actuated Artificial Face

The artificial face is a human-sized face which facial expression actuated by SMA wires instead of human muscles. For the work of this thesis, we have 10 SMA wires corresponding to the primary muscles in the human face. These 10 SMA wires actuate 6 control points and 3 mechanical structures, shown in figure 4.5. Eight of them are constructed as spring-wire system which stands for a spring and a SMA wire joining together controlling the movement of eyebrows, lip and opening and closing of eyelid. Two of them control the opening and closing of the jaw and are constructed as wire-wire system which stands for two SMA wires joining together. The internal structure of the artificial face is shown in figure 4.6. The mechanical construction is shown in figure 4.7. All components, i.e., all construction of SMA wires and springs are embedded inside the skull.

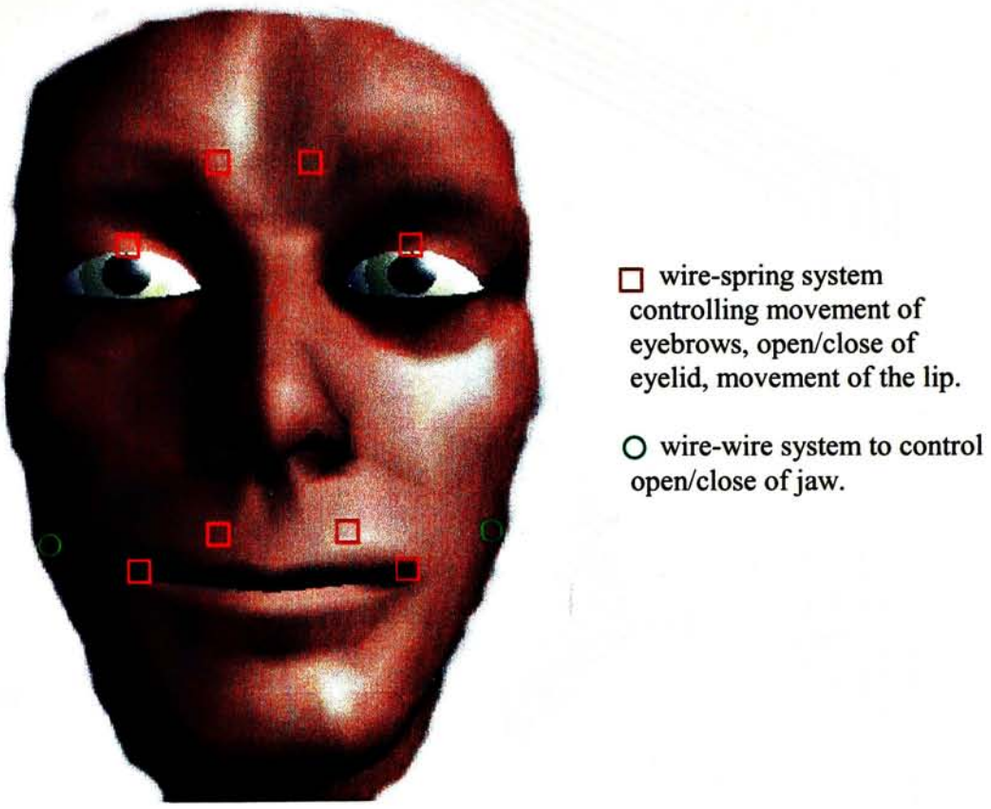


Figure 4.5. Definition of the control points.

Figure 4.6. The real structure of the artificial face.

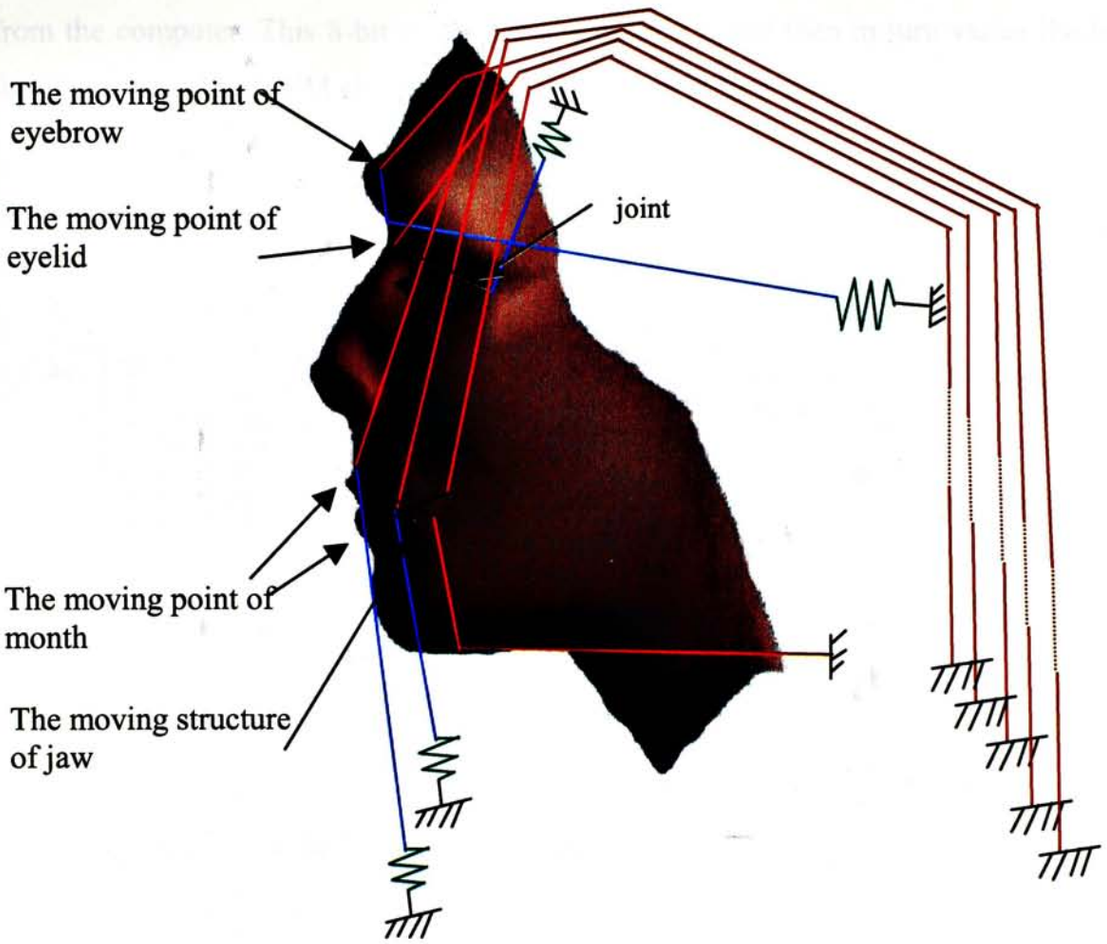


Figure 4.7. The mechanical construction of the artificial face (the red line presents the SMA wire, the blue line presents the fishing line and the green line presents the spring).

