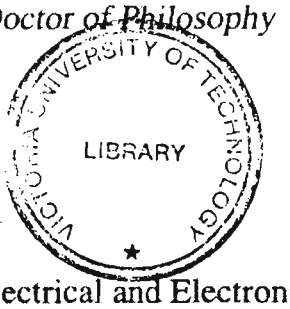
Motion learning by robot apprentices - a fuzzy neural approach

Adrian Stoica

A thesis submitted for the degree of

Doctor of Philosophy



Department of Electrical and Electronic Engineering

Faculty of Engineering

Victoria University of Technology, Australia

1995

Preface

This thesis is the result of my doctoral studies conducted under the guidance of Associate Professor Len Herron and Mr Mike Wingate as supervisors. Some of the research results presented in this thesis were included in the following papers, published in conference proceedings or currently under review for journal publication:

1. A. Stoica, On learning arm movements in behavior-based robots: towards the integration of visual and linguistic information in learning from humans. In *Proceedings of AI'93 Workshop - Machine Learning and Hybrid Systems, 6th Australian Joint Conference on Artificial Intelligence*, Melbourne, Australia, November 16, (1993), 20-24.

2. A. Stoica, Evolving creatures that can learn by imitation: apprentice behavior and its role in robot motor learning. In *International Conference on Autonomous Robots and Artificial Life PERAC'94 - From Perception to Action*, Lausanne, Switzerland, 7-9 September (1994), 440-443.

3. A. Stoica, L. Herron, M. Wingate, L. Reznik and O. Ghanayem, Fuzzy neural networks as distributed fuzzy reasoning systems. In *Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing, EUFIT'95*, Aachen, Germany, 28-30 August (1995), 86-90.

4. A. Stoica, Neurocontrol of visually guided anthropomorphic manipulators. In *Proceedings of the First International US-Mexico Conference on Neural Networks and Neurocontrol*, Quintana Roo, Mexico, 5-15 September (1995), 392-406.

5. A. Stoica and Y. Jin, Learning eye-arm coordination using fuzzy neural networks. In *Proceedings of the Joint Conference on Information Science*, Wrightsville Beach, NC, USA, 28-31 September (1995).

6. A. Stoica, On two new types of fuzzy neuron: fundamental fuzzy neuron and fuzzy neuron with shared weights. Submitted to *IEEE Transactions on Fuzzy Systems*.

Declaration

I hereby declare that this thesis is the result of my own research and has not been submitted for a degree to any other University.

Adrian Stoica
Department of Electrical and Electronic Engineering
Faculty of Engineering
Victoria University of Technology
Melbourne, Australia

Acknowledgments

I wish to express my sincere thanks to my thesis supervisors Associate Professor Len Herron and Mr Mike Wingate for offerring excellent advice and guidance, constructive criticism and stimulating encouregement throught the entire duration of my research program, and great support during writing this thesis. I am also very grateful to them for the financial assistance they have ensured.

I would like to thank Associate Professor Wally Evans, the Head of Department of Electrical and Electronic Engineering at Victoria University, for the support he gave me in all the administrative and technical problems I have approached him about, and for providing me with the financial support for attending overseas conferences. I would also like to thank Professor Ted Walker, the former Head of Department, who provided me with an invaluable help when I joined this Department.

I acknowledge and express my gratitude for receiving the following scholarships and grants: The Victoria University of Technology Postgraduate Award (1993-1995), The Secomb Award for attending Overseas Conferences (1993, 1994), and The Fellowship from Center of Cognitive Science, State University of New York, for attending the First International Summer Institute in Cognitive Science, Buffalo, N.Y. (1994).

I would like to thank Associate Professor Akhtar Kalam for helpful discussions on research strategies, and Dr Aladin Zayegh for offering me the opportunity to learn how to organize an international conference. I am very grateful to Dr Greg Cain for many discussions on the theory of computation. I wish to thank Dr Leon Reznik for many interesting discussions on fuzzy control theory and Dr Jeff Truyen for his advices on

career management. Many thanks to Ms Shirley Herrewyn for her help in multiple administrative matters, and to Mr Ralph Phillips for his assistance in preparing the equipment for experimental tests. I am grateful to Ms Zosia Golebiowski for the English language corrections she suggested to this thesis.

I thank my colleagues, postgraduate students in this Department, for the excellent cooperation we had during these years. There are some to whom I am in great debt for the help they have given me during these years. Included here is Mark Briffa, Reza Berangi, Iqbal Gondal, Xuefeng Leng, Scott Leyonhjelm, Mehrdad Salami and Mahmood Zonoozi. I am also grateful to Dr Juan Shi, who was the perfect office colleague for two years, and to Omar Ghanayem, whose optimism and friendship were very supportive in the last part of writing this thesis.

My gratitude goes to Professor Dan Butnariu of University of Haifa, for advising me on essential aspects of fuzzy theory, providing me with drafts of his latest research results and encouraging me in studying systems based on triangular-norms. I am also grateful to Professor Horia Teodorescu of Technical University of Iasi, for being my academic mentor for seven years, and for introducing me to the field of fuzzy systems.

Finally, I thank my fiance Yili for her love and help, and my brother for being my best friend. I thank my parents for all they taught me and for all they did for me. To them I dedicate this thesis.

Abstract

Futuristic scenarios feature anthropomorphic robots cooperating with humans in daily activities. Efficient cooperation requires new techniques for facilitating man-robot skill transfer. Instead of programming, it is far easier for a human to demonstrate the task, showing the robot the movements it needs to perform. This thesis presents an approach on how robots can learn the visuo-motor coordination of their arms and how they can imitate human arm movements, in order to acquire motor skills from human instructors. It is argued that in skill acquisition that involves arm movements, eye-hand coordination is not sufficient and eye-arm coordination must be developed. A method which allows the robot to learn how to move its arm while watching the human arm is proposed. The robot moves its arm to randomly chosen positions and the human places his arm in similar positions, imitating the robot. Thus the robot can make associations between images of human arm and commands given to its own arm.

Previous research on neural models has offered promising results in the learning of visuo-motor coordination, while fuzzy techniques have been successful in coping with the imprecisely defined concepts used in linguistic instruction and reasoning. The fuzzy neuron is one of the many possible neuro-fuzzy hybrids, which attempt to benefit from the synergism of qualities of neural and fuzzy models. The first part of this thesis attempts to provide a unified framework for modelling and implementing systems by using fuzzy neural networks. In particular, two new types of fuzzy neurons are proposed and analysed: the fundamental fuzzy neuron and the fuzzy neuron with shared weights. The fuzzy neural structures analysed in the first part of the thesis are used in the second part for robot learning and control. It is shown that fuzzy neural networks can be used for learning visuo-motor models, and provide certain advantages over classic neural networks. The main advantage is the transparency of the fuzzy neural models. As the robot used for tests is anthropomorphic only in a planar appearance, human imitation is demonstrated for 2D, while for 3D the robot imitates a second, identically built robot.

List of Tables

List of Figures

List of Principal Symbols and Abbreviations

Symbol or Abbreviation	Description	Defined in Section
AI	Artificial Intelligence	1.1
COAT	Compressed Output As a Threshold	6.3.1
DNF	disjunctive normal form	4.3
FFN	fundamental fuzzy neuron	3.2
FFNN	fundamental fuzzy neural networks	3.2
FN	fuzzy neuron	2.6.1
FNN	fuzzy neural network	2.6.1
FNSW	fuzzy neuron with share weights	4.1
FR	fuzzy relation	2.1
FRC1 to FRC4	fuzzy reasoning of class 1 to 4	2.3
FRE	fuzzy relational equation	2.2.1
GD	gradient descent	3.3.1
$hgt(A)$	height of fuzzy set A	2.1
inc	increment	3.31
I-O	input-output	2.3
KB	knowledge base	5.1

L	Large	3.5
M	Medium	3.5
MAX	maximum	2.2.1
MAX-MIN	type of composition	2.2.1
MAX-T	type of composition	2.2.2
MIN	minimum	2.2.1
MBMS	Maximize if Bigger Minimize if Smaller (Algorithm)	6.3.3
N	Negative	4.2
NB	Negative Big	4.2
NM	Negative Medium	4.2
NN	neural networks	2.6.1
NS	Negative Small	4.2
P	Positive	4.2
PB	Positive Big	4.2
PM	Positive Medium	4.2
PS	Positive Small	4.2
s	parameter of the fundamental t-norms	3.2
s_S	logic parameter for s-norm, somatic parameter	3.2
s_T	logic parameter for t-norm, synaptic parameter	3.2
sup	supremum	2.2.2
supp(A)	support of fuzzy set A	2.1
S	s-norm operator	App. B
SAE	sum of absolute errors	3.4.1
SSE	sum of squared errors	3.3.1
S-T	type of composition	2.2.2
S-T FN	S-T fuzzy neuron	2.6.2

S-T FNN	S-T neural network	3.1.1
S^1 to S^{12}	type of triangular co-norms	3.1.1
T	triangular norm (t-norm)	2.2.2
T^1 (to T^{12})	type of triangular norms	3.1.1
T^*	pointwise t-norm defined on the Cartesian product	4.1
VS	Very Small	3.5
VL	Very Large	3.5
Z	Zero	4.2
α	type of operation	2.2.1
α_T	T-relative pseudocomplement	2.2.2
α-composition	type of composition	2.2.1
α_T-composition	type of composition	2.2.1
αm-composition	type of composition	6.3.2
$\alpha 0$-composition	type of composition	6.3.2
$\mu_A(x)$	degree of membership of element x in fuzzy set A	2.1
ρ	learning rate	3.3.1
σ	type of operation	2.2.1
^{c}	complement of set, e.g. A^c	2.1
*	set of solutions, e.g. R^*	2.2.1
$\hat{}$	maximal solution, e.g. $\hat{R}$	2.2.1
$^{-1}$	transpose, e.g. A^{-1}	2.2.1
$^-$	not, e.g. $\bar{x}$	4.3
$\rightarrow$	logic implication	2.3
$\cup$	union	2.1
$\cap$	intersection	2.1
$\subseteq$	containment, set inclusion	2.1

$\circ$	composition, in general S-T composition	2.2.1
$\circ_t$	MAX-T composition	2.2.2
$\circ_{t,s}$	S-T composition	2.2.2
$\circ_\alpha$	α-composition	2.2.1
$\circ_{\alpha_T}$	α_T-composition	2.2.1
$\emptyset$	empty set	2.2.2
$\vee$	MAX	2.2.1
$\wedge$	MIN	2.2.1
$\Sigma - \Pi$	Sigma-Pi, a model of neuron	4.1

Contents

Appendices

Chapter 1

Introduction

1.1 Teaching by showing - learning by watching

'The long-term vision of Artificial Intelligence is to create intelligent artifacts which can learn from examples, exhibit goal directed behavior, tolerate error and ambiguity, communicate with humans in natural language, and operate in real-time or close to human response time' (from Raj Reddy's [1] plenary speech at the AAAI-94 Conference).

We are at the dawn of a new era in robotics. Engelberger [2] predicts that service robotics will outstrip industrial robotics sometime early in the 21 century. While in 1994 the industrial robot industry shipped about 65,000 robots, the prediction of the market for the elderly-care robots alone amounts to millions [Engelberger 1995].

Human friendly communication is the key of the success of the new robots. Teaching

[1] Raj Reddy, Professor and Director of the Robotics Institute at Carnegie Mellon, is Past President of American Association for Artificial Intelligence (AAAI).

[2] Joseph Engelberger is regarded by many as the 'father' of robotics. He is a founder of Unimation, the first company that produced commercial robots.

them in a similar way we teach humans to perform a task seems much more attractive than programming. If one had to utilize a programming language, or even natural language, for describing to the housekeeper robot how to stir the boiling soup, most of us would give up (or give up dinner). If one can have the robot simply watch how we do it, and then imitate our arm movements, it would be much more effective. For the imitation of human movements to be effective, anthropomorphic robot apprentices are needed.

A motor skill is an ability to perform the solution of a motion planning problem. To get this solution, the robot has the following alternatives: to find it on its own, to use precalculated, embedded solutions, or to be offered a solution. The robot can try to find a solution on its own, using reasoning or exploration. Finding a solution by reasoning may require much intelligence and may be too complex to be practically feasible. In order to decide what is optimal (from the human's point of view) the robot must understand the environment and the task. Finding a solution by exploration may be too costly. For example, *reinforcement learning* and search methods such as *genetic algorithms*, are not efficient alternatives as they take a long time and the cost of hitting an object in the workspace during a search is high [3]. Even intelligent exploration as described in [Schneider 1995] is not practical in this case. As for preprogrammed solutions, these can not consider the infinite variety of possible situations, and coding only a few solutions does not make the robots flexible enough.

A human master can offer the solution, which the robot can apply without understanding it. Using a programming language to communicate the solution isn't human friendly. Natural language instruction is human friendly, but it is not always a good way of teaching movements, which may be difficult to describe in words. A more efficient way of transmitting the movement is *analogic teaching* based on 'guiding' the robot through the movements. Analogic teaching is specially useful when the human *does*

[3]There are papers describing research in the area of discovering useful behaviors by exploration and evolutionary means, mainly with simulated systems, e.g. using genetic algorithms (see e.g. [Davidor 1991]) but also with miniature robots such as Khepera [Nolfi *et al.* 1994].

not know the exact coordinate values of a position but can see what he wants [Sheridan 1992]. The coordination of an anthropomorphic arm in a human-like movement is hard to achieve by guiding it with a teaching pendant or joystick. Acting while having attached a master arm, whose displacements are transmitted to the apprentice robot, would make it easier to accurately transmit the movement. This solution is commonly used in telemanipulation. One of the most advanced telemanipulation systems is the Utah arm, illustrated in Fig. 1.1 (photo from [Sheridan 1992]). Plans for future systems, such as the one from NASA illustrated in Fig. 1.2, envisage virtual reality environments and sensors attached on human joints for gesture tracking [Sheridan 1992]. In these systems communication means are necessary between the human and the robot, to transmit the joint information signals. The transmission of information can not be avoided in teleoperation, however, for teaching a robot in our proximity, it is sufficient to give him vision. The robot can watch the human arm and imitate its consequent postures. No human attached sensors are needed, and the human can move his arm unconstrained by any physical mechanism. The image of the arm can also be directly correlated to the task. This 'teaching by showing - learning by watching' approach is schematized in Fig. 1.3.

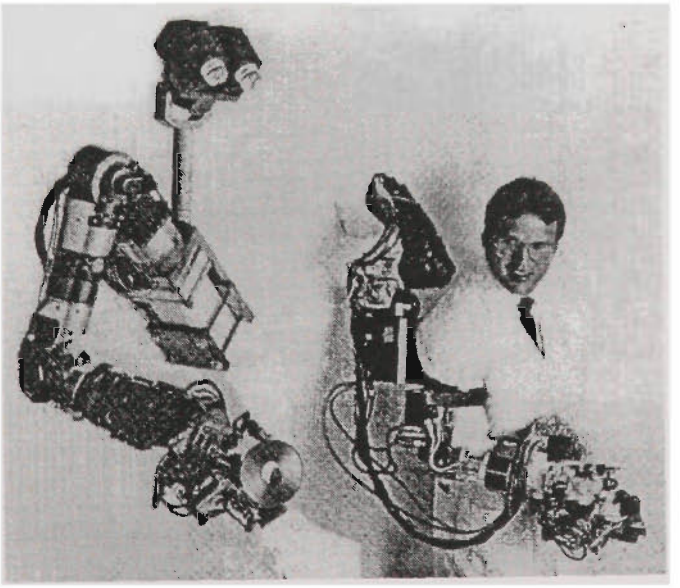


Figure 1.1: The Utah arm

From a learning perspective this approach enters largely the category of 'learning by watching', 'programming by demonstration', or more general, 'learning from

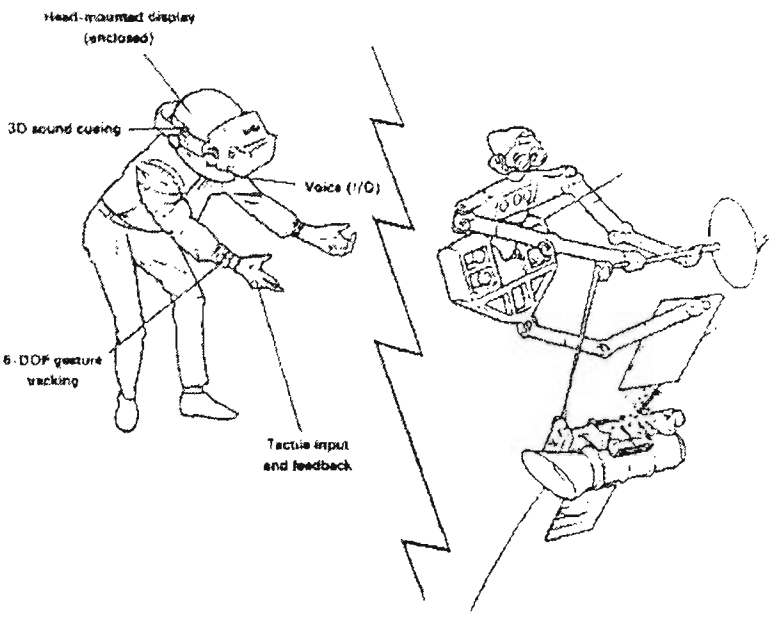


Figure 1.2: Virtual reality for telemanipulation

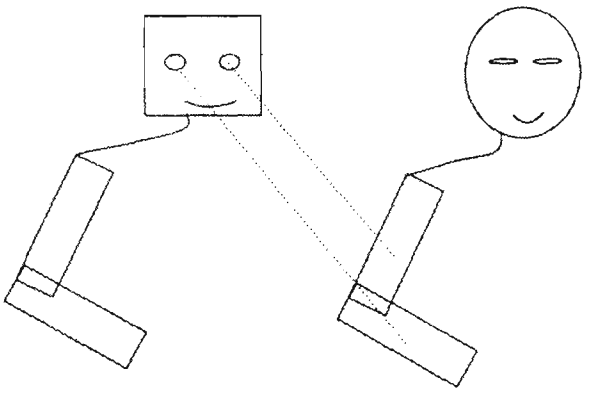


Figure 1.3: Teaching by showing - learning by watching

examples'[4][5].

The research performed in the area of skill transfer has targeted mainly *task learning*, i.e. *what to do*. However, it is also useful to watch *how to do*, and copy/learn the demonstrator's *arm* movement during the task, as this provides a model of movement which is a solution for the robot arm movement problem related to the task. In fact, in many cases, such as teaching tool handling or coaching in sports like tennis or golf, the human learner does observe the instructor's arm and tries to imitate its movements.

In order to imitate the movements of the human arm, the robot has to have the necessary coordination for transforming images of the human arm into commands for its arm. Such visuo-motor coordination can be designed or learned. The design can be cumbersome, as it is hard to predict all aspects of the real environment and cope with them in advance. For this reason, and also for the purpose of creating 'grounded representations' [Harnad 1990] that can help in further cognitive-oriented abilities, the preferred option might be to build sensory-motor models by learning [6].

The most successful results in the development of visuo-motor coordination are based on neural networks. Along with the advantages of being good function approximators, and powerful learning structures, neural networks bring the less desired feature of having solutions difficult to interpet. Fuzzy modelling seems a good complementary technique,

[4]There has recently been a growing interest in this area, reflected, for example, in the number of 1995 AAAI symposia on topics such as Programming by Demonstration, Active Learning or Agent Learning.

[5]Inductive Learning from Examples (LfE) is a well established subject in Machine Learning. 'Pure' LfE is performed automatically without any human interaction. Programming by Demonstration (PbD), on the other hand, can be seen as some kind of 'extreme' form of user-supported LfE where the user continually interacts with a PbD system. Its nearly exclusive focus is the learning of programs (adapted after the Call for Papers of a 1995 Programming by Demonstration Workshop).

[6]Traditional AI was criticized (e.g. by Searle in his Chinese Room argument [Searle 1980]) that it relies on symbols and representations arbitrarily interpretable, which have no 'intrinsic' meaning. To overcome this, and provide meaning by grounding representations in the physical world, it was proposed to link intelligence to the the sensory world through visual systems and robots.

as the knowledge in fuzzy systems is structured (usually in linguistic form) and easy to understand. Thus, neuro-fuzzy hybrids appear as an attractive path to explore. The fuzzy neuron is a special type of neuron, defined using a family of operators used in the fuzzy theory. Systems of fuzzy neurons share properties of fuzzy systems and of neural networks. Thus, fuzzy neurons may be a powerful elementary computational unit to implement in hardware. However, a theory of fuzzy neural modelling must be developed first. Fuzzy neurons and classic neurons need to be compared. Fuzzy neurons need to be compared among themselves, on the basis of different possible fuzzy operators which define them. These are motivations for the research on fuzzy neural networks presented here.

1.2 Thesis statement and main contributions

The claims of this thesis are that anthropomorphic apprentice robots can learn to visually coordinate their arms and consequently acquire motor skills by the imitation of human instructor's arm movements, and that fuzzy neural networks offer advantages over classic neural networks in building learning structures for such robots.

The work presented here solves some problems of fuzzy neural networks theory. Firstly, it identifies a family of triangular norms which is most suited for implementing neural operators in fuzzy neurons, and defines the fundamental fuzzy neuron, which uses this family. Some advantages of using the S-T composition for system modelling are shown, while noting that the problem of identification (resolution of S-T fuzzy relational equations) is unsolved. The equations for learning in fundamental fuzzy neurons derived here provide a numerical resolution method for this problem. It is shown that better models can be obtained by allowing an adaptive S-T composition. The study also provides a solution to the problem of implementing multi-input systems using fuzzy neurons, proposing a model of fuzzy neurons with shared weights.

Some initial results comparing classic [7] and fuzzy neurons have been arrived at. An interesting conclusion of this study is that synaptic adaptation can be used for obtaining better fuzzy neural models, while the most used form of classic neuron is insensitive to such adaptation. The most interesting aspect of the comparison of classic and fuzzy neural models is that fuzzy neural models are transparent. This means that the user is able to understand how the outputs are determined, and is also able to predict the behavior of the system when presented unseen inputs. Because they lack this property, classical neural networks have not been able to penetrate safety-critical areas of application. Although an immediate explanation of the transparency of fuzzy neural models comes from their relational structure, further investigation is needed to evaluate whether this transparency is a general property of fuzzy neural models or it is only a characteristic of particular cases.

Analytical resolution methods for fuzzy relational equations allow incremental on-line learning in fuzzy neural networks, which is an advantage over classic neural models. However, the classical neural models have shown greater approximation capabilities, with their weights allowed to take values in the set of real numbers, while the weights of fuzzy neurons were limited to [0,1]. In the tests presented, an additional restriction of fuzzy neurons has been the use of excitatory inputs only.

Fuzzy neural networks are used for learning robot eye-arm coordination. It is argued that eye-hand coordination is not sufficient for motor skill transfer in (redundant) anthropomorphic robots and that eye-arm coordination is necessary. Eye-arm coordination is learned from examples in a similar associative manner as used for training ALVINN [Pomerleau 1993] for visually guided navigation. Pomerleau stated that the technique of using a neural network trained by supervised learning would not be readily applicable to controlling individual joint of robot arm for which the correct response is hard to

[7] In the context of this thesis, the term classic neuron denotes the basic model of neuron whose output is obtained by a sigmoidal function applied to the weighted sum of inputs. The model is expressed in Appendix G.

determine. This thesis presents a method which solves this problem. In the beginning the robot moves at random, and the *human imitates the robot.* Thus, the robot can make associations between commands given to the joints of its own arm and the visual images of the human arm posture. Once a model is built, images of the human arm generate correspondent commands to the robot arm, and the robot imitates the human.

Methods and theoretical constructions proposed in this thesis have been validated by practical tests. The robot needed under five minutes of on-line incremental learning for building the visuo-motor coordination necessary for tracking a human arm in a planar movement. Successful tests were also performed in 3D where the robot imitated an identical robot.

As the system can track the human arm close to real time [8] it can be considered as a visual servoing system developed through learning and used as a telemanipulator [9], a first step towards what may be called *visual servoing based telemanipulation.* In this work the solutions obtained by imitation have been simply stored for later repetition of the movement. However, in future systems, more efficient encoding schemes should be used for the creation of motor patterns.

[8] As the current implementation is not multi-tasking, visual sampling occurs only when the robot has finished the movement to the previously detected target - this causes an error between the reference and actual performance, which is characteristic to sampling systems. The processing time is not significant, the delay between a detected position and the moment it gets there being limited by the speed of the manipulator, and in general was less than a second (time in which the target has normally changed position).

[9] It can be also used in virtual reality systems or for micro-teleoperation, controlling micro-arms for micro-technology manufacturing (as the arm must be with similar appearance but can be at different scale).

1.3 Organization of thesis

This thesis is organized in two parts. The first part is dedicated to the development of the theoretical aspects of fuzzy neural modelling, which is used in the second part for robot learning and control.

Chapter 2 puts together results which form the basis of the theory of fuzzy neural modelling. Its main components are the relational approach to system modelling, fuzzy relational equations and their composition, and fuzzy neurons performing pointwise composition.

Chapter 3 compares different triangular norms to asses their suitability for the implementation of neural operators. The fundamental triangular norms are the pair found most suitable, and accordingly the fundamental fuzzy neuron is defined. Learning and adaptation mechanisms in fundamental fuzzy neural networks are subsequently established. Learning also provides a numerical solution to the unsolved problem of resolution of S-T fuzzy relational equations. It is shown that adaptive composition leads to better modelling. The 'rules in weights' perspective is introduced, showing that the weights of fuzzy neurons shape structures readily interpretable by humans and which look like distributed rule-bases.

Chapter 4 solves the problem of neural implementation of multi-input distributed associative fuzzy systems by defining a fuzzy neuron with shared weights. Learning mechanisms and the organisation of weights in distributed rule-like clusters are shown for this type of neurons. It is also proved that one neuron can implement any boolean logic function, and has a flexibility which recommends it as a possible general purpose computational element.

The second part of the thesis initiates research in anthropomorphic robot apprentices,

the focus being on developing the capability of imitating human arm movements.

Chapter 5 starts with a review of research on skill transfer, visuo-motor coordination, motor development and other areas related to building robot apprentices able to learn motor skills. In what follows an approach to learning arm movements in anthropomorphic robots is proposed. It introduces the eye-arm coordination and a 'human imitates robot' method, by which the robot learns to coordinate its own arm while looking at the human arm. Once the eye-arm coordination is learned, the robot can imitate the arm movements performed by the human carrying out a particular task. This Chapter also describes the experimental setup used for testing the approach.

Chapter 6 presents an implementation of the approach and the results obtained in tests that enable robots to learn arm movements by imitation. Neural solutions and fuzzy neural models of eye-arm coordination are investigated. For fuzzy neural models incremental on-line learning is shown based on analytical algorithms of solving associated fuzzy relational equations. Interpretations of the neural weights are presented. It is shown that the neural models allowed the tested robot (which was anthropomorphic only in its planar performance, as seen from the top) to imitate 2D arm movements of the human arm. Tests of 3D arm movements were accomplished using a second, identically built robot.

Chapter 7 summarises the contributions this research has made to the fields of fuzzy systems and robotics, and indicates possible areas of future work.

Part I

Fuzzy neural networks in system modelling

This part attempts to provide a unified framework for modelling and implementing systems by using fuzzy neural networks. The neural structures developed here are used in the second part for robot learning and control. The reason for focusing on fuzzy neural networks is the desire to obtain a unique structure. and a unique basic computing element - the fuzzy neuron, which can perform perception-related processing (in which neural networks outperform alternative techniques) and also reasoning based on linguistic knowledge (which were successfully approached by fuzzy logic means). In the case of learning in robots, these relate to learning eye-arm coordination and learning from shown examples, and to learning from linguistic instructions on how to perform movements.

Results from various parts of fuzzy theory are put together to set the basis of fuzzy neural modelling. In order to facilitate the application of neural learning mechanisms, a fundamental fuzzy neuron is proposed, defined on the basis of a set of fuzzy operators found suitable for neural adaptation and learning. It is shown that fuzzy neural networks allow the understanding of their internal representations. A fuzzy neuron with shared weights is proposed to ensure the implementation of multi-input mappings. Its features recommend it as a competitive general purpose computational element.

Chapter 2

Fuzzy relations as basis of fuzzy neural modelling

The aim of this chapter is to provide an introduction to a proposed theory of fuzzy neural modelling. Central to such a theory are the relational approach to system modelling, fuzzy relational equations and their composition, and fuzzy neurons performing the fuzzy composition in a pointwise mode.

The first section introduces basic concepts of fuzzy sets and fuzzy relations. The composition of fuzzy relations leads to fuzzy relational equations, whose resolution is briefly reviewed. A fuzzy relation between inputs and outputs of a system constitutes a fuzzy relational model of the system. Accordingly, system identification is equivalent to finding a solution for the associated fuzzy relational equation. It is shown here that a largely unexplored type of composition has rich modelling capabilities. Finally, it is illustrated how a layer of fuzzy neurons implements a composition of fuzzy relations. This indicates that it is possible to use fuzzy neural networks for system modelling.

2.1 Fuzzy sets and fuzzy relations

The concepts presented in this section were introduced by Zadeh [Zadeh 1965], [Zadeh 1973], with some formulations being taken as in [Zimmermann 1991]. Let X be a space of points (objects), with a generic element of X denoted x. A *fuzzy set (class)* A in X is characterized by a *membership (characteristic) function* $\mu_A(x)$ which associates with each point in X a real number in the interval [0,1], with the value of $\mu_A(x)$ at x representing the 'grade of membership' of x in A (see Fig. 2.1). Thus, a fuzzy set can

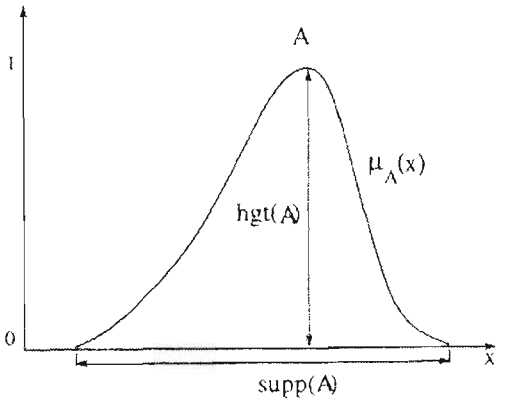


Figure 2.1: A normalized fuzzy set

be considered a set of ordered pairs

$$A = \{(x, \mu_A(x)) | x \in X, \mu_A(x) \in [0, 1]\}. \tag{2.1}$$

It is common to identify fuzzy sets with their membership functions [Di Nola *et al.* 1989]. A *support* of A is the set of points in X at which $\mu_A(x)$ is positive, $supp(A) = \{x \in X | \mu_A(x) > 0\}$. A *fuzzy singleton* is a fuzzy set whose support is a single point in X. The *height* of a fuzzy set A is the supremum of its membership function, $hgt(A) = sup_x \mu_A(x)$. A fuzzy set with hgt(A) = 1 is called *normalized.* A is *contained in* B (or equivalently, A is a *subset of* B, or A is *smaller than or equal to* B) if and only if $\mu_A \leq \mu_B$,

$$A \subseteq B \Leftrightarrow \mu_A \leq \mu_B. \tag{2.2}$$

The *complement* of a fuzzy set A is defined by $A^c(x) = 1 - A(x)$. The *intersection* of two fuzzy sets A and B is defined by $(A \cap B)(x)$ = MIN(A(x),B(x)). The *union* of

fuzzy sets A and B is defined $(A \cup B)(x)$ = MAX(A(x),B(x)). A *fuzzy logic* is defined by a set of rules for determining the complement, intersection and union of fuzzy sets. The above complement, intersection (calculated with MIN) and union (calculated with MAX) define Zadeh's fuzzy logic. Other fuzzy logics are defined later in this chapter. When operating at the level of the label of the fuzzy sets, the equivalent logic operations are NOT, AND and OR respectively.

A *fuzzy relation* R from a set X to a set Y is a fuzzy subset of the Cartesian product $X \times Y$ ($X \times Y$ is the collection of all ordered pairs (x,y) of elements $x \in X$ and $y \in Y$). R is characterized by the membership function $\mu_R(x, y)$, and is expressed by

$$R = \{((x, y), \mu_R(x, y))|(x, y) \in X \times Y\}. \tag{2.3}$$

In the fuzzy relational matrix below, elements express the strength of connections between elements of X, $X = \{x1, x2\}$, and those of Y, Y = {y1, y2 }, and can be visualized as in the associated graph of Fig. 2.2.

$$R = \begin{pmatrix} 0.8 & 0.4 \\ 0.5 & 0.3 \end{pmatrix}$$

A fuzzy relation (FR) can be seen as a mapping between two fuzzy sets described in

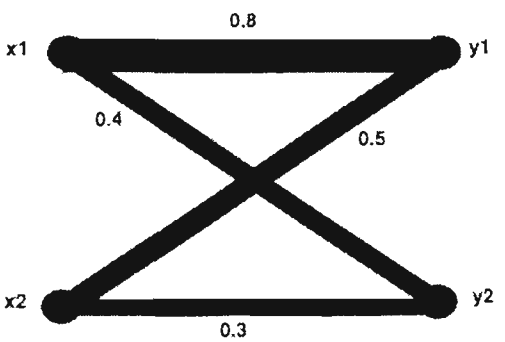


Figure 2.2: Strength of connections between elements of X and Y

terms of values of their membership function in points of their definition domains, or as a mapping of fuzzy sets represented in terms of values of membership to some *reference* fuzzy sets defined on the definition domain. The mapping of sampled membership

functions is referred in the following as *distributed*, while the mapping of labels is referred to as *compact*.

One of the first fuzzy relations discussed in the literature was that of *resemblance*. For example, the resemblance between children x1 and x2 and parents y1 and y2 can be expressed by the previous relation R, where the first row in the matrix indicates that x1 looks like y1 in a degree of 0.8, and also resembles y2 in a degree of 0.4.

Another type of fuzzy relation, which was largely exploited due to its connection with fuzzy rule-based systems, and in general with fuzzy algorithms, is that defined by a *fuzzy conditional statement* 'IF X THEN Y', or $X \rightarrow Y$, where X and Y are fuzzy sets and $\rightarrow$ is a logic implication[1]. For example, the statement 'IF the obstacle is *close* THEN move *slowly*', describes a relation R between a fuzzy variable in the antecedent part of the rule, X (*close*) and the fuzzy variable in the consequent part of the rule, Y (*slowly*), and is generally defined by the logical implication $R = X \rightarrow Y$. For instance, in the context of a robotic manipulator, a fuzzy relation R can model: a) the resemblance between a new posture of the arm (determined by a given command) and existing postures, b) the plausibility of reaching a certain state, c) the reward associated with the transition to a certain state.

2.2 Fuzzy relational equations and their resolution

2.2.1 MAX-MIN fuzzy relational equations

MAX-MIN composition. If Q is a relation from X to Y and R is a relation from Y to Z, then the *composition* [Zadeh 1973] of Q and R is a fuzzy relation denoted by $Q \circ R$

[1] A discussion on logic implication follows in Section 2.3.

and defined pointwise by

$$\mu_{(Q \circ R)}(x, z) = \bigvee_{y} (\mu_Q(x, y) \wedge \mu_R(y, z)) \tag{2.4}$$

where $\vee$ and $\wedge$ denote respectively MAX and MIN. More specific (2.4) defines the *MAX-MIN composition*. In the following a finite universe of discourse is considered, and MAX and MIN appear directly in the equations. The composition between a fuzzy set A and the fuzzy relation R, $R : X \rightarrow Y$, defines the *image* of fuzzy set A into the space Y,

$$B(y_j) = (A \circ R)(x_i, y_j) = \underset{i=1}{\overset{m}{MAX}}[A(x_i) MIN R(x_i, y_j)], \tag{2.5}$$

where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The MAX-MIN composition of fuzzy relations defined on finite universes $Q : X \rightarrow Y, R : Y \rightarrow Z$, is

$$P(x_i, z_k) = (Q \circ R)(x_i, z_k) = \underset{j=1}{\overset{n}{MAX}}[Q(x_i, y_j) MIN R(y_j, z_k)]. \tag{2.6}$$

MAX-MIN fuzzy relational equations. Let A and B be fuzzy sets, R a fuzzy relation and $\circ$ a composition (not necessarily MAX-MIN). An equation of type $A \circ R = B$ is a *fuzzy relational equation* (FRE). A FRE can be addressed in the sense of solving for R, when A and B are known, or for A, if R and B are known (in which case it is called the *inverse problem*). Equ. (2.5) describes a MAX-MIN FRE. The equation of type $Q \circ R = P$, with Q, T, R fuzzy relations, is called a *composite FRE*. As fuzzy relational matrices for Q and P are composed of rows which correspond to fuzzy sets, one can consider that (2.6) describes a system of equations of type (2.5), and therefore it is also called a *system of FRE*.

The resolution of FRE was first addressed by Sanchez [Sanchez 1976], who provided a methodology for solving MAX-MIN FRE, formulating conditions and analysing theoretical aspects of obtaining a *greatest* (or *maximal*) solution [2]. For a detailed presentation of results on MAX-MIN FRE the reader is referred to [Di Nola *et al.* 1989].

[2]That is the greatest element (in the sense of fuzzy inclusion as given by (2.2)) in the set of FR that satisfy FRE.

Conditions of solvability. The necessary and sufficient condition for Equ. (2.5) to have solutions is [3] ([Pedrycz 1994], [Sanchez 1976])

$$hgt(A) \geq hgt(B). \tag{2.7}$$

For a FRE of type (2.6), which is a system of equations of the form $Q_k \circ R = P_k$, a solution exists if and only if the solution set R_k^* of each of the k equations is nonempty, and all n solution sets intersect to a nonempty part,

$$R^* = \bigcap_{i=1}^{n} R_k^*. \tag{2.8}$$

Maximal (greatest) solution. If Equ. (2.5) admits solutions, then a maximal solution exists, which is given by

$$R(x_i, z_k) = (A \alpha B)(x_i, z_k) \tag{2.9}$$

with $a \alpha b = 1$ if $a \leq b$ and $a \alpha b = b$ if $a > b$.

If Equ. (2.6) admits solutions, then a maximal solution exists, which is given by the *α-composition* [Sanchez 1976]

$$\hat{R} = Q^{-1} \circ_\alpha P, \tag{2.10}$$

with the α-composition defined by

$$(Q \circ_\alpha R)(x_i, z_k) = \underset{j=1}{\overset{n}{MIN}} [Q(x_i, y_j) \alpha R(y_j, z_k)] \tag{2.11}$$

(Q^{-1} is the transpose of Q).

[3] In the context of a theory of plausibility (e.g. [Dubois and Prade 1988]) where membership functions are associated to plausibility of events, the interpretation of condition (2.7) is that the degree of plausibility (or certainty) of a conclusion can not be higher than the degree of plausibility of the premise from which it was inferred.

Minimal (lower) solutions. a) For the FRE given by (2.5)

If Equ. (2.5) admits solutions, then minimal solutions M exists, which can be determined by

$$M(x,y) = \begin{cases} B(y) & \text{if x} = x_y \text{ and } y \in Y_1 \\ 0 & \text{otherwise,} \end{cases} \tag{2.12}$$

where $Y_1 = \{y \in Y | B(y) > 0\}$, and x_y are arbitrary elements of $G(y) = \{x \in X | A(x) \geq B(y)\}$.

If Equ. (2.5) admits solutions, then the union of minimal solutions is given by

$$R_k = (A \sigma B)(x_i, z_j) = [A(x_i) \sigma B(z_j)], \tag{2.13}$$

where $a \sigma b = 0$ if a < b and $a \sigma b$ = b if a $\geq$ b.

b) For the composite FRE given by (2.6) (the minimal solution of the system of FRE is the minimal term in the set of solutions that satisfy all equations).

If Equ. (2.6) admits solutions, then minimal solutions exist, which are minimal elements in the set M^* of combinations of unions of minimal solutions for individual equations obtainable by (2.13).