4.3.2 Peripheral Interface

All the output signal of the computer is digital but the SMA wire requires for its operation powered analog signal. Some electrical circuit interfaces are needed between the two. These include the 8255 input output controller card, pulse width modulator, and a current amplifier.

8255 input output controller card:

8255 input output controller card is installed in the PC expansion slot. Through software programming, it outputs many channels of 8-bits digital signal to the pulse width modulator.

Pulse width modulator:

The pulse width modulator (PWM) generates analog-like signal, the pulse width of which is controlled by the 8-bits digital signal from the 8255 input output controller card. The average current value of the analog-like signal hence corresponds directly to the 8-bits digital signal

from the computer. This 8-bit resolution analog-like signal then in turn varies the length of the SMA wire. The PWM electrical circuit is shown in figure 4.8.

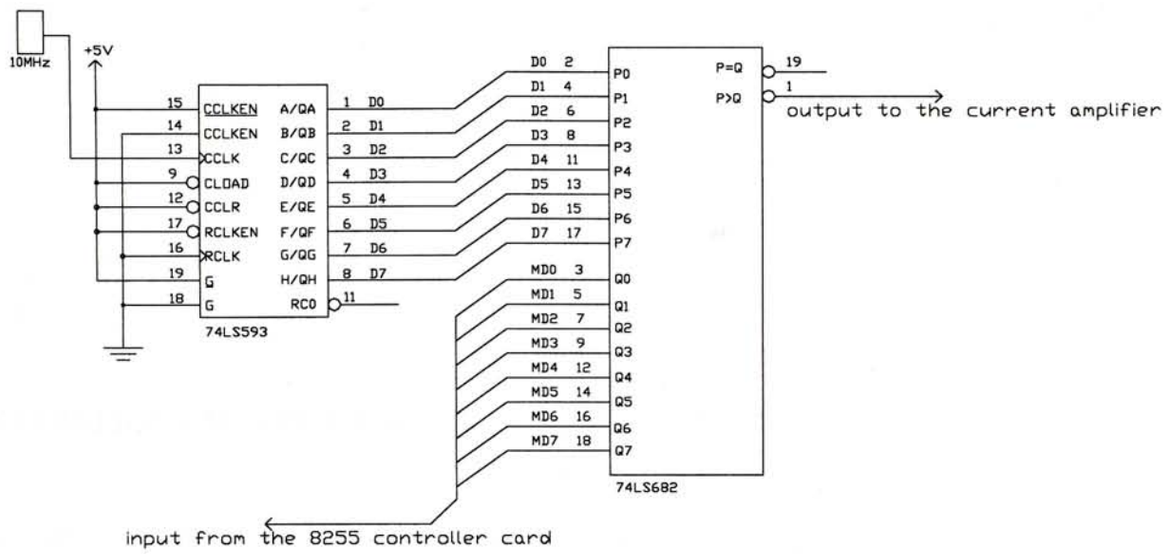


Figure 4.8. The electrical diagram of pulse width modulator.

Current amplifier:

Since the analog-like signal from the PWM is not powered, we need a current amplifier to power the signal to generate enough current to drive the SMA wire. As the original length of each SMA wire in the artificial face is different, the value of the resistance R_b, as shown in figure 4.9, needs to be turned appropriately in order to have the same current passing though each SMA wire.

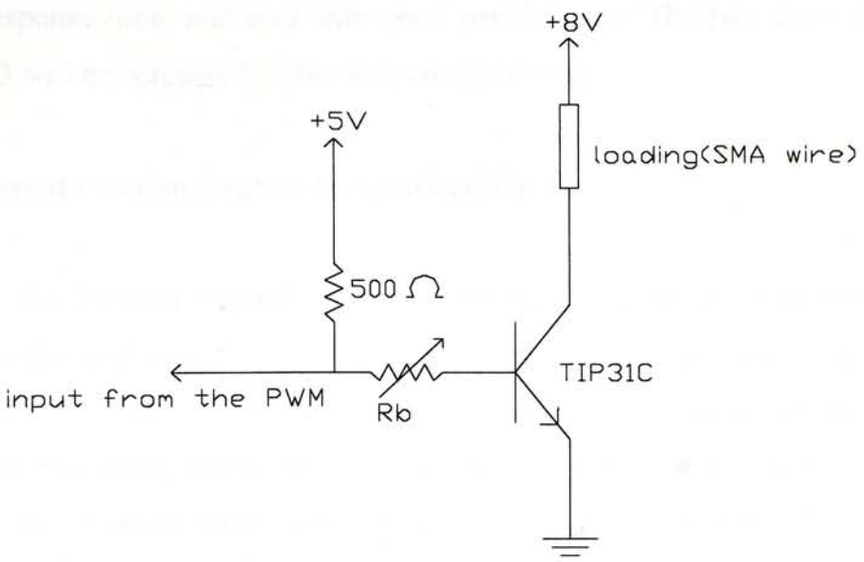


Figure 4.9. The electrical diagram of current amplifier.

4.4 POSITION CONTROL ON THE ARTIFICIAL FACE

From figure 4.7, there are 10 SMA wires on the artificial face. The position of the wire can be controlled by applying currents from the computer. Figure 4.10 shows the eyebrow as an example in control. The eyebrow is a spring-wire system. Because the spring is a passive device, we need only to control the position by actuating the single SMA wire.

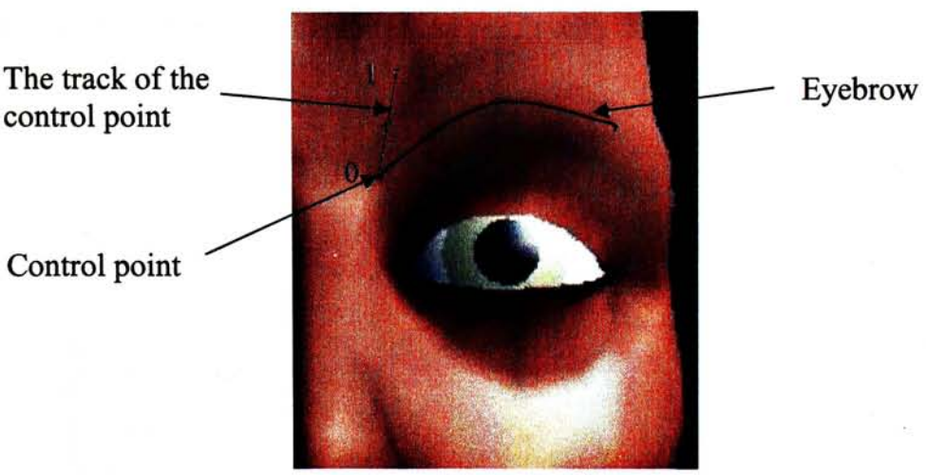


Figure 4.10. The right eyebrow of the artificial face. Definition of the position of the control point.

As mentioned before, Both the model-based control and the neural-fuzzy-based control in chapter 2 and 3 can be applied to controlling the various point on artificial face. However, as the geometry of the artificial face is complicated, we derive a control method which would

yield a fast response time, and also with good performance. The two control methods in chapter 2 and 3 will be compared under such considerations.

4.4.1 Model-based Position Control on Artificial Face

In chapter 2, the position control of a two linking SMA wires, wire-wire system is conducted. For the eyebrow of the artificial face here, it is a spring-wire system but the theory in chapter 2 can be applied just the same, except that the equation of one of the wires is now replaced by a spring displacement force. Assuming that the wire is always in tension during the process, it can be stated that at equilibrium the total strains of the spring and wire ε_{total} must be constant and that the stress in the spring σ_{spring} and wire σ_{wire} must be equal to each other. That is:

$$\begin{cases} \varepsilon_{wire} = (1 - R_{wire})\varepsilon_{A,wire} + R_{wire}\varepsilon_{M,wire} \\ K \cdot \varepsilon_{spring} = \sigma_{spring} \\ \varepsilon_{total} = \varepsilon_{wire} + \varepsilon_{spring} \\ \sigma_{wire} = \sigma_{spring} \end{cases} \tag{4.1}$$

where ε_{total} is a constant and can be measured from the experimental setup. K is the spring constant which can measure from the experiment. Then, using (4.1) with (2.3), (2.4) and (2.9), one can solve for ε_{wire}, i.e., the position of the control point, as a function of the wire current. Figure 4.11 shows the experimental result with this method.

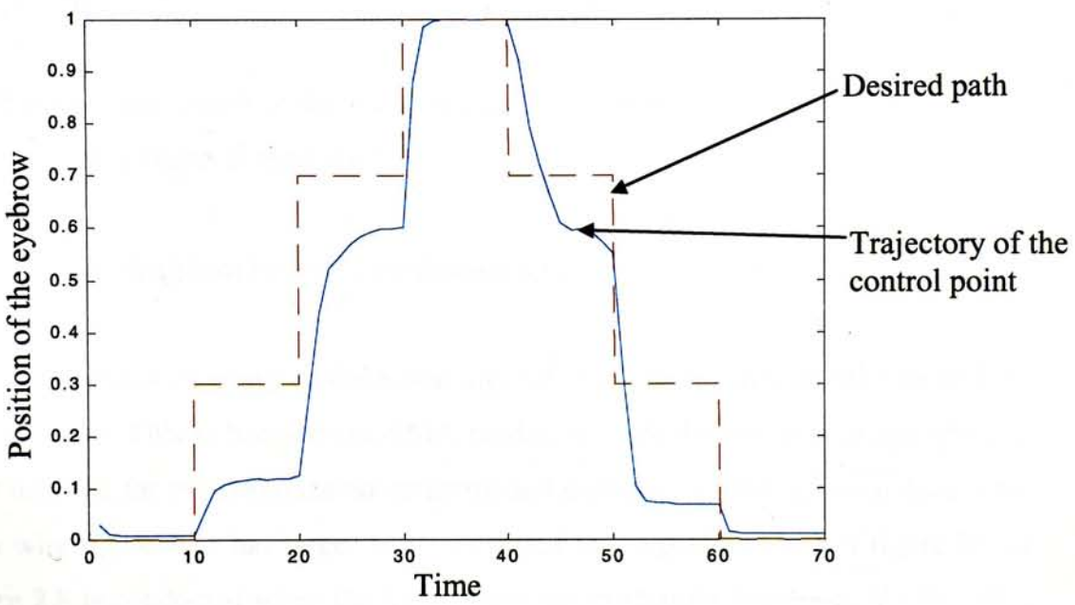


Figure 4.11. Experimental result of the position control of the right eyebrow of the artificial face using model-based control method.

4.4.2 Neural-fuzzy-based Position Control on Artificial Face

In chapter 3, we introduced two neural-fuzzy algorithms, ANFIS and GNN. We illustrated performance by the example of position control of four linking SMA wires in 2 dimensional plane. We also commented that, for simpler implementation with reasonable performance, the reduced GNN controller is to be adopted in our artificial face. To compare with the performance of the previous section, here we will control the eyebrow using the reduced GNN-based controller. We took 500 data sets as training pattern. Each of these data sets is obtained by recording the last position of the control point, and then measuring the ensuing position as resulted from commanding currents through the SMA wire. Figure 4.12 shows the experimental result using a trained GNN-based controller.

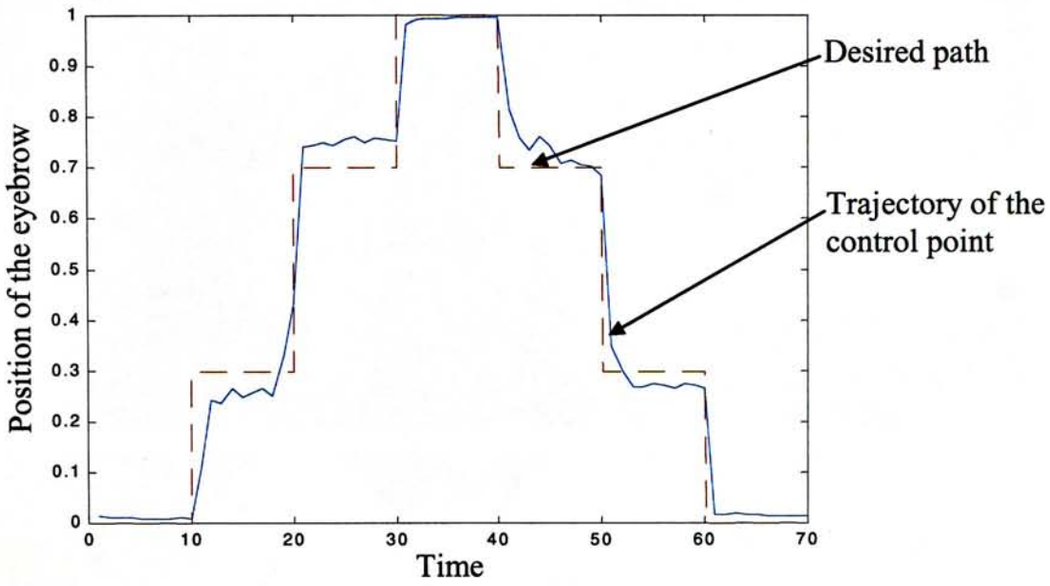


Figure 4.12. Experimental result of the position control of the right eyebrow of the artificial face using neural-fuzzy-based control method.

4.4.3 Comparison of the Model-based and Reduced GNN Control of Artificial Face

It is obvious that figure 4.11 using model-based algorithm is inferior in both the rate and the accuracy of response. This is because the SMA model, identified when it is in straight line form, cannot account for the complicated structure and geometry of the artificial face. This also explains why figure 4.11 has larger error compared to the performance of figure 2.8 as well, as figure 2.8 is conducted when the SMA wires are in straight line form. On the other hand, the neural-fuzzy based control as in the reduced GNN is more adaptive, trained always by experimental data at the moment. They hence show better control performance. We therefore opt to implement on our artificial face using the reduced GNN-based controller.