If Equ. (2.6) admits solutions, then $\sum^* = [\bigvee_k (Q_k \sigma T_k)] \wedge \hat{R}$ ($\hat{R}$ maximal solution) is the union of all elements of M^* (this is not generally the union of the minimal solutions of (2.6)).

Solutions with minimal fuzziness. A special category of solutions is that of solutions with minimal fuzziness, i.e. which have minimal values of a fuzziness measure (see for example [Di Nola and Sessa 1983]). The membership functions of such solutions are

'polarized' towards 0 or 1, and have less elements with membership around the 0.5 value (which indicates maximum fuzziness in membership to a set).

Approximate solutions. If all equations have a solution, and their intersection is nonempty, then the system has a solution, and the presented methods of resolution can be applied. Practical situations may fail to meet the solvability conditions, in which case the system of FRE does not have (exact) solutions. In such a case, the problem of solvability can be approached in a passive way or in active ways [Pedrycz 1991*c*]. The passive approach is related to the determination of a measure of solvability of the FRE, reflected in an index of solvability. One active approach is to look at modifications of fuzzy sets, or to eliminate the equations which impede the existence of an exact solution. Another active approach is to try to fulfill the constraints *to the highest possible degree*, i.e. finding the best *approximate solution* [Pedrycz 1991*c*], [Klir and Yuan 1994]). Most common, the approximation is sought in relation to the optimization of some index factor, which most often is the fuzzy Hamming distance (a sum of absolute errors) or the Euclidian distance between desired outputs and images obtained by composition with the fuzzy relation. Numerical methods are the most common in determining approximate solutions for FRE. The problem of numerical resolution for approximate solutions was first addressed in [Pedrycz 1983*b*], where a modified Newton method was proposed. Before genetic algorithms proved their power [Sanchez 1993], [Pedrycz 1994], [Negoita *et al.* 1994], all numerical methods for finding a solution were gradient-based techniques [Pedrycz 1994].

2.2.2 Fuzzy relational equations with triangular norms

Pedrycz [Pedrycz 1983*a*] extended the MAX-MIN composition allowing MIN to be replaced by any operator from the class of *triangular norms* [Menger 1942].

Triangular norms. [Butnariu and Klement 1993]. A function T: [0,1] x [0,1] → [0,1] is called *triangular norm* (t-norm for short) if it satisfies the following conditions: associativity ($T(x, T(y,z)) = T(T(x,y),z)$), commutativity ($T(x,y) = T(y,x)$), monotonicity ($T(x,y) \leq T(x,z)$, whenever $y \leq z$) and boundary condition T(x,1) = x. Some properties which will be useful in the following sections are:

$$T(x,0) = 0, \tag{2.14}$$

$$T(x,y) \leq T(u,v), \qquad \text{whenever } x \leq u \text{ and } y \leq v. \tag{2.15}$$

A function S: [0,1] x [0,1] → [0,1] is called *triangular conorm* (t-conorm or s-norm for short) if it satisfies conditions of associativity, commutativity, monotonicity, and the boundary condition S(x,0) = x. S and T are *corresponding* (or pairs) if they comply with De Morgan's laws.

N-ary extensions. The n-ary extensions for T and S are defined recursively [Di Nola *et al.* 1989],

$$\overset{m+1}{\underset{i=1}{T}}(x_1, x_2, \ldots, x_m) = \begin{cases} T(x_1, x_2) & \text{if m = 1,} \\ T(x_{m+1}, \overset{m}{\underset{i=1}{T}}(x_1, x_2, \ldots x_m)) & \text{if m} \geq 2, \end{cases} \tag{2.16}$$

$$\overset{m+1}{\underset{i=1}{S}}(x_1, x_2, \ldots, x_m) = \begin{cases} S(x_1, x_2) & \text{if m = 1,} \\ S(x_{m+1}, \overset{m}{\underset{i=1}{S}}(x_1, x_2, \ldots x_m)) & \text{if m} \geq 2. \end{cases} \tag{2.17}$$

Examples of t-norms. MIN and MAX are the simplest pair of triangular norm/conorm. Other examples referred in this chapter are: *product/probabilistic sum* $T(x,y) = x \cdot y$, $S(x,y) = x + y - x \cdot y$, and *Yager's operators* $T(x,y) = 1 - MIN\{1, [(1-a)^p + (1-b)^p]^{1/p}\}$, $S(x,y) = MIN\{1, (a^p + b^p)^{1/p}\}, p \geq 1$.

MAX-T composition. Let A be a fuzzy set defined in X, and R a fuzzy relation between X and Y. The *MAX-T composition* of a fuzzy set A and a fuzzy relation R is the fuzzy set B defined in Y, whose membership function is given by

$$B(y_j) = (A \circ_t R)(x_i, y_j) = \underset{i=1}{\overset{m}{MAX}}[A(x_i)TR(x_i, y_j)]. \tag{2.18}$$

Let $Q : X \rightarrow Y$, $R : Y \rightarrow Z$, fuzzy relations. The *MAX-T composition of fuzzy relations* Q and R is the fuzzy relation denoted $Q \circ_t R$ between X and Z defined by

$$(Q \circ_t R)(x_i, z_k) = \underset{j=1}{\overset{n}{MAX}}[Q(x_i, y_j)TR(y_j, z_k)]. \tag{2.19}$$

Resolution of MAX-T FRE. The problems of resolution are the same as for MAX-MIN FRE and the condition of solvability is similar. If for any $y \in Y$, there exists x, $x \in X$ such that $A(x) \geq B(y)$, then Equ. (2.18) has solutions if T is continuous (this is another way of expressing (2.7)). The system of FRE (2.19) admits solutions if each equation respects these condition and the intersection of solutions to individual equations is nonempty.

If Equ. (2.18) admits solutions, then a maximal solution exists, which is given by

$$R(x_i, y_j) = (A\alpha_T B)(x_i, y_j) = A(x_i)\alpha_T B(y_j), \tag{2.20}$$

where the *T-relative pseudocomplement* α_T is defined by

$$a\alpha_T b = sup\{c \in [0, 1] : T(a, c) \leq b\}. \tag{2.21}$$

In general Equ. (2.18) does not have lower solutions [Di Nola *et al.* 1989]. The *α_T-composition* of Q and R with t-norm T is the fuzzy relation determined by

$$(Q \circ_{\alpha_T} R)(x_i, z_k) = \underset{j=1}{\overset{n}{MIN}}[Q(x_i, y_j)\alpha_T R(y_j, z_k)]. \tag{2.22}$$

The greatest element in the set of solutions of (2.19) is given by $Q^{-1} \circ_{\alpha_T} P$ [Miyakoshi and Shimbo 1985]. The t-norm needs to be lower semicontinuous, examples of α_T being given in [Di Nola *et al.* 1989]. If the system of MAX-T FRE does not have solutions, then approximate solutions may be of interest.

S-T composition. Let A be a fuzzy set defined in X ($X = \{x_1, x_2, \ldots, x_m\}$), and R a fuzzy relation between X and Y. The *S-T composition* of a fuzzy set A and a fuzzy relation R is the fuzzy set B defined in Y ($Y = \{y_1, y_2, \ldots, y_n\}$), whose membership function is given by

$$B(y_j) = (A \circ_{t,s} R)(x_i, y_j) = \underset{i=1}{\overset{m}{S}}[A(x_i)TR(x_i, y_j)]. \tag{2.23}$$

Let $Q : X \rightarrow Y$, $R : Y \rightarrow Z$, fuzzy relations. The S-T composition of Q and R is the fuzzy relation denoted $Q \circ_{s,t} R$ between X and Z defined by

$$(Q \circ_{t,s} R)(x_i, z_k) = \underset{j=1}{\overset{n}{S}}[Q(x_i, y_j)TR(y_j, z_k)]. \tag{2.24}$$

The S-T composition was introduced by Pedrycz in [Pedrycz 1983*a*] [4]. This composition received very little attention in the literature (it was not even mentioned in the most comprehensive monography on fuzzy relational equations [Di Nola *et al.* 1989] which Pedrycz also co-authored). It started being mentioned again by Pedrycz only in the context of fuzzy neurons in [Pedrycz 1992], [Pedrycz *et al.* 1995] and without emphasizing any of its advantages. Independent of Pedrycz work this composition was proposed again in a 1988 French publication ([Bour and Lamotte 1988]).

Resolution of S-T FRE. Conditions of solvability and an algorithm for determining a solution for one FRE given by Equ. (2.23) were presented in [Bour and Lamotte 1988]. For a solution of 2.23 to exist, it is necessary that

$$\forall k \in K, \qquad B(y_k) \leq \underset{j \in J}{S}(A(x_j)) \tag{2.25}$$

where $J = \{1, \ldots, n\}$, $K = \{1, \ldots, p\}$. The condition is sufficient provided S and T are continuous. In order to determine a solution for (2.23) the following algorithm

[4] Pedrycz [Pedrycz 1983*a*] names it $max_q - min_p$ ($Y = X \Diamond_{p,q} R$) where max_q stands for s-norm and min_p stands for t-norm.

was proposed [Bour and Lamotte 1988] (which assumes the existence of a T-relative pseudocomplement (2.21). Denote $a_j = A(x_j)$, $b_k = B(y_k)$, $J_1(k) = \{j \in J | b_k \leq a_j\}$.

- If $J_1(k) \neq \emptyset$, $\forall k \in K$ (a condition similar to the resolution of MAX-T FRE, i.e. it exists a $j'(k)$, $a_{j'(k)} \geq b_k$)

$$r_{j,k} = \begin{cases} a_j \alpha_T b_k & \text{if } j = j' \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

- If exists a $k \in K$, for which $J_1(k) = \emptyset$ (i.e. $\underset{k}{MAX} b_k > \underset{j}{MAX} a_j$), then it exists an index j' for which $S(a_1, \ldots, a_{j'-1}, 0, \ldots, 0) < b_k$ and $S(a_1, \ldots, a_{j'}, 0, \ldots, 0) > b_k$. Denote a' the value of an element in position j' for which the equality holds, i.e. $S(a_1, \ldots, a', 0, \ldots, 0) = b_k$. A solution of (2.23) is given by

$$r_{j,k} = \begin{cases} 1 & \text{if } j < j' \\ a' \alpha_T b_k & \text{if } j = j' \\ 0 & j > j' \end{cases} \tag{2.27}$$

Unfortunately, this algorithm can not be extended to a system of FRE of the form (2.24), for which the problem of resolution is open.

2.3 The relational approach to system modelling

Fuzzy systems are mappings of fuzzy sets, generally expressed as collections of 'pieces' of knowledge of the form X_k *maps to* Y_k, where X_k and Y_k are fuzzy sets. For a particular input, crisp or fuzzy, which is not exactly specified in an expressed mapping, the output may be obtained by employing some form of *fuzzy reasoning*. The standard way of calculating the output is by using a compositional rule of inference of the form $Y_k = X_k \circ R$, where R is a fuzzy relation describing the mapping.

In the following I propose a grouping of methods of fuzzy reasoning in classes. The grouping considers the type of representation of the elements that enter the fuzzy relation and the semantic of the fuzzy relation. The representation can be compact or distributed (as defined in Section 2.1). The semantic aspect refers to the distinction made between conditional and associative flavours of fuzzy relations. The possible classes are shown in Table 2.1.

Table 2.1: Classes of fuzzy reasoning

	Distributed	Compact
Conditional	FRC1	FRC2
Associative	FRC4	FRC3

Fuzzy reasoning of class 1 (FRC1) is distributed and conditional. The fuzzy relation is between sampled fuzzy sets, and is calculated by some form of implication. Fuzzy reasoning of class 2 (FRC2) is compact and conditional. The inputs match input fuzzy sets in some degree. These degrees of matching to various inputs are propagated to the output by some form of implication. Fuzzy reasoning of class 3 (FRC3) is compact and associative. The fuzzy relation expresses a mapping between labels of fuzzy sets, and the fuzzy relation is calculated by identification from input-output (I-O) pairs. Fuzzy reasoning of class 4 (FRC4) is distributed and associative. The fuzzy relation expresses a mapping between sampled fuzzy sets, and is calculated by identification from I-O pairs.

Conditional reasoning is generally associated with the most popular form of fuzzy systems, i.e. fuzzy logic controllers. In fuzzy logic controllers the knowledge of mapping consists of a set of rules presented as fuzzy conditional statement of the form *If* x *is* X_k *then* y *is* Y_k. In FRC1 the fuzzy relation is between sampled fuzzy sets and models an implication, and the reasoning is considered a generalization of *modus ponens* case of propositional logic [Zadeh 1973]. The implications can be calculated for each rule as $R_k = (X_k \rightarrow Y_k)$, and a partial output can be obtained by composition of input with each implication. The total output is calculated as a union of individual contributions.

For MAX-T composition, the same result is obtained if a union of all implications is calculated first, and the input is composed with this union. There are several ways of calculating the implication (see for example [Lee 1990]), the simplest way being by Cartesian product calculated as a pointwise t-norm operation. Various forms of fuzzy reasoning were explored by considering different t-norms in MAX-T composition, combined with different implications or different ways of aggregating the contribution of each rule (see for example [Mizumoto and Zimmermann 1982] [Mizumoto 1991])[5]. As shown for example in [Di Nola *et al.* 1989] the output obtained using this form of reasoning by implication is equivalent to the output resulting in a processing scheme with an interesting interpretation. The scheme has three steps, i.e (1) a matching step, in which the input data is matched (by intersection, generally with MIN) against the premise part of a rule and a number is returned which reflects the degree of matching, (2) an activation step, in which the conclusion part of the rule is modulated (in the way the particular implication dictates, e.g. by MIN in Mamdani type of reasoning [Mamdani and Assilian 1975]) with the degree of matching obtained in previous step, and (3) a combination step in which the contributions of all 'fired' rules are combined together by a union operator, e.g. MAX (a rule was 'fired' if the input matched its premise in a nonzero degree). This scheme of processing became so popular that few ever mention its relational roots. This is the compact conditional reasoning referred here as FRC2, and mentioned again in the discussion on hardware implementations of fuzzy models.

In an alternative approach to system modelling, the conditional issue is not stressed, but rather the mappings are seen as associative, i.e. X_k is *associated with* Y_k, or X_k is *related to* Y_k, or X_k is *similar to* Y_k. In this case, R is seen as a solution for the fuzzy relational equation $X \circ R = Y$ (and not calculated by some implication formula) using methods of resolution indicated in Section 2.2. Because in the most used scheme of

[5]The combination composition - implication does not always respect modus ponens for known mappings, i.e. $X_k \circ (X_k \rightarrow Y_k) \neq Y_k$ [Keller and Tahani 1992]. One example is the combination of MAX-MIN composition with Lukasiewicz implication $a \rightarrow b = MIN(1, 1 - a + b)$. However the same implication with the composition MAX-DP (DP stands for drastic product, defined as T^5 in Table 3.1) does satisfy modus ponens.

reasoning (FRC2) the relational roots are not transparent, and because only the associative approach leads to solving FRE, only the associative approach is usually referred to as relational. In general, a representation in a unique fuzzy relation R is sought, which must satisfy the mapping for each k rule. Compared with FRC2, the associative approach requires additional memory for storing the fuzzy relation, but offers more flexibility in applications [Di Nola *et al.* 1989]. This approach to system modelling is strongly supported in the papers of Pedrycz and collaborators (e.g. [Pedrycz 1991*a*] [Di Nola *et al.* 1991] [Pedrycz *et al.* 1995]) who addresses mainly FRC3.

In this thesis, which investigates fuzzy neural structures, the focus is on FRC4 as the approach best fitting neural qualities, such as distributed processing and graceful degradation. FRC3 as studied by Pedrycz presents the advantage of using logic mappings at concept level, while this study shows that FRC4 uses logic mappings at lower levels.

Information processing in a fuzzy system is illustrated in Fig. 2.3, which exemplifies the mapping of three fuzzy sets. For this system and the first mapping ($X_1 \rightarrow Y_1$), FRC4 is detailed in Fig. 2.4 and illustrates the composition

$$\begin{pmatrix} 1 & 0.7 & 0.3 & 0 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 0 & 0 & 0.2 & 1 \\ 0 & 0 & 0.4 & 0.7 \\ 0 & 0 & 0.7 & 0.7 \\ 0.7 & 0 & 0.4 & 0 \\ 0.4 & 0.7 & 0.2 & 0.6 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0.4 & 1 \end{pmatrix}.$$

Here the definition domains are [0,1] (to which the real domain of inputs and outputs can be mapped), a partition $\{x_1, x_2, \ldots, x_m\}$ on the input space, and $\{y_1, y_2, \ldots, y_n\}$ on the output space, the fuzzy sets being represented in terms of their sampled degrees of membership over the discrete space, and the composition being MAX-MIN. Processing can be considered to be performed by a network of distributed processors, each calculating the output in a particular point of the output domain.

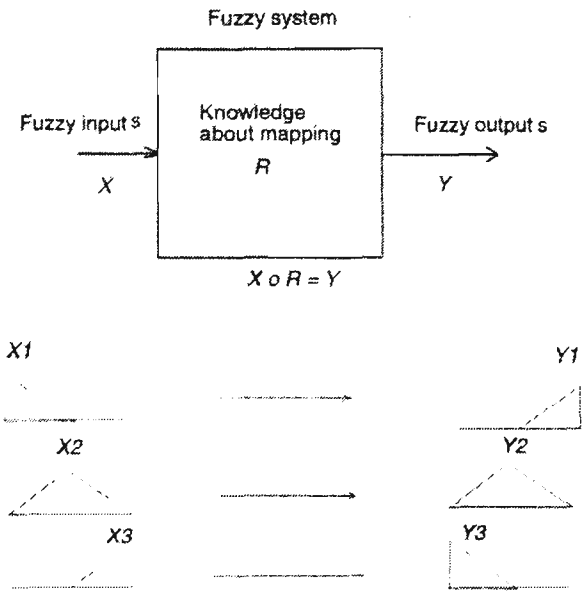


Figure 2.3: Processing in a fuzzy system: outputs are obtained by the composition of inputs with a fuzzy relation.

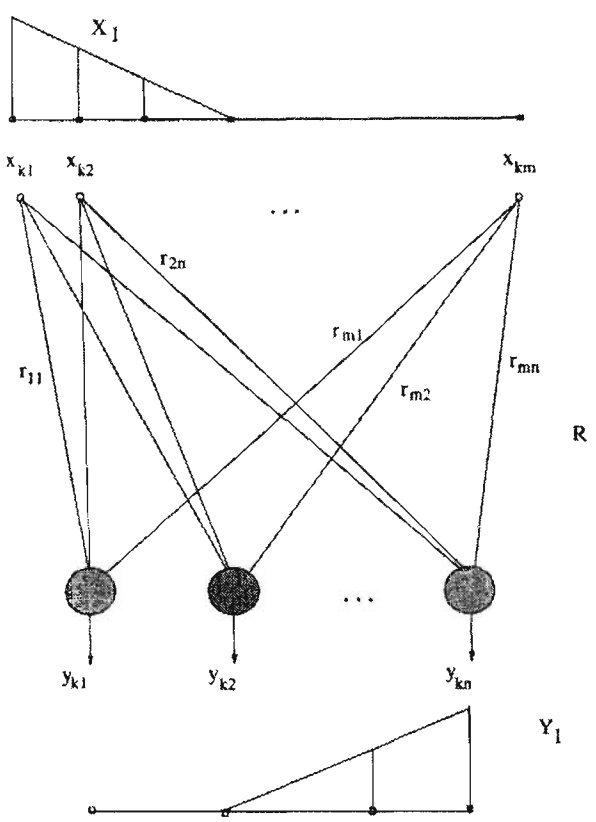


Figure 2.4: A network structure implementing FRC4

2.4 Advantages of using the S-T composition for system modelling

In practical applications, the conditional approach to system modelling has received more attention than the associative approach. The common fuzzy composition for the conditional approach is MAX-T (which includes MAX-MIN as a particular case), which is consistent with the implicational flavour of the fuzzy relation between premise and conclusion. It is reasonable to consider that the confidence in a conclusion (Y) does not exceed the confidence in the premise (X), and this is consistent with the condition of solvability for MAX-T FRE, $hgt(X) \geq hgt(Y)$. This perspective leaves little room for the interpretation of a composition that produces $hgt(Y) > hgt(X)$, and it may be one of the causes of the lack of interest in the S-T composition [6].

S-T composition is acceptable in the associative approach. For example, in the case of resemblance relations [Zadeh 1973], it is perfectly plausible to have the degree of resemblance of the resulting set higher than the degrees of resemblance reflected in composing relations. This is also common to *fuzzy similarity relations* [7] [Zadeh 1971], [Tamura *et al.* 1971] used in fuzzy clustering, for which the transitivity is stated as

$$\mu_R(x,z) \geq \underset{y}{MAX}(\mu_R(x,y) T \mu_R(y,z)). \tag{2.28}$$

One can replace the inequality (2.28) with the equality

$$\mu_R(x,z) = \underset{y}{S}(\mu_R(x,y) T \mu_R(y,z)) \tag{2.29}$$

because (from (3.27)) S_0 (MAX) is the lowest s-norm and thus

$$\underset{y}{S}(\mu_R(x,y) T \mu_R(y,z)) \geq \underset{y}{MAX}(\mu_R(x,y) T \mu_R(y,z)).$$

[6] There are other reasons, e.g. the lack of distributivity $(XSW) \circ_{s,t} R \neq (X \circ_{s,t} R) S (W \circ_{s,t} R)$ [Pedrycz 1983*a*]

[7] A *similarity* relation R in X is a fuzzy relation in X which is (a) reflexive, (b) symmetric and (c) transitive.

The advantage of this replacement is that one can actually determine the result of the fuzzy relation $\mu_R(x, z)$, once S and T are specified (for example as a result of a learning process, discussed in detail in Chapter 3).

A comparative presentation of properties of S-T composition and MAX-T is determined in Appendix 1[8]. Some advantages identified here for S-T composition are:

1. It enlarges the class of problems addressable in a relational perspective, as one can build exact fuzzy relational models of systems that have the maximal value of the output larger than the maximal value of the input (i.e. not limited by (2.7)).

2. It allows a precise specification of composition of fuzzy similarity relations.

3. When S and T are differentiable (and thus the function that maps inputs to outputs is differentiable), gradient based methods of resolution of FRE are directly implementable (MAX, in the MAX-T composition, is not differentiable). When using parametric t-norms that have MIN as a limit, and s-norms that have MAX as a limit, the solution for MAX-MIN FRE can also be obtained.

4. In many modelling cases (including the context of fuzzy neural networks) it is important to have a global cumulative effect on composing elements and not only a local effect as imposed by MAX (in which case only the maximal value dictates the output).

5. The lack of distributivity can model a memory property of a system for which the order of applying the inputs is important.

[8]For all composition presented here, *dual compositions* are defined in the literature (see for example [Di Nola *et al.* 1989]) by replacing MAX with MIN, and the t-norm with the associated s-norm. For example, MIN-MAX is dual to MAX-MIN, and MIN-S is dual to MAX-T. One can define also the general T-S composition as a dual to S-T composition, and as a particular case S-MIN or T-MAX, which are distributive.

An example of a situation that needs the S-T composition for appropriate modelling is given in the following. Consider a set of learned postures of a robot arm to be $q = \{q_1, q_2\}$ consisting of two postures whose spatial placement can be linguistically described in a set of terms $l = \{left, front, right\}$ (see Fig. 2.5). One can linguistically

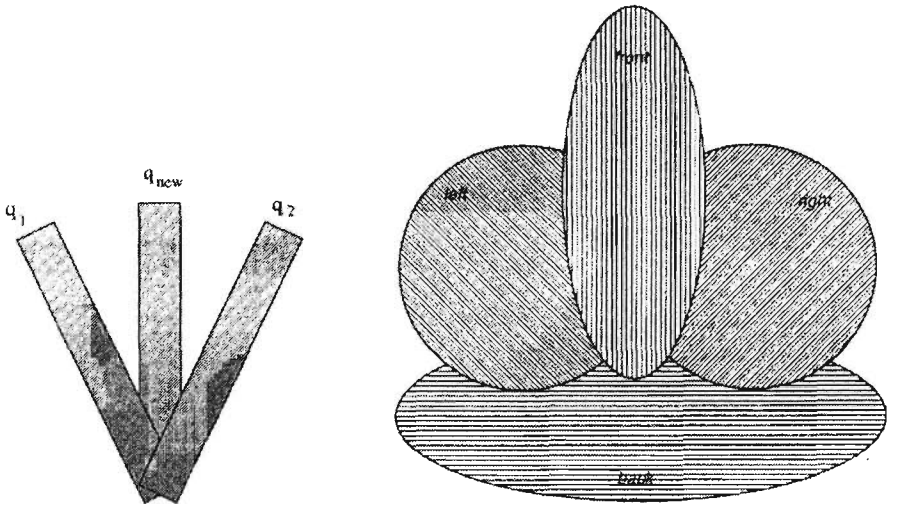


Figure 2.5: Arm postures and a linguistic description of the space

characterize the first posture by $W(q_1, l) = \{0.3/left, 0.8/front, 0/right\}$ and the second posture by $W(q_2, l) = \{0/left, 0.8/front, 0.3/right\}$. A new command to the arm can place it in the new posture q_{new}, in between the two and resembling them each in a degree of 0.8, thus $X(q_{new}, q) = \{0.8/q_1, 0.8/q_2\}$. The linguistic description of the new posture would be given by the composition

$$Y(qnew, l) = X(qnew, q) \circ W(q, l) = \begin{pmatrix} 0.8 & 0.8 \end{pmatrix} \circ \begin{pmatrix} 0.3 & 0.8 & 0 \\ 0 & 0.8 & 0.3 \end{pmatrix}.$$

If the $\circ$ composition is MAX-MIN composition then $Y(qnew, l) = (0.3, 0.8, 0.3)$, i.e. $\{0.3/left, 0.8/front, 0.3/right\}$. No MAX-T composition can give more than $0.8/front$. As the drawing shows however, the new posture is $1/front$. This can be modeled by an S-T composition, e.g. for $S^{12}_{s=10^6}, T^{12}_{s=10^6}$ (S^{12} and T^{12} are given by formulas in Table 3.1), one obtains the description $Y(qnew, l) = (0.1, 1, 0.1)$, i.e. $\{0.1/left, 1/front, 0.1/right\}$ (*left* and *right* overlap with *front*).

2.5 Fuzzy logics

Zadeh's fuzzy logic is defined based on MIN and MAX operations for intersection and union. A more general case is that of fuzzy logics based on triangular norms [9], T and S, for which the complement A^c of a fuzzy set A is defined by

$$A^c(x) = 1 - A(x), \tag{2.30}$$

the intersection ATB of fuzzy sets A and B is defined by

$$(ATB)(x) = T(A(x), B(x)), \tag{2.31}$$

and the union ASB of fuzzy sets A and B is defined by

$$(ASB)(x) = S(A(x), B(x)). \tag{2.32}$$

In the following, the specification of certain fuzzy logic is semantically equivalent to the specification of an associate composition, and vice-versa. For example, 'applying the MAX-MIN composition' is equivalent to 'working under the MAX-MIN (S_0, T_0) logic'.

[9] This paragraph is adapted from [Butnariu *et al.* 1995], where fuzzy logics are defined based on intersection and complement only (considering complementary t-norms).

2.6 Fuzzy neural modelling

2.6.1 A brief account of approaches combining fuzzy and neural elements

Fuzzy models have proved their usefulness in a variety of applications (for some recent accounts see [Yager and Filev 1994], [Yager and Zadeh 1994]). An equally powerful modelling technique is that of neural networks (NN). These two techniques have a degree of complementing each other, which may be exploited in structures that combine the individual strengths.

Combinations of elements of fuzzy logic and neural networks were initiated in the seventies, a first model of *fuzzy neuron* (FN) and a *fuzzy neural network* (FNN) being introduced in [Lee and Lee 1975]. In the late eighties, stimulated by the revival of interest in neural networks research, and mainly using the backpropagation algorithm [Rumelhart *et al.* 1986], several fuzzy-neuro or neuro-fuzzy approaches have been proposed[10]. Work in this interdomain sector is mainly driven by the aim of combining the powerful features of each paradigm, the transparent knowledge embodied in fuzzy systems and the learning ability of NN. For a review of work in this area the reader is referred to [Takagi 1990], [Hellendoorn 1994], [Jang 1995].

In this account I refer only to those alternatives of mapping fuzzy processing into neural structures which are patterned after the approaches to fuzzy modelling discussed[11].

The first alternative is to use hierarchical structures in which layers in the networks

[10]In general, fuzzy neuro refers to structures using fuzzy neurons, and neuro-fuzzy refers to fuzzy systems implemented with classic neurons

[11]A synthesized overview on neural implementations of fuzzy systems is presented in Appendix B. The details include the topology of the network, the type of neurons used, and training related aspects.

implement steps of processing information as in the compact reasoning schemes (FRC2 and FRC3). In such neural architectures (e.g. [Takagi *et al.* 1992] for FRC2, [Pedrycz and Rocha 1993] for FRC3) some neurons are associated with linguistic values, and fire in the degree in which the linguistic value is matched by the input, while other neurons are associated with rules, and fire in the degree in which the rule is fired. The architecture of the network reflects the structure of knowledge. This is what I call here '*the rules in neurons*' approach. It is not a distributed representation as it does not allow 'graceful degradation' (meaning that the removal of one neuron should not greatly affect the result of processing). Learning in networks implementing FRC2 aims at finding appropriate membership functions and rules, in the context of a fixed reasoning method (often Sugeno's). Given the membership functions and the way they map (rules), Keller and Krishnapuram[Keller and Krishnapuram 1992] investigate which parameters of reasoning are the best suited ones for the mapping. In the case of FRC3 the network is considered to form a cognitive map between concepts [Pedrycz *et al.* 1995]. The weights of the neurons in the network are elements of the fuzzy relation modelling the system, and system identification is equivalent to learning by the neural network of a solution of a fuzzy relational equation [Pedrycz 1990*b*], [Pedrycz 1991*b*], [Pedrycz 1991*c*], [Pedrycz and Rocha 1993].

The second alternative of mapping fuzzy processing into neural structures is to have a structure of neurons which directly reflects the distributed mapping between discretized inputs and discretized outputs. This structure can be multilayer, and was proposed to employ classic neurons [Keller and Tahani 1992], or fuzzy neurons [Keller and Krishnapuram 1992]. This thesis investigates unilayer implementations of fuzzy systems, using fuzzy neurons which perform the relational mapping between discretized inputs and discretized outputs. Since Lee's first model of fuzzy neuron [Lee and Lee 1975], different types of neurons incorporating elements of fuzzy theory were defined, ranging from various hybrid structures as in [Pedrycz 1991*b*], [Pedrycz 1992], [Goh and Lui 1991], [Kwan and Cai 1994] to implementations of complete fuzzy inference systems

in one neuron [Yamakawa *et al.* 1992]. In particular, fuzzy logic neurons were defined [Gupta 1992], employing fuzzy logic operators to model *synaptic* and *somatic*activities [12]. For example, a MAX-MIN fuzzy neuron (or simply MAX-MIN neuron) was defined based on Zadeh's fuzzy logic, i.e. using MAX as union of activities at somatic level, and MIN as intersection or joint effect of inputs and synaptic states [Pedrycz 1990*b*], [Saito and Mukaidono 1992]. Pedrycz [Pedrycz 1990*b*] has shown that a layer of MAX-MIN neurons implements the MAX-MIN composition and can be considered as the underlying structure of a fuzzy system in its relational definition (i.e. $X \circ R = Y$ where X are input fuzzy sets and Y are outputs fuzzy sets, R is a fuzzy relation between input and output, and $\circ$ is the MAX-MIN composition).

Fuzzy logics other than Zadeh's MAX-MIN can be used for defining logic neurons. A general model of fuzzy logic neuron, which operates on triangular norms is discussed in the following.

2.6.2 S-T fuzzy neurons and S-T composition

Consider a neuron as an information processing element, having a number of inputs $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ and an output y. The inputs affect the neuron via synapses, which modulate the inputs with the values of the weights $\mathbf{w} = [w_1, w_2, \ldots, w_m]$. The modulation can be modeled by a t-norm operation

$$t_i = T(x_i, w_i), \tag{2.33}$$

and the effect of the modulated inputs as perceived by neuron consists in a set of t_i. All these t_i, i = 1,2,...,m, are somatic input contributions to the neuron, and their aggregation

[12]This terminology adopted from Neurobiology is used for referring to the main operations that take place in an artificial neuron model: a modulation of inputs with weights (by neural *synapses*), and a nonlinear aggregation of all weighted contributions (by neural *soma*).

in order to determine an output can be modeled by an s-norm operation

$$y = \overset{m}{\underset{i=1}{S}}(t_i). \tag{2.34}$$

Eqs. (2.33) and (2.34) define an *S-T fuzzy neuron* (S-T FN), which is illustrated in Fig. 2.6. This model of neuron was defined as such by Gupta [Gupta 1992] and Pedrycz

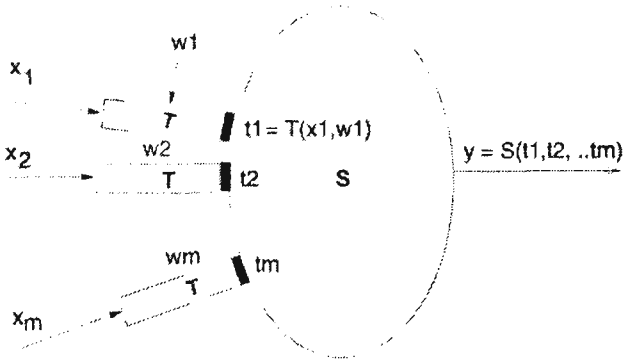


Figure 2.6: S-T fuzzy neuron

and Rocha [Pedrycz and Rocha 1993] [13].

Consider now n similar neurons, each receiving inputs x via the set of weights W = $[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$. The equations for the j-th neuron are

$$t_{i,j} = T(x_i, w_{i,j}),$$

$$y_j = \overset{m}{\underset{i=1}{S}}(t_{i,j}).$$

The output of the layer of n neurons is the vector $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ calculated with

$$\mathbf{y} = (\mathbf{x} \circ W) = S(T(\mathbf{x}, W)) \tag{2.35}$$

[13]Pedrycz addressed a rich variety of fuzzy neurons [Pedrycz 1991*b*], [Pedrycz 1992], [Pedrycz and Rocha 1993], [Hirota and Pedrycz 1994], the one defined by Equ. (2.33) and (2.34) being the *OR neuron* [Pedrycz and Rocha 1993].

which has the formalism of the S-T composition of **x** and W.

The inputs presented to the layer are a function of time. The j-th neuron at instant k is characterized by

$$t_{i,j}^{k} = T(x_{k,i}, w_{i,j}), \tag{2.36}$$

$$y_{k,j} = \underset{i=1}{\overset{m}{S}}(t_{i,j}^{k}). \tag{2.37}$$

Considering that inputs at discrete moments of time 1,2,...,p, are rows in a matrix X = [$\mathbf{x}_1$, $\mathbf{x}_2$,...,$\mathbf{x}_p$] and the same for outputs Y =[$\mathbf{y}_1$, $\mathbf{y}_2$,...,$\mathbf{y}_p$] then the detailed form of this batch processing is (see also Fig. 2.7)

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \ldots x_{1,m} \\ x_{2,1} & x_{2,2} & \ldots x_{2,m} \\ \ldots & & \\ x_{p,1} & x_{p,2} & \ldots x_{p,m} \end{pmatrix} \circ \begin{pmatrix} w_{1,1} & w_{1,2} & \ldots w_{1,n} \\ w_{2,1} & w_{2,2} & \ldots w_{2,n} \\ \ldots & & \\ w_{m,1} & x_{m,2} & \ldots x_{m,n} \end{pmatrix} = \begin{pmatrix} y_{1,1} & y_{1,2} & \ldots y_{1,n} \\ y_{2,1} & y_{2,2} & \ldots y_{2,n} \\ \ldots & & \\ y_{p,1} & y_{p,2} & \ldots y_{p,n} \end{pmatrix} \tag{2.38}$$

or in compressed form

$$Y = (X \circ W)(y_{k,j}) = \underset{i=1}{\overset{m}{S}}(T(X(k,i), W(i,j))) \tag{2.39}$$

which is the formal description of S-T composition of fuzzy relations.

The network consists of fuzzy neurons performing the S-T composition. Presented with I-O pairs, learning in such a network means the identification of a fuzzy model.

2.7 Summary

This chapter presented basic concepts of the relational approach to systems modelling. Processing in relational systems is based on the composition of fuzzy relations, and

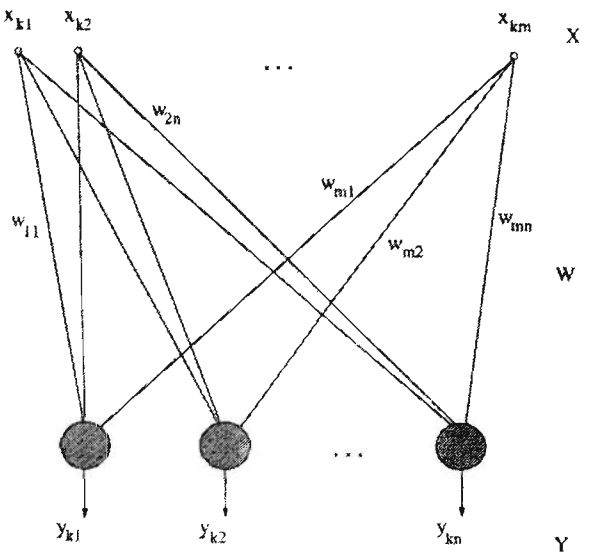


Figure 2.7: A layer of S-T neurons

systems identification is equivalent to finding a solution for a fuzzy relational equation. Analytical methods of resolution of some FRE were briefly reviewed. It was argued that the little investigated S-T composition offers some modelling advantages. S-T fuzzy neurons were shown to be elementary processors performing a pointwise S-T composition. The neural weights are elements of a fuzzy relation, and learning in S-T fuzzy neural networks can be viewed as a numerical method of resolution for corresponding FRE.

The purpose of this chapter was to introduce concepts and present results setting the basis for developing a theory of fuzzy neural systems. The fundamental elements for such a construction are the theory of fuzzy relational equations, the relational perspective to system modelling (in particular the distributed associative type of reasoning), and the definition of fuzzy neurons as computational elements performing a fuzzy composition.

Fuzzy neural structures differ from their classic neural counterparts in the type of operators employed. Network architectures and learning mechanisms used for classic neurons can be investigated in conjunction with fuzzy neurons. One question that arises is whether this plethora of techniques is applyable to fuzzy neurons employing any type

of operators, or are some operators more appropriate. Another question is what type of fuzzy neuron is most suitable for implementing multi-input mappings. These questions are the motivation for the research presented in the next chapters.

Chapter 3

Fundamental fuzzy neural networks

The Chapter introduces a new form of fuzzy neuron operating on t-norms. Firstly, different t-norms are compared in order to select the most suitable one for implementing synaptic and somatic operators for fuzzy neurons. According to a set of criteria that favours neural learning, the family of fundamental t-norms is chosen, and from this the fundamental fuzzy neuron is defined. Learning aspects in fundamental fuzzy neurons are addressed, and the equations which allow gradient-descent learning in networks of such neurons are derived. It is shown how learning can lead to a simultaneous identification of weights and synaptic/somatic parameters, which in the context of S-T FRE means the simultaneous determination of the optimal FR and fuzzy composition. This approach presents advantages in comparison with the classic case, where the search for a solution of FRE is performed under the assumptions of a chosen composition. As particular cases, solutions can be found for classic MAX-MIN and MAX-T FRE. The Chapter ends introducing the 'rules in the weights' perspective, showing that the neural weight space configures a distributed rule base (and for this purpose can be used as a rule extraction mechanism) directly interpretable by humans, allowing thus fuzzy neural

networks to alleviate the 'black-box' drawback of classic neural networks [1] [2].