4.5 EXPERIMENTAL RESULT

Upon construction of the whole system, each SMA wire in the artificial face is trained one by one using independent reduced GNN algorithms. Afterwards, given the movement of each muscle in the software facial model, the reduced GNN algorithm calculates the appropriate current to be commanded to the corresponding SMA wires. This way, individual SMA wire in artificial face can be controlled to achieve the particular assigned control position. Figure 4.13 compares the expressions of the facial model and the artificial face for some preset facial expressions. Figure 4.14 shows other facial expressions.

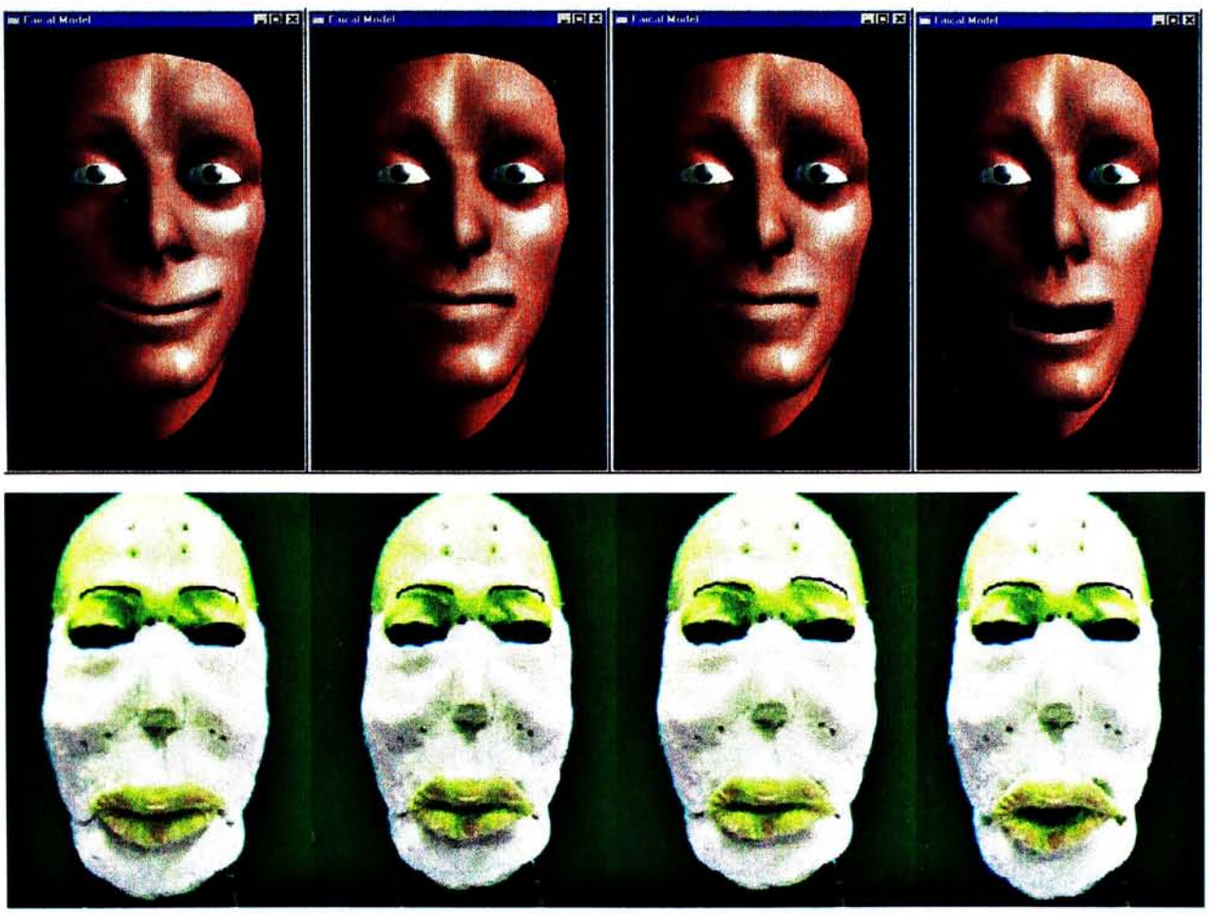


Figure 4.13. The comparison of the facial model and the artificial face in some facial expressions: happy, anger, disgust and jaw opening.

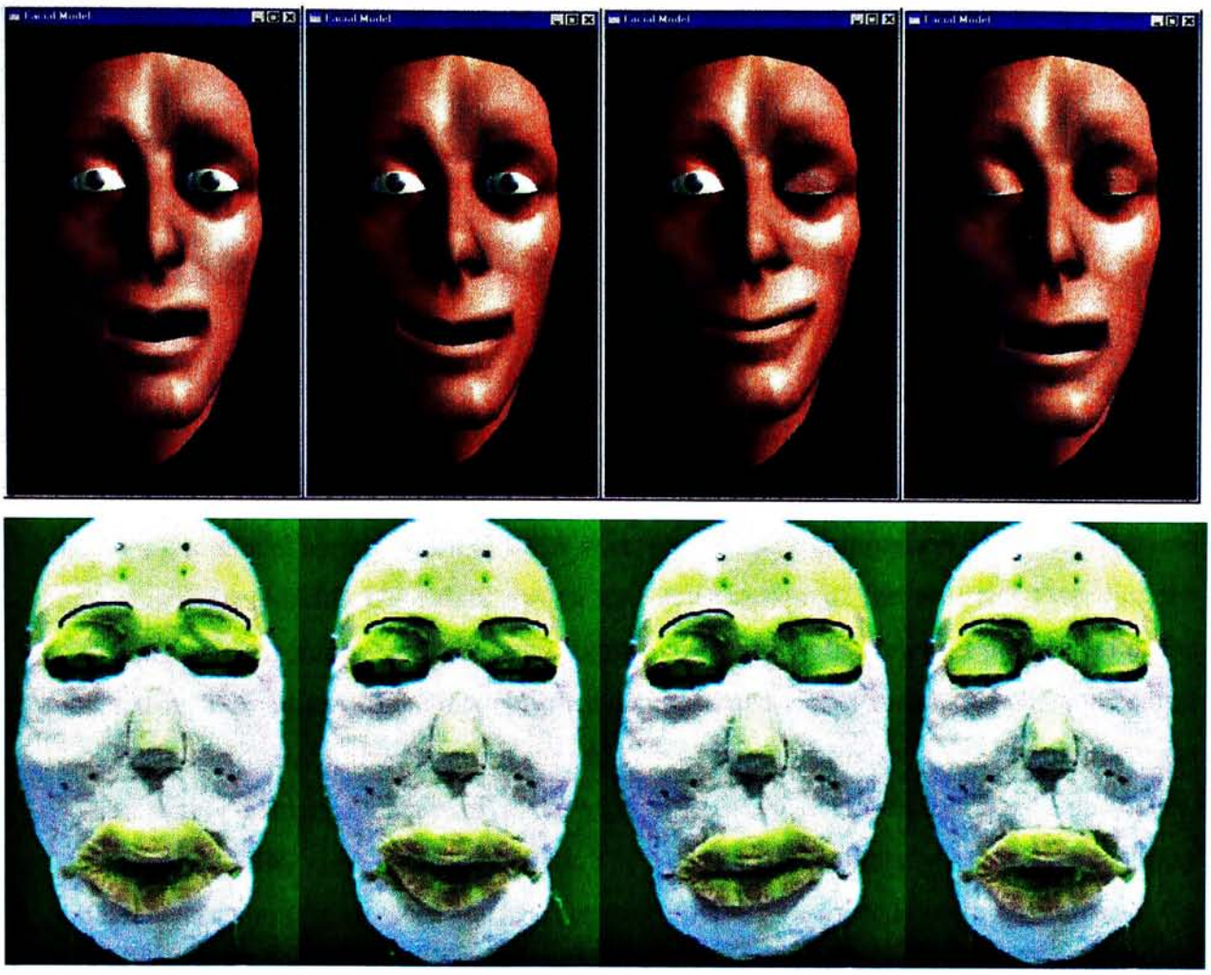


Figure 4.14. Facial expressions of the artificial face.