3.1 Choosing the appropriate synaptic and somatic operators for fuzzy neurons

The equations for fuzzy neural processing written in terms of T and S operators are general, and in order to study in more detail the behavior of such structures, or for application purposes, a selection has to be made for a particular pair of triangular norm/conorm to replace T and S in the formulas. The most used pairs in fuzzy neural modelling are: MIN/MAX (which is treated by the majority of researchers) [Pedrycz and Rocha 1993], [Pedrycz 1994], [Blanco *et al.* 1994], product/probabilistic sum [Pedrycz and Rocha 1993], [Pedrycz 1994] or Hamacher's operators (T^6, S^6 in Table 3.1) [Pedrycz and Rocha 1993]. At present, there is no recommendation specifying which operators one should choose (although it is observed that MIN/MAX needs some modifications to allow learning, e.g. [Pedrycz and Rocha 1993], [Blanco *et al.* 1994]) and it is considered that in general 'one can select any combination of the triangular norms' [Pedrycz 1994]. In order to benefit from the great number of topologies and learning rules developed for classic neurons, fuzzy neurons may need to respect some conditions. Also, the choice of a t-norm should consider aspects of hardware realisation, likely to follow successful simulation studies. The following compares how different t-norms qualify for the implementation of S-T fuzzy neurons.

[1] Several mechanisms for reading the neural weights were proposed. For example, in [Pomerleau 1993] the number of hidden nodes is kept small.

[2] From here onwards the word neuron refers to the type of fuzzy neuron under discussion. Classic (non-fuzzy) neurons are referred as such, and refer to the model described in Appendix G.

3.1.1 Conditions imposed on the synaptic-somatic operators

- Condition 1. The neural model should permit gradient-descent search (on which the majority of neural learning algorithms are based). Functions defined with thresholding by MIN or MAX are poor choices for gradient-descent like methods. Specifically, the derivative is not defined at the thresholding point, as discussed for example in [Pedrycz and Rocha 1993]. Also, having the MAX operator thresholding at 1, as illustrated in Fig. 3.1, results in a null derivative for the ceiling region, where the search can not continue. For these reasons, a condition imposed on the operators involved is that they have nonzero, finite derivatives everywhere.

- Condition 2. The functions should be parametric. Parametrization offers a great modelling flexibility, remarked for example in [Zimmermann 1991], [Di Nola *et al.* 1989], [Pedrycz and Rocha 1993]. In particular Pedrycz and Rocha [Pedrycz and Rocha 1993] suggest that neural learning can involve the parameters of triangular norms. A change of the somatic parameter is equivalent to the modification of the slope of the activation function, which for the case of classic sigmoidal neurons has been shown to affect the learning rate in backpropagation [Thimm *et al.* 1995].

- Condition 3. The selected t-norm (s-norm) should cover the MIN (MAX) case. Parametric families should have these values as their limit for specific values of their parameters. Apart from being the most thorough investigated pair, there are many problems which are best modeled by them.

Selecting a triangular norm. The comparison is made between the 12 operators presented in Table 3.1. These operators are gathered from [Di Nola *et al.* 1989], [Gupta and Qi 1991*b*] and [Butnariu and Klement 1993], to which the reader is referred for more details. In particular Gupta and Qi [Gupta and Qi 1991*b*] enumerate and give properties of all but one of these operators, and Butnariu and Klement [Butnariu and Klement 1993]

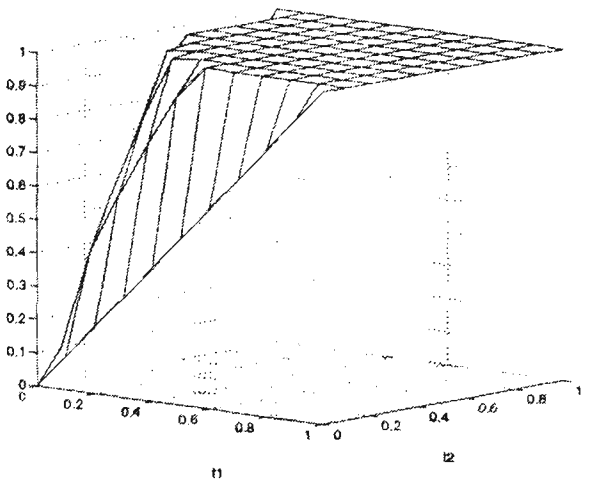


Figure 3.1: Yager's s-norm implementing a neuro-somatic operator

give a clear description of several families of t-norms, including suggestive graphical comparisons. Most applications are based on MIN and product, mainly due to their simplicity and because there is no clear advantage of other t-norms, the best choice being very much problem dependent. For a study that compares the efficiency of using different t-norm pairs in implementing fuzzy logic controllers, the reader is referred to [Gupta and Qi 1991*a*]. According to the three conditions imposed, the selection of a t-norm pair is straightforward (the following refer to T only, but the same applies to S). T^1, T^2, T^3, T^4, T^5 are not parametric and thus do not satisfy Condition 2. In the remaining set only T^6, T^8 and T^{12} are not thresholded by MIN or MAX, and accordingly are the only ones that satisfy Condition 1. One should note now the following inequalities [Gupta and Qi 1991*b*]

$$T^5 \leq T \leq T^1 \tag{3.1}$$

$$T^5 < T^3 < T^2 < T^4 < T^1 \tag{3.2}$$

and the following limits [Gupta and Qi 1991*b*] [Butnariu and Klement 1993]:

$$\lambda \to 0, \;\; T^6 \to T^5. \;\; \text{Also } \lambda \to \infty, \;\; T^6 \to T^4. \tag{3.3}$$

$$\lambda \to 0, \;\; T^8 \to T^5. \;\; \text{Also } \lambda \to \infty, \;\; T^8 \to T^1. \tag{3.4}$$

Table 3.1: Triangular norms and co-norms

T-norms	S-norms
$T^1(x,y) = MIN(x,y)$	$S^1(x,y) = MAX(x,y)$
$T^2(x,y) = x \cdot y$	$S^2(x,y) = x + y - xy$
$T^3(x,y) = MAX(x+y-1,0)$	$S^3(x,y) = MIN(x+y,1)$
$T^4(x,y) = \frac{xy}{x+y-xy}$	$S^4(x,y) = \frac{x+y-2xy}{1-xy}$
$T^5(x,y) = \begin{cases} x & \text{if y = 1} \\ y & \text{if x = 1} \\ 0 & \text{otherwise} \end{cases}$	$S^5(x,y) = \begin{cases} x & \text{if y = 0} \\ y & \text{if x = 0} \\ 1 & \text{otherwise} \end{cases}$
$T^6(x,y) = \frac{\lambda xy}{1-(1-\lambda)(x+y-xy)}$	$S^6(x,y) = \frac{\lambda(x+y)+xy(1-2\lambda)}{\lambda+xy(1-\lambda)}$
$T^7(x,y) = MAX(1-((1-x)^p+(1-y)^p)^{1/p},0)$	$S^7(x,y) = MIN((x^p+y^p)^{1/p},1)$
$T^8(x,y) = \frac{1}{1+((\frac{1}{x}-1)^\lambda+(\frac{1}{y}-1)^\lambda)^{1/\lambda}}$	$S^8(x,y) = \frac{1}{1+((\frac{1}{x}-1)^{-\lambda}+(\frac{1}{y}-1)^{-\lambda})^{-1/\lambda}}$
$T^9(x,y) = \frac{xy}{MAX(x,y,\lambda)}$	$S^9(x,y) = 1 - \frac{(1-x)(1-y)}{MAX(1-x,1-y,\lambda)}$
$T^{10}(x,y) = MAX(\frac{x+y-1+\lambda xy}{1+\lambda},0)$	$S^{10}(x,y) = MIN(x+y+\lambda xy,1)$
$T^{11}(x,y) = MAX((1+\lambda)(x+y-1)-\lambda xy,0)$	$S^{11}(x,y) = MIN(x+y+\lambda xy,1)$
$T^{12}(x,y) = \log_s[1+\frac{(s^x-1)(s^y-1)}{s-1}]$	$S^{12}(x,y) = 1 - \log_s[1+\frac{(s^{1-x}-1)(s^{1-y}-1)}{s-1}]$

$$s \to \infty,\ T^{12} \to T^3.\ \text{Also } s \to 0,\ T^{12} \to T^1. \tag{3.5}$$

According to these observations and Condition 3 one can eliminate T^6, which has T^4 as its upper limit and therefore can not reach T^1 (MIN) as required.

T^8 and T^{12} are the only ones that satisfy the imposed requirements. At this point one should remark that it is possible to create new t-norms, for example as shown in [Kaufmann and Gupta 1988], some of which may satisfy as well all the conditions imposed here. The purpose of this comparison was to select from the set of currently defined operators the one that is best suited for neural modelling according to the selected criteria. In order to continue the investigation of S-T fuzzy neural networks (S-T FNN), one must chose either T^8 or T^{12}, and thus specify the neuron by functions which allow numerical calculations.

T^{12} has the advantage of covering the operators used in probabilistic reasoning, product/probabilistic sum (T^2/S^2), which is a limit case for (T_s^{12}/S_s^{12}) when $s \to 1$. The disadvantage is that in order to cover the case $s = 1$ (in which point T^{12} is defined equal to T^2, see (3.6)) one must switch between functions T^{12} and T^2. If a hardware implementation is envisaged, then a solution must be found to this switching. T^8 has the advantage of covering the full range of possible t-norms [T^5, T^1] (however, this does *not* cover T^2, i.e. the two are not identical for any value of λ). Its disadvantage is that is not defined for null arguments, and for values of x (or y) close to 0 the term 1/x (or 1/y) becomes very large, which may also cause hardware problems.

It is considered here that T^{12} is a preferable choice, and it is the one selected for further study.

3.2 Fundamental fuzzy neurons

T^{12} can be associated with other t-norms, to form the family of t-norms defined by

$$T_s(x,y) = \begin{cases} MIN(x,y) & \text{if } s = 0, \\ x \cdot y & \text{if } s = 1, \\ \log_s[1 + \frac{(s^x-1)(s^y-1)}{s-1}] & \text{if } 0 < s < \infty, s \neq 1, \\ MAX(0, x+y-1) & \text{if } s = \infty. \end{cases} \tag{3.6}$$

$$S_s(x,y) = \begin{cases} MAX(x,y) & \text{if } s = 0, \\ x + y - x \cdot y & \text{if } s = 1, \\ 1 - \log_s[1 + \frac{(s^{1-x}-1)(s^{1-y}-1)}{s-1}] & \text{if } 0 < s < \infty, s \neq 1, \\ MIN(1, x+y) & \text{if } s = \infty. \end{cases} \tag{3.7}$$

This family of triangular norms, which was initially investigated by Frank [Frank 1979], received special attention in [Butnariu and Klement 1993] where they were referred to as *fundamental* t-norms. This term is adopted here to specify a class of fuzzy neurons. Due to Condition 1, the class of fuzzy neurons is restricted to $0 < s < \infty$, for which T_s is also continuous, as it was shown in [Butnariu and Klement 1993] that

$$\lim_{s \to s_0} T_s = T_{s_0}, \qquad \forall s_0 \in [0, \infty) \tag{3.8}$$

(and the same property is valid for S).

Definition. A *fundamental fuzzy neuron* (FFN) is an S-T fuzzy neuron for which the S and T operators are the fundamental t-norms given by Eqs. (3.6), (3.7), for $0 < s < \infty$. A network of FFN forms a *fundamental fuzzy neural network* (FFNN).

The characteristic function of a FFN with m inputs is expressed by

$$y = 1 - \log_{s_S}[1 + \frac{(s_S^{1-t_1} - 1)(s_S^{1-t_2} - 1)\ldots(s_S^{1-t_m} - 1)}{(s_S - 1)^{(m-1)}}], \tag{3.9}$$

where each t_i represents the synaptic contribution of input x_i modulated by the weight w_i

$$t_i = \log_{s_T}[1 + \frac{(s_T^{x_i} - 1)(s_T^{w_i} - 1)}{s_T - 1}]. \tag{3.10}$$

The parameter of the s-norm, s_S, is referred to as the somatic parameter, and the parameter of the t-norm, s_T, is referred to as the synaptic parameter.

The characteristic function for a S-T neuron with two inputs is presented in Fig. 3.2. This illustrates the function y = $S_{100}(t_1, t_2)$, where t_1 and t_2 are the inputs after the synaptic composition.

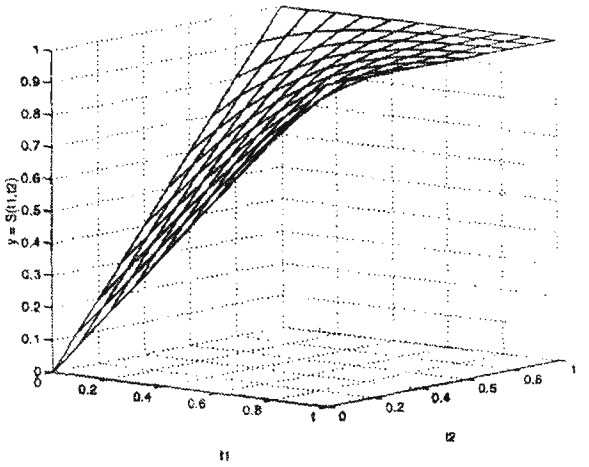


Figure 3.2: Somatic characteristic of the fundamental fuzzy neuron

3.3 Learning in fundamental fuzzy neural networks

For a system whose behavior is known in terms of input-output pairs, one can identify a model to fit the available data. If one chooses to do the modelling using a neural network structure, the identification consists in finding the appropriate neural parameters which ensure the mapping. In particular, if the neurons are fuzzy then a relational interpretation supports the model. In this case, finding the appropriate neural weights is equivalent to finding the elements of a fuzzy relation, which is a solution to the FRE characterising the system (see Eqs. (2.38), (2.39)). The extension of learning to include modifications of the synaptic and somatic parameters is equivalent to the search for a suitable composition for the FRE, or, in other terms, for a suitable fuzzy logic for reasoning in such a system. This section derives the equations that allow learning by FFNN.

3.3.1 Gradient-descent learning in fundamental fuzzy neural networks

Most NN learning techniques are gradient descent (GD) based (the best known is back-propagation [Rumelhart *et al.* 1986]). In a fuzzy relational context, learning by gradient descent was first applied by Pedrycz [Pedrycz 1991*b*]. The equations for learning depend on the t-norms that define the chosen composition. In the following, the gradient-descent equations for FFNN are derived. Consider the case of an input matrix X(p,m), a weight (relation) matrix W(m,n), an output matrix Y(p,n) calculated by S-T composition, and a target matrix B(p,n) representing the desired outputs. This is a case of k input-output pairs, with m inputs for each of the n output nodes. The equation to solve is Equ. (2.39) detailed in (2.38). For simplicity, the equations are written in a single t-norm parameter (s), i.e. considering that the t-norm and the s-norm have the same parameter $s_S = s_T$, in which case they comply with De Morgan's laws and are called *corresponding t-norms*. The results can be easily extended to non-corresponding t-norms by replacing s with s_S

and s_T as will be indicated at the end of the calculations. The weights are determined in successive approximations using a GD algorithm. The algorithm is based on the modifications of weights as to minimize a distance between the outputs produced by information processing through the network with current weights and the target output values. The function to be minimized is the sum of squared errors (SSE) taken over all examples (k = 1,...,p) and all output neurons (j = 1,...,n),

$$SSE(\tau) = \sum_{k=1}^{p}\sum_{j=1}^{n}(y_{kj}(\tau) - b_{kj}(\tau))^2, \tag{3.11}$$

where τ is the iteration index number, y_{kj} the j-th output in the k-th example, b_{kj} the target element kj.

In the search for the a minimum of SSE, w_{ij} moves along the gradient of SSE. Its value at instant $(\tau + 1)$ is obtained from its predecessor at the moment τ, using the updating formula

$$w_{ij}(\tau + 1) = w_{ij}(\tau) - \rho\frac{\partial SSE(\tau)}{\partial w_{ij}}, \tag{3.12}$$

where ρ is the learning rate (increment). To simplify the presentation, τ is omitted from equations. The partial derivative of SSE with respect to w_{ij} is given by:

$$\frac{\partial SSE}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}(\sum_{k=1}^{p}\sum_{j'=1}^{n}(y_{kj'} - b_{kj'})^2 = \sum_{k=1}^{p}\sum_{j'=1}^{n}\frac{\partial}{\partial w_{ij}}(y_{kj'} - b_{kj'})^2 = \sum_{k=1}^{p}2(y_{kj} - b_{kj})\frac{\partial y_{kj}}{\partial w_{ij}} \tag{3.13}$$

(as $j' \neq j$ does not depend on the weights received by node j (w_{ij})).

The output of the neuron j presented with the k-th input example is

$$y_{kj} = 1 - \log_s[1 + \frac{(s^{1-t^k_{1j}} - 1)(s^{1-t^k_{2j}} - 1)\ldots(s^{1-t^k_{mj}} - 1)}{(s-1)^{(m-1)}}], \tag{3.14}$$

or in the form

$$y_{kj} = 1 - \log_s[1 + \frac{\prod_{i=1}^{m}(s^{1-t_{ij}^k} - 1)}{(s-1)^{m-1}}]. \tag{3.15}$$

The synaptic contribution of input i to neuron j in example k, is

$$t_{ij}^k = \log_s[1 + \frac{(s^{x_{ki}} - 1)(s^{w_{ij}} - 1)}{s-1}]. \tag{3.16}$$

The change that the weight w_{ij} produces on the output of neuron j in example k is

$$\frac{\partial y_{kj}}{\partial w_{ij}} = \frac{\partial y_{kj}}{\partial t_{ij}^k}\frac{\partial t_{ij}^k}{\partial w_{ij}}. \tag{3.17}$$

The change of output as affected by synaptic input t_i is

$$\frac{\partial y_{kj}}{\partial t_{ij}^k} = \frac{-(1 + \frac{(s^{1-t_{1j}^k}-1)\ldots(s^{1-t_{ij}^k}-1)\ldots(s^{1-t_{mj}^k}-1)}{(s-1)^{m-1}})'}{ln(s)(1 + \frac{(s^{1-t_{1j}^k}-1)\ldots(s^{1-t_{ij}^k}-1)\ldots(s^{1-t_{mj}^k}-1)}{(s-1)^{m-1}})}, \tag{3.18}$$

which leads after derivation and simplifications to

$$\frac{\partial y_{kj}}{\partial t_{ij}^k} = \frac{s^{1-t_{ij}^k}\prod_{g=1,g\neq i}^{m}(s^{1-t_{gj}^k} - 1)}{(s-1)^{m-1} + \prod_{g=1}^{m}(s^{1-t_{gj}^k} - 1)}. \tag{3.19}$$

The synaptic inputs depend on the weight, (omit for the moment indices i and j)

$$\frac{\partial t}{\partial w} = \frac{(1+\frac{(s^x-1)(s^w-1)}{s-1})'}{ln(s)(1+\frac{(s^x-1)(s^w-1)}{s-1})} = \frac{s^w(s^x-1)}{(s-1)+(s^x-1)(s^w-1)} \tag{3.20}$$

i.e.

$$\frac{\partial t_{ij}^k}{\partial w_{ij}} = \frac{s^{w_{ij}}(s^{x_{kj}}-1)}{(s-1)+(s^{x_{kj}}-1)(s^{w_{ij}}-1)}. \tag{3.21}$$

Replacing (3.19) and (3.21) in (3.17) we obtain

$$\frac{\partial y_{kj}}{\partial w_{ij}} = \frac{\partial y_{kj}}{\partial t_{ij}^k}\frac{\partial t_{ij}^k}{\partial w_{ij}} = \frac{s^{1-t_{ij}^k}\prod_{g=1,g\neq i}^{m}(s^{1-t_{gj}^k}-1)}{(s-1)^{m-1}+\prod_{g=1}^{m}(s^{1-t_{gj}^k}-1)}\frac{s^{w_{ij}}(s^{x_{kj}}-1)}{(s-1)+(s^{x_{kj}}-1)(s^{w_{ij}}-1)} \tag{3.22}$$

which (with t_{ij}^k given by (3.16)) is introduced in (3.13) allowing the weight update given by (3.12).

The equations for non-corresponding t-norms can be obtained directly from the previous equations by replacing s with s_S in (3.14), (3.15), (3.18), (3.19), and replacing s with s_T in (3.19), (3.20), (3.21), and also replacing s_S in the first term of (3.22), and s_T in the second term of (3.22).

The computations need the specification of the synaptic/somatic parameters for t-norms and of the learning rate (or increment *inc*). A method which automatically detects optimal t-norm parameters is presented in the next section. As for the learning rate, it is

known from the literature on GD searches how the size of the step affects the outcome: if it is too small the convergence is slow, while if it is too big the search may fail to converge. This is illustrated in Fig. 3.3 which plots the SSE versus the number of steps.

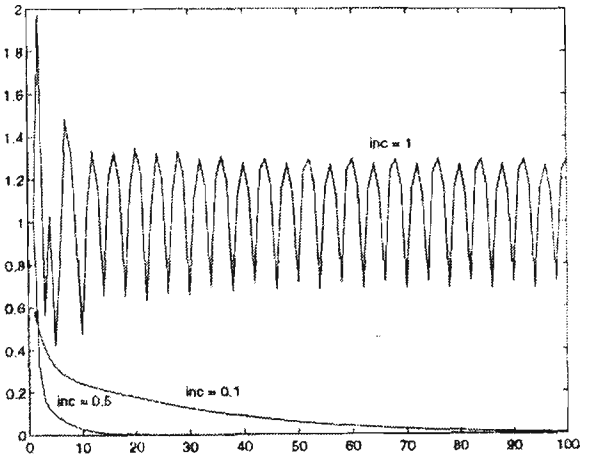


Figure 3.3: The learning rate affects the convergence

3.3.2 The effects of synaptic and somatic adaptation in neural processing

The equations of information processing by FFNN, and also the learning equations derived in the preceding section, indicate that parameters s_S and s_T have an influence on the results. This section shows more precisely how these parameters affect network behavior. Modifications of the logic parameter of the neurons have an equivalent in the classic NN literature. A direct analogy can be drawn between the somatic parameter s_S and the parameter which controls the slope of the activation function of a classic neuron. In [Thimm *et al.* 1995] it is shown that changing the gain (steepness of the slope) is equivalent to changing the learning rate and the weights. The sigmoidal function to which Thimm refers to is [3]

$$y(a) = \frac{\gamma}{1 + e^{-\beta a}} \tag{3.23}$$

[3]The results are valid for other activation functions that depend on the product βa.

where β is the gain, $\gamma\beta$ is the steepness (slope) of activation function and a is given by the weighted sum of the inputs

$$a = \sum_i w_i x_i. \tag{3.24}$$

Thimm shows that if two networks are topologically identical, and one has the gain β times smaller, the learning rate β^2 times higher, and the weights β times higher than the other, then the networks are equivalent under on-line backpropagation. Large gains produce results similar to large learning rates. For FNN, a larger value of s_S determines a steeper activation function. As a particular case, different s values can be used by different neurons, in the same way as pointed out by Rumelhart [Rumelhart *et al.* 1986] for the classic NN.

Apart from increasing the learning speed, a more interesting modification would be to adapt the neural parameters in order to achieve a better mapping. In [Hirota and Pedrycz 1994] a nonlinear sigmoidal element (of a similar form to the one used by Thimm) is placed in series with an OR neuron, and the nonlinearity parameter of its activation function is modified during learning. The adaptation leads to improved results. It is interesting to observe that not the parametric nonlinearity *per se* was the cause of a better results as suggested in the paper ([Hirota and Pedrycz 1994]). For the same nonlinearity, as follows from Thimm's result, a modification of its parameter β would result in a scaled solution which would produce the same mapping. What contributed in the case of [Hirota and Pedrycz 1994] is the fact that the input to the activation function was a non-linear combination of inputs and weights, as opposed to the linear case of (3.24).

The following interesting observation can be made here. For classic neural networks, with neurons as described by equations similar to (3.23), (3.24), modifications of the somatic parameter can not offer better mappings [4]. This follows directly from Thimm's result. However, for fundamental fuzzy neurons, or for parametric S-T neurons in

[4]This is valid for the common case, when all neurons have the same gain for the activation function.

general, better mappings are possible, and adaptation mechanisms for somatic/synaptic parameters can lead to improved modelling capabilities. Equations (3.6) (3.7) show that the relationship between weights and the gain parameter is not linear (for other parametric t-norms, see Table 3.1). It follows that some synaptic/somatic combinations offer better modelling solutions, and this is illustrated in an example in the next section. An algorithm which permits finding such optimal parameters is also described in the next section.

Processing in FFNN with different synaptic/somatic parameters can be studied in relation to the ordering of parametric S-T compositions. In order to derive such an ordering, it is useful to note some important properties of the operators on which the S-T composition is based. These properties are proved in [Butnariu and Klement 1993].

1. The fundamental t-norms form a *decreasing family* :

$$T_\infty \leq T \leq T_0, \tag{3.25}$$

and also for all a, b $\in$ $[0, \infty]$ with a < b

$$T_b \leq T_a. \tag{3.26}$$

By duality, fundamental s-norms form an increasing family

$$S_0 \leq S \leq S_\infty, \tag{3.27}$$

and also for all a, b $\in$ $[0, \infty]$ with a < b

$$S_a \leq S_b. \tag{3.28}$$

2. Fundamental t-norms are *strict* in the sense that they are continuous and satisfy the property

$$T(x, y) < T(x, z), \text{ whenever } x > 0 \text{ and } y < z. \tag{3.29}$$

Fundamental s-norms are strict, being continuous and having the property that

$$S(x, y) < S(x, z), \text{ whenever } x < 1 \text{ and } y < z. \tag{3.30}$$

Consider the simplified notation $ST(s,t)$ standing for $S_s - T_t$, the S-T composition having s as the parameter for the s-norm and t the parameter for the t-norm. Consider P, Q, R fuzzy relations, and for each (x_i, z_k) the S-T composition expressed as a function of (s,t) (the argument (x_i, z_k) being omitted for simplicity) $ST(s,t) = \overset{n}{\underset{j=1}{S_s}}[R(x_i,y_j)T_tQ(y_j,z_k)]$.

Proposition. Let a and b be two positive real numbers satisfying the relation a < b. The following relations exists:

$$ST(a,b) \leq ST(a,a) \leq ST(b,a), \tag{3.31}$$

$$ST(a,b) \leq ST(b,b) \leq ST(b,a). \tag{3.32}$$

Proof:

1. The arguments of the s-norm in both ST(a,a) and ST(b,a) are the same (i.e. $[R(x_i,y_j)T_aQ(y_j,z_k)]$). As shown in Equ. (3.28), S is an increasing function in its logic parameter. With a < b that means $ST(a,a) \leq ST(b,a)$.

2. $ST(a,b) \leq ST(b,b)$. The same reasoning as before, but for a different argument of the S operator (now obtained using T_b).

3. The argument of $ST(a,a)$ (which is $[R(x_i,y_j)T_aQ(y_j,z_k)]$) is greater (or equal) than the argument of ST(a,b), (which is $[R(x_i,y_j)T_bQ(y_j,z_k)]$) according to (3.26). From (3.30) it follows that $ST(a,b) \leq ST(a,a)$.

4. The same argumentation as before, this time for $ST(b,b) \leq ST(b,a)$.

The relation between ST(b,b) and ST(a,a) depends on the values of the relational matrices involved in the composition, as one function (T) is decreasing and the other (S) is increasing.

Knowledge of this ordering allows neurons to guide the modification of neural parameters in response to the input data available at inputs. For example if the outputs

are too high and saturate[5], then a decrease of the somatic parameter (which can be combined with an increase in the synaptic parameter) will scale back the output into the active range. For example for neurons with 192 weighted inputs used in the visuo-motor coordination described in Chapter 6, all the S-T compositions that have $s_S = s_T$ saturate the output. The choice of $s_S = 0.1$ or smaller, and $s_T = 10$ or larger, is more appropriate.

3.4 Applications to resolution of fuzzy relational equations and fuzzy system identification

The learning mechanisms presented in the previous section have direct applications to the resolution of fuzzy relational equations, which, as explained in Chapter 2, determine the identification in systems modeled by fuzzy relations. Neural learning provides a powerful numerical technique for FRE resolution. This is particularly useful for systems of S-T FRE, for which analytical resolution methods are not available. It is also a way of finding approximate solutions for any type of FRE.

3.4.1 Application to solving FRE of a given composition

The equations derived in the previous section allow the numerical resolution of FRE based on S-T composition or its particular cases, such as MAX-T, MAX-MIN, or even S-MIN, which can be employed to extend MAX-MIN by incorporating a global effect. For S-T FRE the gradient descent search is made directly by replacing the particular values of parameters s_S and s_T in equations. For FRE that involve MAX and MIN the continuity of parametric t-norms (3.8) is used. For example MAX-MIN is the instantiation of S-T

[5]Here saturation is considered when neural outputs take values in a small vicinity of 1, generally due to a large number of inputs.

composition for s = 0, $(S\text{-}T)_{s\to 0}$ = MAX-MIN. Using the GD equations, solutions are searched for S-T FRE with $s \to 0$ [6].

The following discuss the results of applying GD learning in FFNN, for the purpose of finding solutions for MAX-MIN FRE. Details of the examples are given in Appendix F.

Example 1: An exact solution (see also Example 1 in Appendix F).
In a first case, which is an example taken from [Blanco *et al.* 1994], an exact solution exists, and after a single step the solution was identified with an accuracy better than 0.0001 per element. The search was performed for $s = 10^{-9}$ and learning rate $\rho = 1$. The result is qualitatively similar to the one obtained in [Blanco *et al.* 1994] using a smooth derivative for a MAX-MIN NN. One should remark here that the solution was also obtainable by analytic methods. In this case the maximal solution given by (2.10) and the minimal one given by (2.13) coincide, thus the solution being unique.

Example 2: An approximate solution (see also Example 2 in Appendix F).
In a second case, which is an example from [Pedrycz 1990*a*] (also used as a test case in [Negoita *et al.* 1994]), the system does not admit exact solutions and an approximate solution is sought. The search was performed for $s = 10^{-6}$ and the search was stopped after 200 steps. The resulting solution is qualitatively equivalent to the solution obtained in [Negoita *et al.* 1994] using genetic algorithms, which is a better solution than the one obtained in [Pedrycz 1990*a*]. Four indices were used for comparison: (1) maximum error per input-output pair, (2) maximum error in all input-output pairs, (3) sum of absolute

[6]Gradient learning in MAX-MIN FNN was proposed for solving MAX-MIN FRE [Pedrycz 1991*b*]. However, for MAX-MIN FNN the gradient search can not be employed directly, as derivatives have a binary character. The solutions proposed are based on an approximation of MAX and MIN with differentiable functions or 'smoothing' the derivative [Pedrycz and Rocha 1993], [Blanco *et al.* 1994]. The possibility of approximation of MAX-MIN with any parametric family of triangular norms that tends to maximum/minimum for some values of the parameter was also suggested in [Pedrycz and Rocha 1993], however I am not aware of any implementation of this proposal.

errors (SAE), (4) sum of squared errors (SSE). Except for SAE, the result was better for FFNN trained by GD. Note however that the results are not directly comparable, because the optimization index in [Negoita *et al.* 1994] was SAE, and in GD methods the optimization is done for SSE (and other indices may also be small as a consequence of using SSE). The number of necessary steps for obtaining the results was smaller for GD than for the genetic algorithm: 200 and ~ 10^3 respectively. However, different learning rates may give different numbers of steps for the same accuracy [7].

The two examples show that FFNN can be successfully applied for solving MAX-MIN FRE. The advantage of using a FFNN (as opposed to a genetic algorithm) is that a unique structure is used for representing the relation and for implementing the learning mechanism. A FFNN has also greater flexibility than a MAX-MIN NN as used in [Blanco *et al.* 1994], as it can change the composition to obtain a model which better 'fitts' the data. This is illustrated in the next section.

3.4.2 Resolution of FRE with adaptive composition: fuzzy relation - fuzzy composition optimality

The classic approach to fuzzy modelling is to choose a composition and to identify a fuzzy relation. There is little indication of which composition to choose, and quite often MAX-MIN is chosen for its simplicity, while not necessarily being the best choice. Instead of limiting the identification to finding a fuzzy relation, one can search also for an optimal composition. In the context of FFNN that means extending the search for a weight matrix to include finding optimal synaptic and somatic parameters for neurons. As indicated in a previous section, this has the potential of offering a better mapping. Using the t-norm parameter as the only variable in an evidence aggregation

[7] Pedrycz has reported a case in which a combined genetic - neural search gives better results than a simple neural search [Pedrycz 1994].

network based on Yager's operators, Keller and Krishnapuram [Keller and Krishnapuram 1992] have shown that training can lead to the value of the parameter that optimises a performance index. Their search however was performed assuming *fixed* weights. Even earlier in [Pedrycz 1983*a*], Pedrycz expressed the idea of best 'fitting' of the model by involving parameter search on the composition. However, he restricted the form of the acceptable fuzzy relation, by calculating it with a formula which depends on the parameter of the composition, and the search was performed to identify the optimal value of this parameter. Thus, the solution obtained is optimal only for the given form of fuzzy relation. In here I propose to use an unconstrained search for the fuzzy relation and the parameters of the t-norms which define the composition. The parameters of the t-norms can be used in the same way as the other variables (elements of the fuzzy relation) in an iterative search, to find an optimal value for the approximation.

Example 3: A better approximate solution for adaptive composition (see also Example 3 in Appendix F).
Using the same data as in Example 1, a combined search for W and s gives a weight matrix and a logic giving a better approximation ($SSE = 0.93$) than the one obtained for MAX-MIN ($SSE = 1.04$).

Example 4: An exact solution in which both fuzzy relation and fuzzy logic are found (see also Example 4 in Appendix F).
The focus now is on a system for which an exact solution exists, as the output Y is determined by the S-T composition (with $s = s_S = s_T = 10$) of input X with W, X and W randomly generated. Searching for a fuzzy relation only in the context of a fixed given composition gives models with various power of approximation, as illustrated in Fig. 3.4. A combined search identified the fuzzy relation and the best composition. In this example, each step of the GD procedure was intercalated with a selection of the s parameter (of same, higher, or lower value than the current one) that gave the lower SSE. Fig. 3.5 shows the convergence of SSE while s was converging from smaller values of

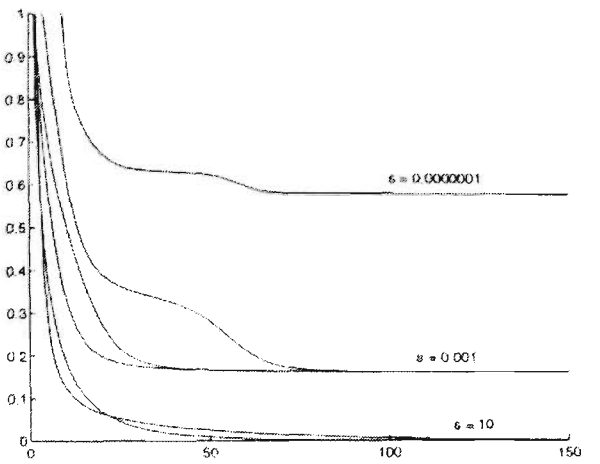


Figure 3.4: Accurate modelling for the composition with s = 10, and errors in modelling with others (sum of squared errors versus number of iterations)

s to s = 10 as shown in Fig. 3.6. The same convergence towards the optimal logic (s = 10) is shown when the search starts from larger values of s.

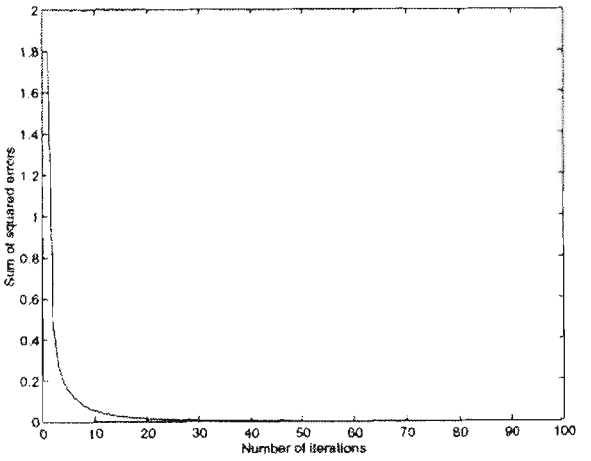


Figure 3.5: SSE convergence in the combined search for fuzzy relation and fuzzy logic

3.5 The 'rules in weights' representation

A weakness of the artificial neural networks approach to system modelling is the difficuly of interpreting their internal representations. A classic neural network appears as a 'black box', which performs well in the cases for which it was trained, but whose behavior

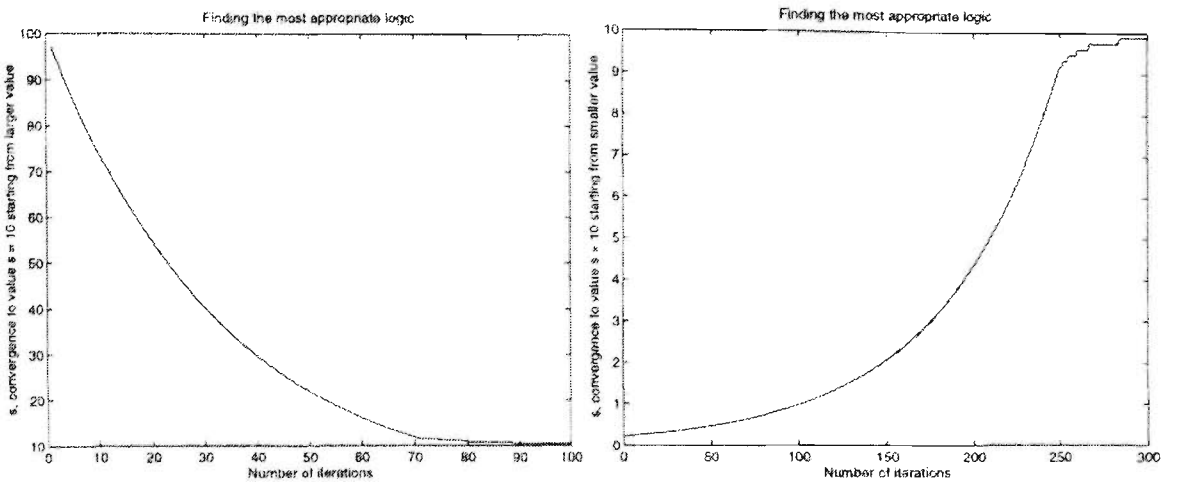


Figure 3.6: Convergence towards the optimal composition while also finding the fuzzy relation

in new situations is hard to predict, and which offers no logical explanation on how it has reached a certain decision. For this reason they are not selected for applications where safety is critical, such as air traffic control, power plant operation, etc. Currently there are several architectures which offer rule extraction from artificial neural networks, e.g. KBANN [Towell and Shavlik 1993], or BRAINNE [Sestito and Dillon 1994] (see [Andrews *et al.* 1995] for a recent survey of extraction techniques). In neural implemented fuzzy rule based systems knowledge extraction consists in identifying the rules and membership functions that fit the training data. Implementations with classic neurons are reported in [Horikawa *et al.* 1992], [Takagi *et al.* 1992], [Jang 1995] for FRC2. Pedrycz and Rocha [Pedrycz and Rocha 1993] advise the use of fuzzy neurons in knowledge based-networks, which are networks that map concepts and implement FRC3. The input interface is ensured by *matching* neurons. Compact methods (FRC2, FRC3) lead to the 'rules in neurons' form of representation. The representation observed here in connection to FRC4 is a 'rules in weights' representation and offers all the advantages of true distributed representations.

When presented with examples in the form of input-output pairs, the FNN learns the mapping in terms of corresponding fuzzy relation (weights) and S-T composition (synaptic/somatic parameters). It is interesting to observe that clusters in the weight

space take the shape of a distributed rule table-like, which is an image of a macro-level rule table modulated by membership functions. This is in fact a rule extraction procedure[8].

In the example which follows, to facilitate the analysis, the fuzzy system which generated the input-output data are shown at the start. The problem can be seen as a transformation of representation, or how a neural system can learn a model from a linguistic description. In practice, the focus is generally on learning from data, the only information available being examples of I-O pairs.