CHAPTER FIVE

CONCLUSION

In this project, we constructed a SMA-actuated artificial face which can mimic human facial expressions. Here, we adopted SMA wires for actuation because of it special features: clean, bio-friendly and compact construction. The present work is composed of 3 main parts: modeling of model identification of SMA wire, performance comparison of the model-based and neural-fuzzy-based algorithm, design and construction of the artificial face. For the modeling part, we adopt the model of K.Ikuta and identified the relevant parameters of the SMA wire. The validity of the model is then demonstrated via a 1 dimension position control experiment. However, understanding the validity of the model-based method is limited due to the highly nonlinear nature of the present setup, two neural-fuzzy based approaches, ANFIS and GNN, are also investigated. The ANFIS is functionally equivalent to fuzzy inference system. It gives very good performance but the structure is more complicated. On the other hand, the GNN is of reasonable performance but its structure is reducible using SVD complexity reduction. Because there are many SMA wires in the artificial face, to reduce the calculation time of the algorithm, we used the reduced GNN algorithm as the controller. As constructed the hardware artificial face is linked to a software facial model. From the dictated movement of the muscle in the software model, the GNN algorithm calculates the appropriate current to be commanded to the corresponding SMA wires. This way, individual SMA wire in the artificial face is controlled to the assigned control position. Hence, by dictating the facial expressions of the software facial model on the computer, the artificial face is then commanded to mimic that dictated expression

Eventually, we would like to hook up our artificial face with a video system which can recognize facial expressions of human in order to mimic their emotion. This is an area of future work. Another area of research is the response of the SMA wire. Slow response of the wire now makes slow implementation of the facial expression. Improving the response of the SMA wire hence enables mimicking of human facial expressions more lively. Also, the artificial face for now is controlled by an open loop control technique algorithm. For the more accurate movement, close loop control needs to be applied. This is also our future direction.

Appendix 1

Product-sum-gravity (PSG) [20]

Characterisation of input output

Let the input and output universe X and Y, respectively.

Characterisation of fuzzy sets

Antecedents: The antecedent fuzzy sets $A_t : \mu_{A_t}(x)$, $x \in X$, $t = 1..m$ are defined on universe X in *Ruspini*-partition, where m is the number of antecedent sets.

Consequents: The consequent fuzzy sets are defined as singleton, crisp sets:

$$B_t : \mu_{B_t}(y) = \delta(b_t), \ \forall t : b, y \in Y$$

defined on universe Y.

Observation: Let the input value x^* be fuzzificated into singleton observation fuzzy set A^* such as:

$$A^* : \mu_{A^*}(x) = \delta(x^*),$$

Rules:

$$\text{If } A_t \text{ then } B_t$$

Characterisation of the inference

The inference technique is based on product-sum-gravity [9].

Product: This step yields the contribution of the consequent sets to the output according to the degree of matching among the observation and the antecedent sets. The product has a role in the multi-variable case. Therefore the contribution is:

$$\mu_{A_t}(x^*).$$

Sum-gravity: This step follows the center of gravity defuzzification technique. All consequent sets B_t are weighted by its corresponding contribution $\mu_{A_t}(x^*)$. The defuzzificated value is the weight-point:

$$y^* = \frac{\sum_{t=1}^{m} \mu_{A_t}(x^*) b_t}{\sum_{t=1}^{m} \mu_{A_t}(x^*)}$$

The *Ruspini*-partition of the antecedent sets implies that $\sum_{t=1}^{m} \mu_{A_t}(x^*) = 1$, hence the output value is:

$$y^* = \sum_{t=1}^{m} \mu_{A_t}(x^*) b_t$$

Appendix 2

Singular Value Decomposition Reduction (SVDR)

Suppose that matrix $\underline{\underline{\mathbf{B}}}_{(n_1 \times n_2)} = [b_{i,j}]$ is given. Applying singular value decomposition yields:

$$\underline{\underline{\mathbf{B}}} = \underline{\underline{\mathbf{U}}}_1 \underline{\underline{\mathbf{D}}}\ \underline{\underline{\mathbf{U}}}_2{}^{\mathrm{T}} = [\underline{\underline{\mathbf{U}}}_1^{\mathrm{r}} \mid \underline{\underline{\mathbf{U}}}_1^{\mathrm{d}}] \begin{bmatrix} \underline{\underline{\mathbf{D}}}^{\mathrm{r}} & \underline{\underline{\mathbf{O}}} \\ \underline{\underline{\mathbf{O}}} & \underline{\underline{\mathbf{D}}}^{\mathrm{d}} \end{bmatrix} [\underline{\underline{\mathbf{U}}}_2^{\mathrm{r}} \mid \underline{\underline{\mathbf{U}}}_2^{\mathrm{d}}]^{\mathrm{T}}.$$

The elements of matrix $\underline{\underline{\mathbf{O}}}$ are zeros. Matrices $\underline{\underline{\mathbf{U}}}_1$ and $\underline{\underline{\mathbf{U}}}_2$ are orthogonal. Matrix $\underline{\underline{\mathbf{D}}}$ contains the singular values in decreasing magnitude, as the diagonal elements. The zero or the smallest of singular values (smaller than singular value threshold T_0, say) can be discarded to yield a simpler system. Let $\underline{\underline{\mathbf{D}}}^{\mathrm{r}}$ contain the retained and $\underline{\underline{\mathbf{D}}}^{\mathrm{d}}$ contain the discarded singular values. Let the result of SVDR be:

$$\hat{\underline{\underline{\mathbf{B}}}} = \underline{\underline{\mathbf{U}}}_1^{\mathrm{r}} \underline{\underline{\mathbf{D}}}^{\mathrm{r}} \underline{\underline{\mathbf{U}}}_2^{\mathrm{r}\,\mathrm{T}}$$

If B^d contains only zero singular values, then $\hat{\underline{\underline{\mathbf{B}}}} = \underline{\underline{\mathbf{B}}}$.