The system for which it is proposed to identify the fuzzy relational model has one input and one output. The fuzzy sets and the way they map are represented in Fig. 3.7 (each input maps to the output on its right) and in Table 3.2 [9]. The input and output domains are mapped onto the [0,1] interval, and are sampled in 11 points (the same resolution was also found suitable in [Keller and Krishnapuram 1992]). The training pairs are formed by the sampled representations of mapping sets (the data set is shown in Appendix E). The training was done by GD. As the input and output sets were each sampled in 11 points, the resulting fuzzy relation is of size (11,11) and the associated network has 11 nodes, each connected to the 11 inputs. At the end of learning, the weights form clusters which constitute a distributed rule-base (Fig. 3.8) reflecting directly the rules in Table 3.2.

The knowledge shaped in the weight space is the combined knowledge from the rule table and the membership functions. Adaptation is simplified, as tuning of rules and membership functions does not need to be made separately. A change of rule at macro-level (Table 3.2) affects many points, and it is equivalent to modifying an entire cluster

[8]If the weights outside the cluster contribute this can be seen as 'exceptions to rules', small deviations which are recorded in this way (the concept of rules is our construction and may not capture a situation perfectly , not even if rules are fuzzy).

[9]The abbreviations of fuzzy set labels stand for the following: VS - Very Small, S - Small, M - Medium, L - Large, VL - Very Large.

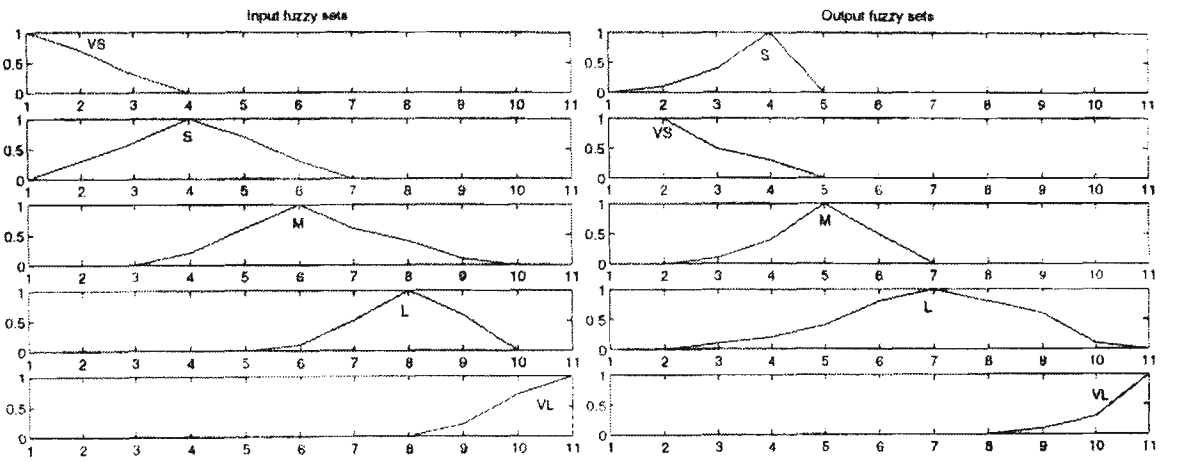


Figure 3.7: Fuzzy sets and their mapping: inputs and corresponding outputs

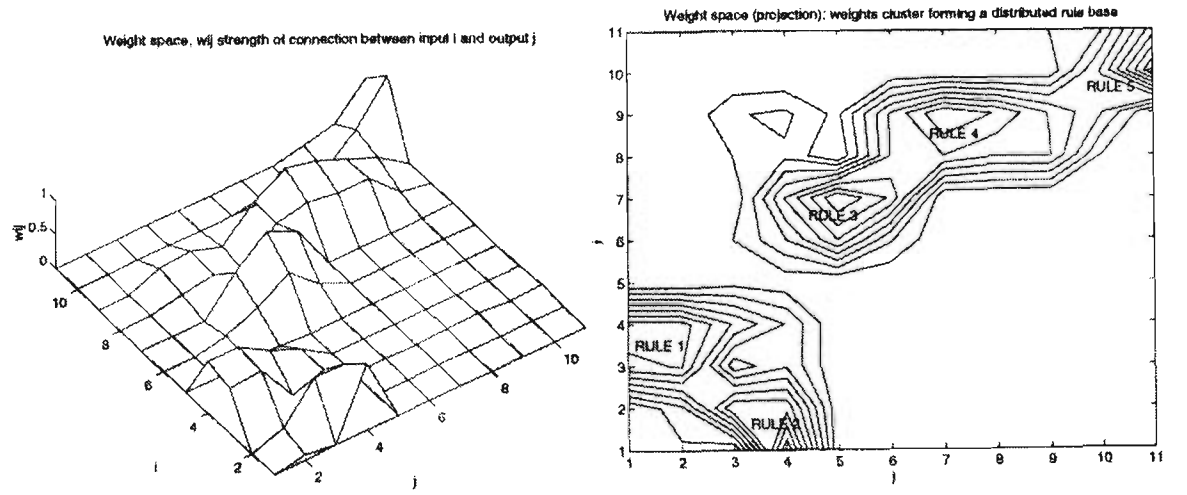


Figure 3.8: The weights shape a distributed, microlevel rule table

Table 3.2: Rule table describing the set mapping

	Output				
Input	VS	S	M	L	VL
VL					Rule 5
L				Rule 4	
M			Rule 3		
S	Rule 1				
VS		Rule 2			

in Fig. 3.8. On the contrary, the change of one weight (a mini-rule) in the distributed rule base of Fig. 3.8 has a local effect, allowing for a finer control in shaping the system.

The network encodes the knowledge of mapping in neural weights, which reflect the fuzzy relation between inputs and outputs. The neurons act as distributed elementary fuzzy processors performing the S-T composition of inputs with the fuzzy relation. The network is thus a *distributed fuzzy system*, and the weights have the meaning of a point to point relationship between samples in the input and samples in the output space. It may happen in fuzzy modelling that although the membership functions and rules are in the way the experts have expressed, the result of processing differs from experts' predicted result. One thing that experts can not express is the logic of their reasoning (i.e. the composition used, which in not necessary MAX-MIN), to which a good approximation can be identified from examples, as shown in a previous section.

3.6 Summary

A new type of fuzzy neuron operating on triangular norms was defined and investigated. The fundamental fuzzy neuron was defined based on the fundamental t-norms which were found to be the optimal t-norms (from a learning point of view) for implementing synaptic and somatic operators in fuzzy neurons. It was shown that synaptic modifications can

not improve models of classic neurons, but they affect the quality of the solution using fuzzy neurons, and optimal values may be found. GD equations for learning by FFNN were derived. It was shown how learning can address the simultaneous identification of optimal weights and synaptic/somatic parameters. In terms of FRE resolution this means a search for both fuzzy relation and fuzzy composition, which leads to better modelling than the classic approach, which look for solutions for FRE under the assumption of a prespecified composition. Finally, it is shown that during learning the weights cluster and form a distributed rule base in the weight space. The 'rules in weights' representation allows S-T FNN to be seen as 'transparent boxes' rather than 'black boxes'.

Chapter 4

Fuzzy neurons with shared weights

Fuzzy neurons discussed so far allow the implementation of distributed reasoning for single-input systems. In this chapter, a general model of fuzzy neuron is proposed, which supports the implementation of multi-input systems. This neuron has properties of a general purpose computational element.

4.1 Implementation of multi-dimensional fuzzy systems

The fuzzy relational equation (2.39) formalises information processing in a relational structure modelling linguistic statements of the form *Input X maps to output Y*. The case when X stands for a conjunction of inputs ($X = X_1$ AND X_2 AND ...AND X_m) can be described by the FRE of complex structure (see also [Di Nola *et al.* 1989])

$$(X_1 \; T \; X_2 \; T \; \ldots \; X_n) \circ W = Y \tag{4.1}$$

with the fuzzy relation W defined in the Cartesian product [1] $W : \overset{m}{\underset{i=1}{\times}} X_i \times Y$. One should detail here a distinction between the compact and distributed classes of fuzzy reasoning. In compact reasoning (FRC2, FRC3) the conjunction of terms in antecedent is interpreted as a scalar operator, and each input x_i is matched against the classes defined on that variable, returning a degree of membership, $\mu_{X_i}(x_i)$. The degree of matching the antecedent is then calculated as $\mu_X(x_1, \ldots, x_m) = \overset{m}{\underset{i=1}{T}} \mu_{X_i}(x_i)$ the result being uni-dimensional (1D). This degree of membership then modulates the output set selected by the rule. The contribution of all rules gives a fuzzy set, which is the final result, unless a crisp value is required and in this case a defuzzification procedure must be applied.

In distributed reasoning (FRC1, FRC4, see also [Takagi *et al.* 1992] and [Foulloy *et al.* 1994]) the mapping is from the m-dimensional space $\overset{m}{\underset{i=1}{\times}} X_i$ to the one dimensional space Y. In the perspective of FRE this is interpretable by considering T^* a pointwise T between all combinations of elements of X_i. Thus, a m-dimensional (mD) input is matched against a mD fuzzy set $\overset{m}{\underset{i=1}{T^*}} X_i$, giving a mD result, which is further composed with the fuzzy relation W. Equ. (4.1) can be rewritten as

$$Y = \overset{m}{\underset{i=1}{T^*}} X_i \circ W = S(T(\overset{m}{\underset{i=1}{T^*}} X_i, W)). \tag{4.2}$$

A neural implementation of this form of reasoning is suggested by an analogy with the classic Sigma-Pi model [Williams 1986]. The *Sigma-Pi activation function* is defined as $y = f * g$, with

$$g(x_1, \ldots, x_m) = \sum_{E_j \in P} w_j \cdot \prod_{i \in E_j} x_i$$

where f is nondecreasing function into [0,1] (usually by a sigmoid), $*$ is the composition of the two functions in a right-left manner, $\sum$ is the simple sum, Π is the product, and P is the power set of [1,...,m]. This equation can be rearranged as

$$g(x_1, \ldots, x_m) = \sum_{E_j \in P} (\Pi(\prod_{i \in E_j} (x_i), w_j). \tag{4.3}$$

[1] The Cartesian product of two fuzzy sets is defined as a pointwise AND between all combinations of elements of sets [Zadeh 1973].

The main characteristics of the model are: weight sharing, aggregation of inputs and non-decreasing (differentiable) nonlinearity on output. (Observe that for example for 3 inputs $\{x1, x2, x3\}$, the power set is $\{\{x1\}, \{x2\}, \{x3\}, \{x1, x2\}, \{x1, x3\}, \{x2, x3\}, \{x1, x2, x3\}\}$ (the empty set was omitted), and for each element of the set there is a synapse weighting the contribution of the element). By analogy, one can define a fuzzy logic counterpart of the model, by replacing Π with T, and $\sum$ with S. However, instead of operating on the power set, we restrict to the Cartesian product of some subsets, which may correspond to classes of similar inputs. In other words, the number of inputs associated with a shared weight is fixed, and equal to the number of classes.

Definition. The *S-T activation function* is defined by the function y, $y:[0, 1]^m \rightarrow [0, 1]$, of the form

$$y(x_1, \ldots, x_m) = \underset{E_j \in \underset{k=1}{\overset{m}{\times}} F_k}{S} (T(\underset{i \in E_j}{T} x_i, w_j)), \tag{4.4}$$

where S is an s-norm, T is a t-norm, and F_k are classes (in a total of m) of inputs. The neuron operating on (4.4) is called *S-T fuzzy neuron with shared weights* abbreviated as FNSW.

Fig. 4.1 illustrates a neuron with inputs coming from two classes **x** and **v**, each synapse having exactly two inputs (i.e. inputs x_i and v_j share the weight w_{ij}). The total number of weights is given by the cardinality of the Cartesian product between the two sets of inputs. For a layer of FNSW, an element of the relation matrix W is a weight w_{ijk} connecting the input samples x_i, v_j, to the k-th neuron. To produce the output Y, each k-th neuron performs a somatic operation (S) on synaptic inputs $T(x_i, v_j, w_{ijk})$, for all i,j combinations. Considering m input classes, it is possible to implement m-dimensional mappings (Equ. (4.2)) with one layer of fuzzy neurons with shared weights described by (4.4).

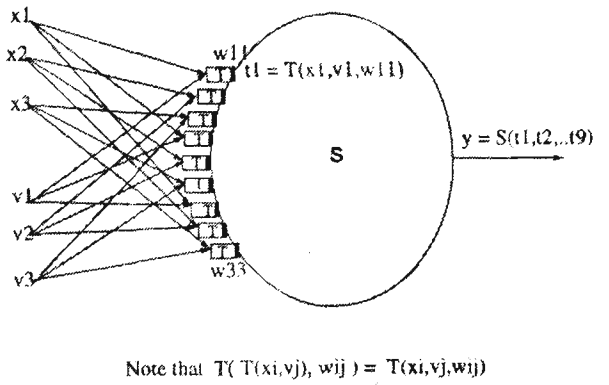


Figure 4.1: S-T fuzzy neuron with shared weights

4.2 Learning multi-dimensional mappings

In this section it is shown how a fuzzy neural structure based on S-T neurons with shared weights can learn the mapping from multi-input variables to an output variable. The neural weights obtained after learning configure a distributed rule-table which reflects the initial rule-table used to generate the I-O training sets. In practice, the data is obtained from the unknown system subject to modelling, and the network performs a knowledge extraction.

FNSW can learn by the same mechanisms as FFN. This is made possible by the fact that one can calculate a combined input (a combination of individual inputs by a t-norm) which is modulated by the shared weight. Having the combined input as a unique input for the weight, the FNSW reduces to a FN.

Example 1. Consider the fuzzy system with two inputs and one output, with membership functions illustrated in Fig. 4.2 and the mapping of the fuzzy sets given by Table 4.1.[2]

[2]The abbreviations of fuzzy set labels stand for the following: N - Negative, Z - Zero, P - Positive, NB

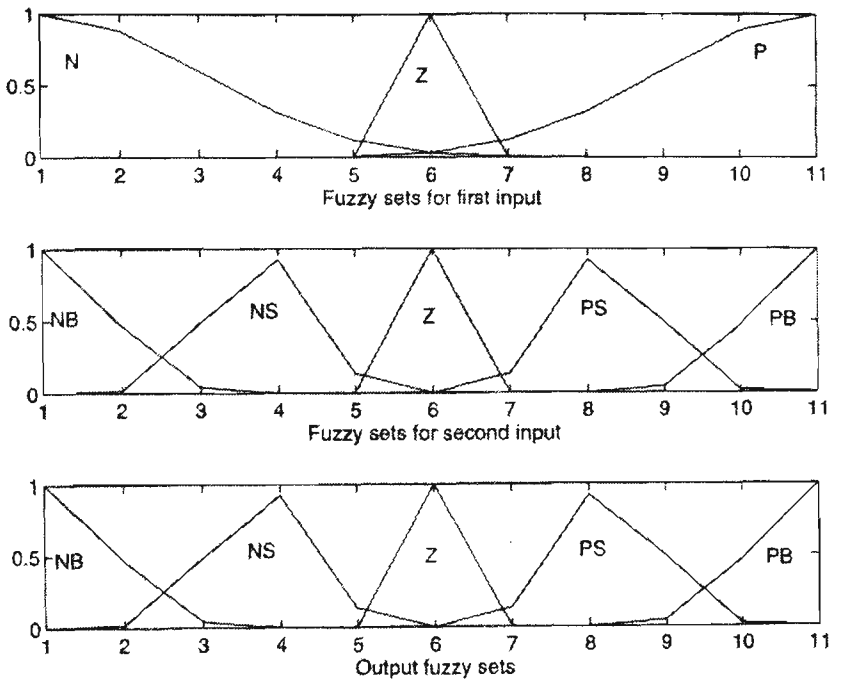


Figure 4.2: Membership functions for inputs and output

At the chosen resolution of 11 points per definition domain, the neural model consists of 11 neurons (associated to points in output domain), each having a maximum of 121 active synapses. The training set was obtained from the mapping of discretized sets. The mappings to be learned are from 2D input fuzzy sets, obtained by pointwise logic AND of two input fuzzy sets, to 1D output fuzzy sets. For example, one training pair is determined by the first rule in Table 4.1. The pair represents the mapping between

Table 4.1: Rule table for the two input system

	*In*1		
*In*2	N	Z	P
NB	NB / R1	NB / R2	NS / R3
NS	NB / R4	NS / R5	Z / R6
Z	NS / R7	Z / R8	PS / R9
PS	Z / R10	PS / R11	PB / R12
PB	PS / R13	PB / R14	PB / R15

- Negative Big, NS - Negative Small, Z - Zero, PS - Positive Small, PB - Positive Big.

T(N,NB) and NB and is illustrated in Fig. 4.3. Each of the 11 neurons receives inputs from all the input cells of the bi-dimensional fuzzy set in Fig. 4.3. In computations, the matrix was transformed into a long vector by appending consequent rows to each other. For example, the 11 by 11 bi-dimensional fuzzy set 'N AND NB' in Fig. 4.3 was transformed into a 121 element vector, which is the bottom line array (on the left) in Fig. 4.4. Similarly, the 11 points of the fuzzy set 'NB' in Fig. 4.3 become the bottom line in the right array in Fig. 4.4. The grey level reflects the magnitude of the element: the bigger the element the darker its representation. Thus for each rule a vector was mapped to a vector, obtaining the training pairs shown in Fig. 4.4.

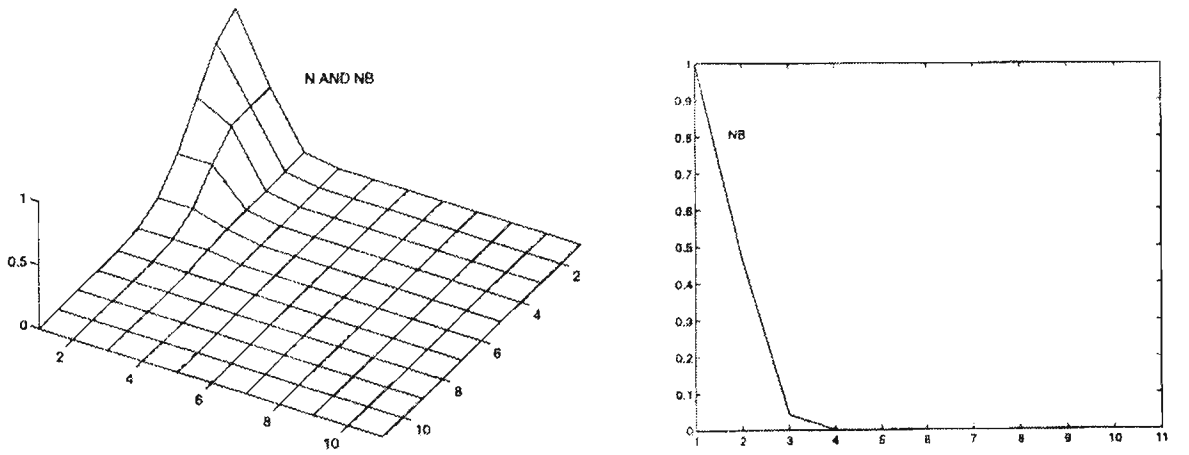


Figure 4.3: 'N AND NB' maps to 'NB' (2D to 1D fuzzy set mapping)

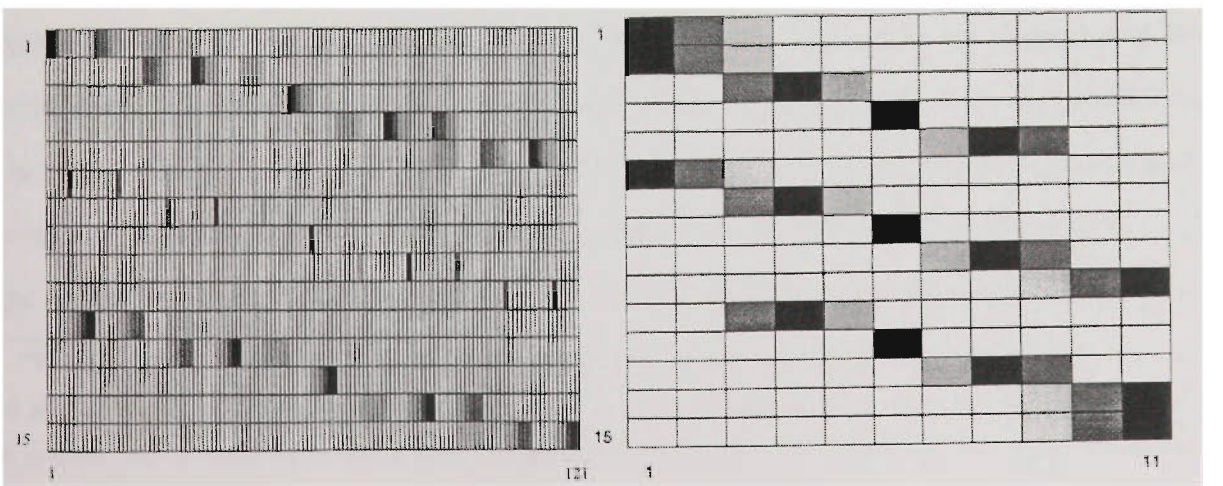


Figure 4.4: Visualisation of the training set: inputs map to outputs. The bottom line corresponds to the training pair shown in Fig. 4.3

The resulting fuzzy relation is of size (121,15) [3]. After learning, the neural structure encodes the knowledge from rules and membership functions in the synaptic weights. 'Slices' of weight space look like distributed rule tables as shown in Fig. 4.5, and reflect the separated rule tables for each output set shown in Table 4.2. A FNSW from the set of 11 performing the mapping is detailed in Fig. 4.6. Potentially the model suffers from the 'curse of dimensionality', the number of weights increasing exponentially with the number of inputs. However, as seen in Fig. 4.6, few of the 121 possible connections are active. To ensure a reasonable number of connections, a competitive learning mechanism can be imposed, thereby limiting the number of allowed synapses in a trade-off with the modelling precision desired. For an alternative approach to distributed modelling in fuzzy systems the reader is referred to [Pedrycz *et al.* 1995] [4].