If $\underline{\underline{\mathbf{B}}}^d$ contains nonzero singular values, then $\hat{\underline{\underline{\mathbf{B}}}}$ is an approximation of $\underline{\underline{\mathbf{B}}}$, and the maximal difference between the values of $\underline{\underline{\mathbf{B}}}$ and $\hat{\underline{\underline{\mathbf{B}}}}$ [16]:

$$E_{SVDR} = \left| \underline{\underline{\mathbf{B}}} - \hat{\underline{\underline{\mathbf{B}}}} \right| \le \left(\sum_{i=n_r+1}^{n_{SVD}} \lambda_i \right) \underline{\underline{\mathbf{1}}}_{(n_1 \times n_2)}$$

Appendix 3

Singular Value Decomposition Extended (SVDE)

Transformation: SN (*Sum-Normalisation*). Let matrix $\hat{\underline{\underline{B}}} = \underline{\underline{U}}_1^r \underline{\underline{D}}^r \underline{\underline{U}}_2^{r\,T}$ be given. SN transforms $\underline{\underline{U}}_1^r$ $\underline{\underline{D}}^r$ and $\underline{\underline{U}}_2^r$ into $\underline{\underline{U}}_1'$ $\underline{\underline{D}}'$ and $\underline{\underline{U}}_2'$, respectively, while keeping $\hat{\underline{\underline{B}}} = \underline{\underline{U}}_1' \underline{\underline{D}}' \underline{\underline{U}}_2'^{T}$ and ensuring that the sum of all elements in each row of $\underline{\underline{U}}_1'$ and $\underline{\underline{U}}_2'$ equals 1. Let $\vec{sum}(\Phi_i) = \vec{sum}((\underline{\underline{U}}_i^r)^T)$, where $i = 1,2$ and $\vec{sum}(\underline{\underline{U}})$ is the column vector summing over the rows of $\underline{\underline{U}}$. If $\vec{sum}((\underline{\underline{U}}_i^r)^T)$ does not contain zero elements, the matrix ϕ_i is chosen as $\phi_i = diag[\vec{sum}((\underline{\underline{U}}_i^r)^T)]$. Then according to [13][15],

If $sum((\underline{\underline{U}}_i^d)^T) = 0$ then $\underline{\underline{U}}_i'$ can be determined as:

$$\underline{\underline{U}}_i' = \underline{\underline{U}}_i^r \phi_i$$

If $sum((\underline{\underline{U}}_i^d)^T) \neq 0$ then $\underline{\underline{U}}_i'$ can be determined as:

$$\underline{\underline{U}}_i' = \begin{bmatrix} \underline{\underline{U}}_i^r & \underline{\underline{U}}_i^d \vec{sum}((\underline{\underline{U}}_i^d)^T) \end{bmatrix} \begin{bmatrix} \Phi_i & 0 \\ 0 & 1 \end{bmatrix}$$

After $\underline{\underline{U}}_i'$ is determined one can always find $\underline{\underline{D}}'$ so that $\hat{\underline{\underline{B}}} = \underline{\underline{U}}_1^r \underline{\underline{D}}^r \underline{\underline{U}}_2^{r\,T} = \underline{\underline{U}}_1' \underline{\underline{D}}' \underline{\underline{U}}_2'^{T}$.

Transformation: NN (*Non-negativeness*). Let matrix $\hat{\underline{\underline{B}}} = \underline{\underline{U}}_1' \underline{\underline{D}}' \underline{\underline{U}}_2'^{T}$ be given. NN transforms $\underline{\underline{U}}_1'$ $\underline{\underline{D}}'$ and $\underline{\underline{U}}_2'$ into $\underline{\underline{U}}_1''$ $\underline{\underline{D}}''$ and $\underline{\underline{U}}_2''$, respectively, while keeping

$\hat{\underline{\underline{\mathbf{B}}}} = \underline{\underline{\mathbf{U}}}_1'' \underline{\underline{\mathbf{D}}}'' \underline{\underline{\mathbf{U}}}_2''^{\mathrm{T}}$ and ensuring that the elements of $\underline{\underline{\mathbf{U}}}_1''$ and $\underline{\underline{\mathbf{U}}}_2''$ are in the interval [0,1]. According to [13][15], let:

$$\varsigma_i = \begin{cases} 1 & if \quad a_{i\min} \geq -1 \\ \dfrac{1}{|a_{i\min}|} & \text{otherwise} \end{cases}$$

where $a_{i\,min}$ is the minimum element of $\underline{\underline{\mathbf{U}}}_i''$. Then

$$\underline{\underline{\mathbf{U}}}_i'' = \underline{\underline{\mathbf{U}}}_i'' \frac{1}{c_i + \varsigma_i} (1_{c_i \times c_i} + \varsigma_i I_{c_i \times c_i})$$

where c_i is the number of columns of $\underline{\underline{\mathbf{U}}}_i'$. With $\underline{\underline{\mathbf{U}}}_i''$ one can always find $\underline{\underline{\mathbf{D}}}''$ so that

$\hat{\underline{\underline{\mathbf{B}}}} = \underline{\underline{\mathbf{U}}}_1' \underline{\underline{\mathbf{D}}}' \underline{\underline{\mathbf{U}}}_2''^{\mathrm{T}} = \underline{\underline{\mathbf{U}}}_1'' \underline{\underline{\mathbf{D}}}'' \underline{\underline{\mathbf{U}}}_2''^{\mathrm{T}}$.

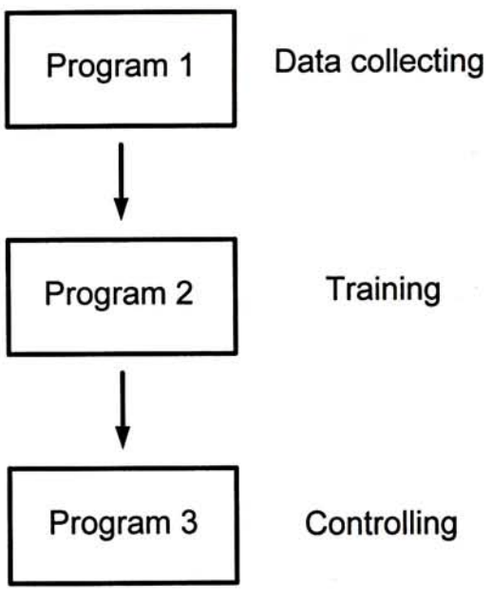
Having the above transformations, SVDR is extended (SVDE) with *non-negativeness*, *sum-normalisation*, in order to maintain the *Ruspini* partition in the case of piece-wise linear sets. SVDE results in matrices $\underline{\underline{\mathbf{A}}}_i$, where all elements are in interval [0,1] and the sum of the elements in each row is one (i.e. *Ruspini* partition).

This function includes the SVDR, SN and NN. Let $\underline{\underline{\mathbf{B}}}_{(n_1 \times n_2)}$ be a given matrix. Let $\hat{\underline{\underline{\mathbf{B}}}} = \underline{\underline{\mathbf{U}}}_1^r \underline{\underline{\mathbf{D}}}^r \underline{\underline{\mathbf{U}}}_2^{r\,\mathrm{T}}$ be the reduced form after SVDR, SN, and NN, in that order, where $\underline{\underline{\mathbf{U}}}_1^r = \underline{\underline{\mathbf{U}}}_1''$, $\underline{\underline{\mathbf{D}}}^r{}_{(n_1^r \times n_2^r)} = \underline{\underline{\mathbf{D}}}''$ $n_i^r \leq n_i : i = 1,2$ and $\underline{\underline{\mathbf{U}}}_2^r = \underline{\underline{\mathbf{U}}}_2''$.

Appendix 4

Algorithm to Implement the System

This appendix introduces the program implementation of the 2D position control system. The following block diagram shows the flow of implement of the system.



Program 1: data collecting

This program is written in C language which is collecting the 1000 input-output data sets.

```
count = 0;
_outp(Base+0, 0xff);
_outp(Base+1, 0xff);
_outp(Base+2, 0xff);
_outp(Base+4, 0xff);
delay(3000);
srand( (unsigned)time(NULL) );

for (i=0; i<1000; i++)
{
    getposition();
    if (reco == 1)
    {
        x = (x-minx)/lengthx;
        y = (y-miny)/lengthy;
        test_data[count][0] = (float)x;
```

```
        test_data[count][1] = (float)y;
        reco1 = 1;
    }
    else
    {
        reco1 = 0;
    }
    rand_n = rand();
    oe1 = floor(((float)rand_n)/4);
    oe2 = ((float)rand_n)/4;
    if (oe1 == oe2) {
        rand_n = (int)floor((((float)rand())/32767)*220);
        out0=rand_n;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out1=rand_n;
        out2=0;
        out3=0;
    }
    else if ( (oe1+0.25) == oe2 ) {
        out0=0;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out1=rand_n;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out2=rand_n;
        out3=0;
    }
    else if ( (oe1+0.5) == oe2 ) {
        out0=0;
        out1=0;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out2=rand_n;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out3=rand_n;
    }
    else if ( (oe1+0.75) == oe2 ) {
        rand_n = (int)floor((((float)rand())/32767)*220);
        out0=rand_n;
        out1=0;
        out2=0;
        rand_n = (int)floor((((float)rand())/32767)*220);
        out3=rand_n;
    }

    out0=255-out0;
    out1=255-out1;
    out2=255-out2;
    out3=255-out3;

    _outp(Base+0, out0);
    _outp(Base+1, out1);
    _outp(Base+2, out2);
    _outp(Base+4, out3);
    delay(2000);
```

```
    getposition();

    if ( (reco == 1) & (reco1 == 1) )
    {
        x = (x-minx)/lengthx;
        y = (y-miny)/lengthy;
        test_data[count][2] = (float)x;
        test_data[count][3] = (float)y;
        test_data[count][4] = (float)out0;
        test_data[count][5] = (float)out1;
        test_data[count][6] = (float)out2;
        test_data[count][7] = (float)out3;
        count++;
        printf("wire0=%d, wire1=%d, wire2=%d, wire3=%d\n",out0,out1,out2,out3);
    }
}
```

Program 2: training

This program is written in MATLAB program which is for training the GNN model of the 2 dimensional position control system.

```
load w00;
load w10;
load w20;
load w30;
load x;
load y;
load w0;

w00=w00/255;
w10=w10/255;
w20=w20/255;
w30=w30/255;

lr=0.0005;

count=0;

while (1),

load b0;

for tt=1:10,
  err=0;
  ee1=0;
  ee2=0;
  ee3=0;
  ee4=0;
  ee5=0;
  ee6=0;
  count=count+1;

  for tp=1:999,
    x00=triang21(w00(tp));
```

```
    x10=triang21(w10(tp));
    x20=triang21(w20(tp));
    x30=triang21(w30(tp));
    xx=triang21(x(tp));
    yy=triang21(y(tp));

    ans=x00*b(1,:)'+x10*b(2,:)'+x20*b(3,:)'+x30*b(4,:)'+xx*b(5,:)'+yy*b(6,:)';

    err=err+(w0(tp)-ans)^2;
    ee1=(w0(tp)-ans)*x00+ee1;
    ee2=(w0(tp)-ans)*x10+ee2;
    ee3=(w0(tp)-ans)*x20+ee3;
    ee4=(w0(tp)-ans)*x30+ee4;
    ee5=(w0(tp)-ans)*xx+ee5;
    ee6=(w0(tp)-ans)*yy+ee6;

  end

  err
  b(1,:)=b(1,:)+lr*ee1;
  b(2,:)=b(2,:)+lr*ee2;
  b(3,:)=b(3,:)+lr*ee3;
  b(4,:)=b(4,:)+lr*ee4;
  b(5,:)=b(5,:)+lr*ee5;
  b(6,:)=b(6,:)+lr*ee6;
  error(count)=err;

end

save b0 b;

end
```

Program 3: controlling

This program is written in C language which is for controlling the 2 dimensional position control system using GNN algorithm.

```
 /* tracking path */
 radius = 65;
 for (i=0; i<37; i++)
 {
     tra_pathx[i] = ((rx[0]+radius*cos(((10*i)*pi)/180))-minx)/lengthx;
     tra_pathy[i] = ((ry[0]+radius*sin(((10*i)*pi)/180))-miny)/lengthy;
 }

 out0 = 0;
 out1 = 0;
 out2 = 0;
 out3 = 0;

/*  for (count=0; count<1; count++)*/
  {
   for (cir=0; cir<37; cir++)
```

```
{
    if (cir == 0)
    {
        fprintf(fopx,"new circle \n");
        fprintf(fopy,"new circle \n");
    }

    fprintf(fopx,"%1.2f ",x);
    fprintf(fopy,"%1.2f ",y);

    app0=(float)out0/255;
    app1=(float)out1/255;
    app2=(float)out2/255;
    app3=(float)out3/255;

    xom0 = mf(app0, xom0);
    xom1 = mf(app1, xom1);
    xom2 = mf(app2, xom2);
    xom3 = mf(app3, xom3);
    xom4 = mf(tra_pathx[cir], xom4);
    xom5 = mf(tra_pathy[cir], xom5);

    app0 = 0;
    app1 = 0;
    app2 = 0;
    app3 = 0;

for (i=0; i<21; i++)
{
        app0=xom0[i]*yom0[0][i]+xom1[i]*yom0[1][i]+xom2[i]*yom0[2][i]+xom3[i]
            *yom0[3][i]+xom4[i]*yom0[4][i]+xom5[i]*yom0[5][i]+app0;
        app1=xom0[i]*yom1[0][i]+xom1[i]*yom1[1][i]+xom2[i]*yom1[2][i]+xom3[i]
            *yom1[3][i]+xom4[i]*yom1[4][i]+xom5[i]*yom1[5][i]+app1;
        app2=xom0[i]*yom2[0][i]+xom1[i]*yom2[1][i]+xom2[i]*yom2[2][i]+xom3[i]
            *yom2[3][i]+xom4[i]*yom2[4][i]+xom5[i]*yom2[5][i]+app2;
        app3=xom0[i]*yom3[0][i]+xom1[i]*yom3[1][i]+xom2[i]*yom3[2][i]+xom3[i]
            *yom3[3][i]+xom4[i]*yom3[4][i]+xom5[i]*yom3[5][i]+app3;
}

    do
    {
     out0 = (int)app0;
     out1 = (int)app1;
     out2 = (int)app2;
     out3 = (int)app3;

     if (out0<0)
     {
         out0 = 0;
     }
     if (out0>255)
     {
         out0 = 255;
```

```
    }
    if (out1<0)
    {
        out1 = 0;
    }
    if (out1>255)
    {
        out1 = 255;
    }
    if (out2<0)
    {
        out2 = 0;
    }
    if (out2>255)
    {
        out2 = 255;
    }
    if (out3<0)
    {
        out3 = 0;
    }
    if (out3>255)
    {
        out3 = 255;
    }

    _outp(Base+0, out0);
    _outp(Base+1, out1);
    _outp(Base+2, out2);
    _outp(Base+4, out3);

    delay(800);
    getposition(cir);
    x = (x-minx)/lengthx;
    y = (y-miny)/lengthy;

    printf("0=%d,1=%d,2=%d,3=%d,ex=%1.2f,ey=%1.2f,cir=%d\r",out0,out1,out2,out3,
        x-tra_pathx[cir],y-tra_pathy[cir],cir);

  } while ( (fabs(y-tra_pathy[cir]) > 0.06) | (fabs(x-tra_pathx[cir]) > 0.06) );
  printf("ex=%1.2f,ey=%1.2f,cir=%d \n",x-tra_pathx[cir],y-tra_pathy[cir],cir);
 }
}
```

BIBLIOGRAPHY

[1] MIT's Technology Review, pp. 14-15, October 1997

[2] Breazeal, C. and Scassellati, B., "A context-dependent attention system for a social robot". In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99). Stockholm, Sweden. 1146—1151, 1999

[3] Achenbach M and Muller I, "A model for shape memory", *J. Physique* 12-C4, pp.163-167, 1982

[4] Burns J and Spies R, "A numerical study of parameter sensitivities in Landau-Ginsburg models of phase transitions in shape memory alloys", *J. Intell. Mater. Syst. Struct.* 5, pp.321-332, 1994

[5] Tanaka K, "A thermomechanical sketch of shape memory effect: one dimensional tensile behavior", *Res. Mechanica* 18, pp.251-263, 1986

[6] Liang C and Rogers C, "One dimensional thermomechanical constitutive relations for shape memory materials", *J. Intell. Mater. Syst. Struct.* 1 pp.207-234, 1990

[7] Brinson L, "One-dimensional constitutive behavior of shape memory alloy: Thermomechanical derivation with non-constant material functions and redifined martensite internal variable", *J. Intell. Mater. Syst. Struct.* 4, pp.229-242, 1993

[8] K.Ikuta, M.Tsukamoto, S.Hirose, "Mathematical Model and Experimental Verification of Shape Memory Alloy for Designing Micro Actuator", *Micro Electro Mechanical Systems, 1991, MEMS '91, Proceedings. An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots. IEEE*, pp.103-108, 1991

[9] Daniel R.Madill, David Wang, "Modeling and L2-Stability of a Shape Memory Alloy Position Control System", *IEEE Transactions on Control, Systems Technology*, vol. 6, no. 4, pp.473-481, July 1998

[10] Nelly Troisfontaine, Philippe Bidaud, Maiwenn Larnicol, "Optimal Design of Micro-Actuators based on SMA Wires", *Smart Mater. Struct*, vol. 8, pp.197-203, 1999

[11] Kin-fong Lei and Yeung Yam, "Modeling and Experimentation of a Positioning System of SMA Wires", SPIE's 7th Annual International Symposium on Smart Structures and Materials, 2000

[12] Kin-fong Lei and Yeung Yam, "ANFIS Controlled SMA Wire Positioning Control System", Int. Conf. MicroCAD 2000, Miskolc, Hungary

[13] Frank P.Incropera, David P.Dewitt, *Fundamentals of Heat and Mass Transfer*, Wiley, pp.212-223, 1996

[14] J.-S. R. Jang, C.-T. Sun, E. Minzutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, pp. 335-345, 1997

[15] Wang L.X., "Fuzzy Systems are Universal Approximators", *Proceedings International Conference on Fuzzy Systems (San Diego)*, pp. 1163-1170, 1992

[16] Y.Yam , P.Baranyi and C.T.Yang, "Reduction of Fuzzy Rule Base via Singular value Decomposition", *IEEE Trans. on Fuzzy Systems*, vol. 7, no.2, ISSN 1063-6706, pp.120-131, 1999

[17] P.Baranyi and Y.Yam, "Fuzzy Rule Base Reduction" in Fuzzy IF-THEN Rules in Computational Intelligence: Theory and Applications (Eds., D.Ruan and E.E.Kerre), (Kluwer, 2000)

[18] R. Rojas, Neural Networks, A Systematic Introduction. Springer-Verlag, Berlin, 1996

[19] Hornik, K. – Stinchcombe, M. – White, H. "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks 2*, pp.359-366, 1989

[20] M.Mizumoto, "Fuzzy Controls by Product-sum-gravity Method", *Advancement of Fuzzy Theory and systems in China and Japan*, Eds. Liu and Mizumoto, International Academic Publishers, c1.1-c1.4, 1990

[21] Johansson, E.M., Dowla, F.U., Goodman, D.M., "Backpropagation Learning for Multilayer Feed-Forward Neural Networks Using the Conjugate Gradient Method", *Int. Journal of Neural Systems*, vol.2, no. 4, pp. 291-301, 1992

[22] Kin-fong Lei, Peter Baranyi and Yeung Yam, "Approximation and Complexity Reduction of the Non-singleton Based Generalised Neural Network", Technical Report TUB-TT-2000-01, Dept. Telecommunications and Telematics, Technical University of Budapest, 2000

[23] P.Baranyi, Y.Yam and C.T.Yang, "Complexity Reduction of the Rational General Form", *8th IEEE Int. Conf. On Fuzzy Systems* (FUZZ-IEEE'99), Seoul, Korea, pp.366-371, 1999

[24] P.Baranyi, Y.Yam, H.Hashimoto, P.Korondi, P.Michelberger "Approximation and Complexity Reduction of the Generalized Neural Network" Submitted to *IEEE Trans. on Fuzzy Systems*, 2000

[25] P.Baranyi, K.F.Lei and Y.Yam "Complexity Reduction of Singleton Based Neuro-fuzzy Algorithm", accepted to *IEEE SMC*, 2000

[26] Kin-fong Lei, Peter Baranyi and Yeung Yam, "Complexity Reduction of Non-singleton Based Neuro-fuzzy Algorithm", accepted in *IIZUKA*, 2000

[27] Kin-fong Lei, "Error Bound of the SVD Based Neural Network Reduction", accepted in *Int. Symposium on Smart Structures and Microsystems*, 2000

[28] Frederic I. Parke, Keith Waters, Computer Facial Animation, A K Peters, Wellesley, Massachusetts, 1996