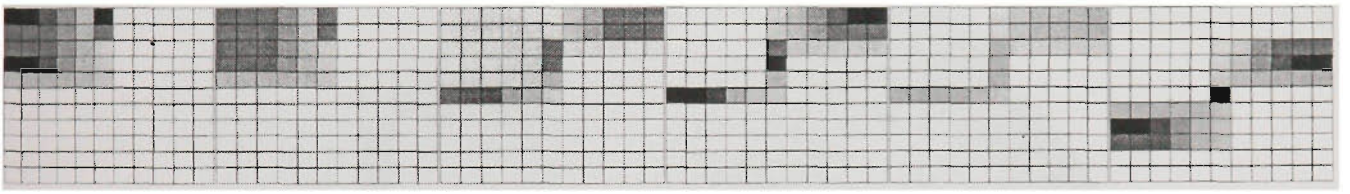


Figure 4.5: Projections in weight space for neurons 1-6 associated with the first 6 points in the discretised output domain. Darker colors proportionally represent larger weights.

Excitatory and inhibitory inputs. The inputs to any type of fuzzy neuron can be *excitatory* (x_i), or *inhibitory*, in which case they are *complemented* ($1 - x_i$). By inhibitory it is meant that an increase on that input will reflect in a decrease on the output, while excitatory inputs increase the output. The definition of FNSW does not specify the type of t-norms used for S and T, however for the reasons presented in Section 3, the

[3]The neurons are independent and can be trained independently to solve $y_j = X_k \circ R_{kj}$ for all k = 1,...,p, and all j = 1,...,n. This reduces the computational burden of training a network with 1331 synapses, to the simpler task of training neurons with 121 synapses. However, one must realize that the freedom of finding different synaptic and somatic parameters for each neuron offers greater approximating power at the cost of a non-uniform reasoning (composition) law.

[4]According to the classification proposed in Section 2.3 the approach proposed by Pedrycz is not distributed, but compact.

Table 4.2: Rule tables for outputs NB, NS, Z.

NB	*In*1			NS	*In*1			Z	*In*1		
*In*2	N	Z	P	*In*2	N	Z	P	*In*2	N	Z	P
NB	R1	R2		NB			R3	NB			
NS	R4			NS		R5		NS			R6
Z				Z	R7			Z		R8	
PS				PS				PS	R10		
PB				PB				PB			

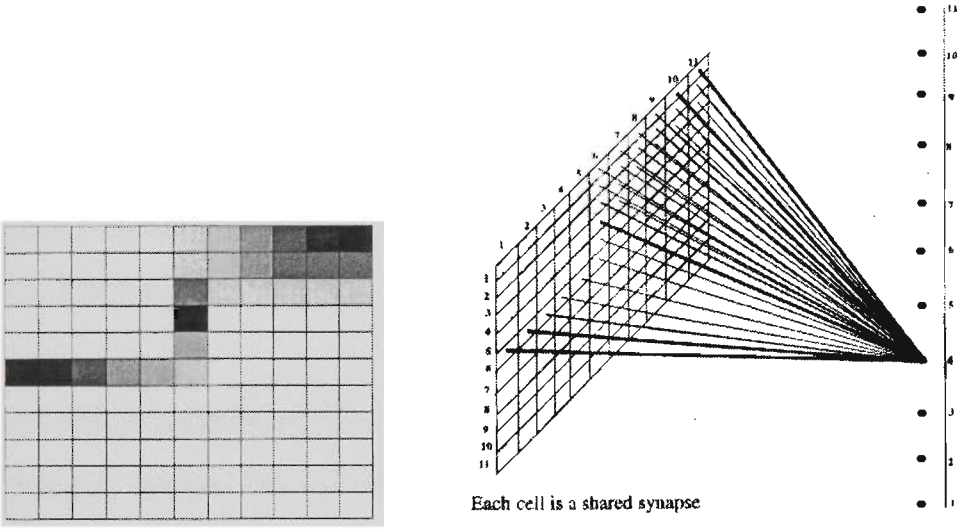


Figure 4.6: A FNSW and its weight space

fundamental t-norms are preferable and in the following FNSW are assumed to be fundamental.

4.3 Implementation of logic functions

Proposition. Any boolean function (and in consequence any logic gate) can be implemented with a single FNSW, working under any fuzzy logic. (Otherwise stated, any boolean function is vertex equivalent to an S-T activation function whose weights are all

either 0 or 1.)

Proof: The above follows directly from the fact that any boolean function can be expressed in a disjunctive normal form (DNF). The DNF contains one term for each output TRUE, represented by 1 in the complete truth table. Its general form is y = OR ((x$_1$ AND ... AND x$_m$) AND w$_i$), where OR is taken on all pairs x$_1$,...,x$_m$, for all possible combinations of inputs and their conjugates. As S generalises the boolean OR, and T generalises the boolean AND (they become the same for binary arguments), the expression can be written y = S((x$_1$ T ... T x$_m$) T w$_i$), which is equivalent to Equ. (4.4).

Example 2. The following shows the FNSW implementation of the XOR. The XOR table (Table 4.3) leads to y = (x$_1$ AND $\bar{x}_2$) OR ($\bar{x}_1$ AND x$_2$). This is a particular case for y = (w$_1$ AND ($\bar{x}_1$ AND $\bar{x}_2$)) OR (w$_2$ AND ($\bar{x}_1$ AND x$_2$)) OR (w$_3$ AND (x$_1$ AND $\bar{x}_2$)) OR (w$_4$ AND (x$_1$ AND x$_2$)) with w$_1$ = w$_4$ = 0, w$_2$ = w$_3$ = 1.

In the simulations performed the neuron learned the XOR solution in one GD step. More precisely it learned the OR function, as the inputs for training come after the neuron preprocesses the inputs by pointwise AND, with the T^* operation given by $T^*\{\{x_1,\bar{x}_1\},\{x_2,\bar{x}_2\}\} = \{x_1 T x_2, x_1 T \bar{x}_2, \bar{x}_1 T x_2, \bar{x}_1 T \bar{x}_2\}$ as in Table 4.4[5]. The FNSW implementation of the fuzzy XOR and its characteristic for s = 0.01 are illustrated in Fig. 4.7.

[5] $\bar{x}$ is *not* x, calculated by $1 - x$.

Table 4.3: XOR logical table

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.4: Training set for neuron implementing XOR

$\bar{x_1}$ T $\bar{x_2}$	$\bar{x_1}$ T x_2	x_1 T $\bar{x_2}$	x_1 T x_2	y
1	0	0	0	0
0	1	0	0	1
0	0	1	0	1
0	0	0	1	0

4.4 Implementation of various connectives and relation with other fuzzy neuron models

FNSW can be reduced to implement the T operation (which models the logic AND) by having a unique synapse with multiple inputs. Alternatively, FNSW can be reduced to implement the S operation (which models the logic OR) by having only one input per weight, and all the weights equal to 1. Configurations in between these two extremes allow intermediate logical characteristics between AND and OR (as 'pure' AND and OR may not cope well with experimental data [Hirota and Pedrycz 1994]) and thus it exhibits a similar functionality to the OR/AND neuron (a layered arrangement of OR and AND neurons), proposed in [Hirota and Pedrycz 1994] as a generic model of local connectives[6].

[6]In knowledge-based applications, the concepts are often linked by *and* or *or* connectives. Some concepts are strongly related to others, while others are almost unrelated. For example *and* in 'red *and*

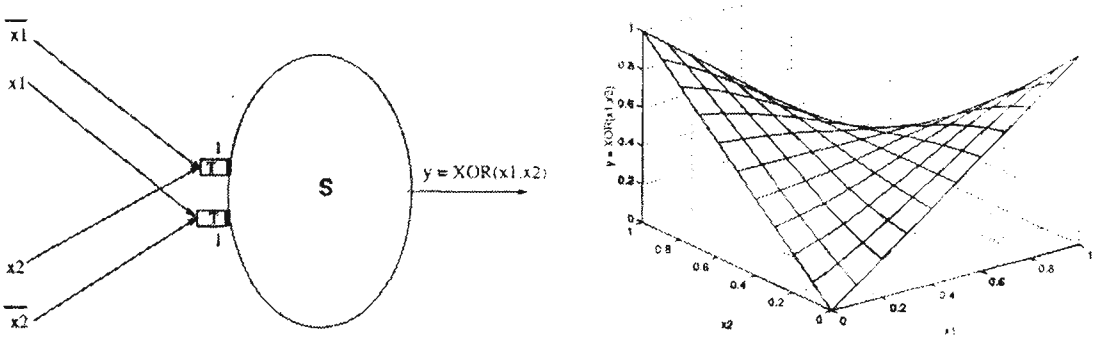


Figure 4.7: FNSW implementing XOR, and the fuzzy XOR surface

Similarities can be observed between the FNSW as proposed here and the SOM (sum of miniterms) architecture of logic processors [Pedrycz and Rocha 1993], [Pedrycz *et al.* 1995]. SOM is a network with a layer of AND neurons ($z_j = T(v_i S x_i)$) which realize the miniterms, and a unifying OR neuron ($y = S(w_j T z_j)$) which does the sum of miniterms. FNSW stays between the OR neuron (to which it can reduce by not allowing shared weights (i.e. having one input per weight) and the SOM network which reduces to FNSW if some weights in the AND neurons are 1's, and some are 0's.

4.5 Summary

The Chapter extends the fuzzy neural models to allow implementations of multi-input fuzzy systems. For this purpose, the fuzzy neuron with shared weight was defined. The multiple inputs that share the same weight come from different input variables. Thus

new car' is different to *and* in 'large *and* expensive car' (example from [Di Nola *et al.* 1989]). In the first construction the properties are unrelated, whereas in the second case the properties are related. In each case a different t-norm may be the most appropriate for modelling. T-norms (other than MIN) have compensatory effects (i.e. $t = T(x_1, x_2)$ can be obtained for a different x_1 by a change in x_2). It is also possible to define compensative logic operators (such as the 'compensatory and' or 'γ operator' [Zimmermann 1991]), for which the aggregated output is between 'pure' *and* and *or*.

a layer of FNSW is a direct implementation of the multi-input system. An example of learning a neural representation for a two input system is given. As in the case of single input systems, the weights reflect distributed rule tables, and such an organization obtained through learning is in fact a rule extraction mechanism. It is proved that one FNSW can implement any boolean function and can simulate several types of fuzzy neurons, qualities that recommend it as a potential general purpose computational unit.

Part II

Towards robot apprentices

This part initiates a new direction in robotics research: that of anthropomorphic robots learning to move by imitating human movements. Imitation is based on vision, and employs neural and fuzzy neural models of eye-arm coordination. Results of learning are demonstrated in 2D for a robot imitating human arm movements, and in 3D for the robot imitating the movements of a second robot, of identical construction.

The researcher interested in fuzzy systems can treat this second part as providing application results for fuzzy neural models treated in Part I. On the other hand, the researcher in robotics would have had treated Part I as providing theoretical details for the computational methods employed for solving a problem in robotics.

Chapter 5

Learning arm movements from a human instructor

The first part of this chapter reviews results which contribute to an integrated approach to motor development in anthropomorphic robots. The focus is on learning the visuo-motor coordination and on skill transfer. The second part of this chapter presents a scenario on learning arm movements from a human instructor. It argues in favor of developing eye-arm coordination for motor skill acquisition in anthropomorphic robots. Once the eye-arm coordination is learned, the robot can imitate the human arm movements for solving a particular task. To provide the training examples necessary for learning eye-arm coordination, a technique in which the human imitates the robot is proposed.

5.1 Towards an integrated approach to robot motor learning

In the last few years a number of papers have described research in the area of developing *sensory-motor coordination* for robotic manipulators. Some researchers use robots as a vehicle for the investigation of learning systems and behavior, while others are mainly interested in developing systems which by learning become better adapted to their working environments, and cope easier with changes is such environments or their own structure. Some representative systems that have the development of sensory-motor coordination as one of their goals are INFANT [Kuperstein 1991], MURPHY [Mel 1991], and the DARWIN series of automata [Edelman *et al.* 1992] (see also [Reeke and Sporns 1993] for a review). Different in many respects, and particularly in the purpose for which they were built [1], these systems share the use of a mechanism which Piaget called *circular reaction*, and which consists in correlating self-generated actions with consequent perceptions. Thus, these robots learn by exploration, without having an initial model, for example by flailing their hands and perceiving the effects, and are subsequently able to achieve the coordination that enables them to grasp and track moving objects.

Most of the work in learning visuo-motor coordination by robotic manipulators is related to the idea of obtaining an association between the hand position in the image and the corresponding joint values that determine the positioning of the hand at that position [Kuperstein 1991], [Martinez *et al.* 1989], [Graf and LaLonde 1989], [Walter and Schulten 1993], [Smagt *et al.* 1993]). The learning structures are neural based and in general the image is preprocessed in order to locate the center of the hand/object (its

[1] INFANT was addressed at learning visuo-motor coordination of a multi-joint arm able to grasp objects in the 3D space. MURPHY was focused on neurally-based visually guided reaching, which includes movement planning in an environment with obstacles. In the DARWIN series of automata the emphasis was on demonstrating learning based on Neural Darwininsm, Edelman's theory of neural group selection [Edelman 1987].

x,y coordinates *in the image*). The neural network learns the mapping between center coordinates and joint angles, mapping which is a particular solution for the inverse kinematic problem[2]. When two cameras are used the two pairs of coordinates of hand in the image are combined in a four-dimensional vector, for example as in [Walter and Schulten 1993]. In the case of redundant manipulators there is more than one configuration of the arm which can be chosen for placing the end-effector in a desired position. Teaching a particular configuration avoids the problem of choosing a solution from a set of alternatives. Alternative methods include, for example, the choice of a solution (i.e. arm configuration) at random , making a choice based on previous configurations, or based on the optimisation of some index function, etc. Restrictions on the internal joint coordinates can be made based on appropriate criteria such as distances of the links from the boundaries, as well as from external obstacles. In [Palm 1992] the arm avoids obstacles whereas the end-effector follows a planned path. Distances and corrections are denoted as fuzzy terms, and various criteria for controlling the arm are formulated linguistically by fuzzy production rules. In [Guez *et al.* 1992] the solution to the inverse kinematics problem attempts to capture the solution (and obey the same criteria) that a human used when asked to move an object in free space, in a plane parallel to the ground.

The common approach to visually-guided manipulation assumes that there are no obstacles in the environment. If obstacles exist than the solutions obtained by eye-arm coordination models can fail, and motion planning is necessary. In MURPHY [Mel 1991] motion planning was possible by learning a forward model of the arm, and building 'mental' images of the arm, which were used in an error minimization try-and-error search for a path to the target. To learn the forward kinematics function MURPHY stepped his arm through a uniform sample of approximately 17,000 arm configurations.

[2]Traditionally, inverse kinematics refers to the mapping between the hand position in *world coordinates* and the joint commands. In the case of eye-arm coordination the mapping is between the hand position in *camera coordinates* and arm commands.

In general, learning of sensory-motor coordination is characterized by the following:

- it is self-controlled and the predominant form of learning is learning by exploration
- the input patterns (models of movement) are not provided by a human teacher, and there is no reinforcement from a teacher
- the desired movement is not known in advance (in terms of trajectory), but it is subject to a selection process and its acceptance depends on the particular task
- it makes no use of prior experience from external sources.

In contrast, *skill acquisition* is characterized by:

- learning is mainly human-guided (possibly by reinforcement), and the predominant form is learning from examples or hints
- the input patterns are selected by a teacher and the error signals and reinforcement are produced mainly by the teacher
- the desired movement is explained (shown), and its reproduction is attempted
- it makes use of the teacher's experience.

Work in computational models of human skill acquisition and their application to robots was reported by Gelfand, Handelman and Lane in [Lane *et al.* 1990].

Inspired from a classic classification of phases of human motor skill acquisition [Fitts and Posner 1967], they proposed a model based on a knowledge-base in which rules about movement already exist.

In the beginning there is a Declarative Phase in which a knowledge-based execution monitor determines how to accomplish a given control task using rules and algorithms within the KB. In the Hybrid Phase the KB system supervises the training of a NN which gradually takes responsibility for control. In the Reflexive Phase, reinforcement learning takes over to further refine system performance (trajectory, etc). In more recent work [Gelfand *et al.* 1992], machine vision input was used for planning and executing movements under an algorithmic controller, while a NN learned the control using sensory feedback. The accent in these constructions is on the transformation of representation, mainly on the transfer from the KB to the NN. From the perspective of the approach adopted in this thesis, their work has the limitation that the knowledge about the movement is assumed available. However, in practice, obtaining the knowledge into the KB is usually the most difficult problem.

Contributions in the direction of knowledge acquisition come, for example, from the work of Asada and colleagues (see for example [Asada and Liu 1990], [Asada and Liu 1991]), who investigated the possibility of acquiring task performance skills from human experts. In their reports they describe how an operator demonstrated the task (a deburring task) and a set of signals, including forces and positions, were recorded. The data was used to train a neural network for performing the mapping between process parameters (such as material properties and workspace geometry) and control actions (i.e. tool manipulation parameters). The training was done off-line. In more recent work [Liu and Asada 1992], the operator linguistically expressed his strategies while he was performing the task. The hybrid numeric-linguistic system, structured around if-then rules which form a collection of local control strategies applying to a particular situation, lead to a more efficient system then its neural predecessor. The focus in Asada's work is on learning the task and not the arm movements to perform the task.

Automated learning of tasks by a robotic system through observation of a human operator was also considered by Belmans [Belmans 1990]. He modeled a task by an

error function between the actual state of the robot and the reference state (or reference trajectory) it should occupy, and the robot was required to minimize the error function. This servo-control perspective is analogous to telemanipulation.

In telemanipulation the human moves its arm, which has attached a master arm, and its movement is reproduced by the slave arm. It is important to have a suitable impedance[3], however, what is the optimal impedance for master-slave teleoperators is still a matter of debate (see [Sheridan 1992] for a review). Some argue that an ideal teleoperator is one that is transparent, i.e. infinitely stiff and weightless mechanism between slave and master arm end effectors. It is also important to note that operators get tired when holding their arms in awkward positions or applying constant forces, as master-slave systems often require. A camera-driven telemanipulation may be thought of as a solution to the above problems, in which nothing is attached to the arm and there is a complete freedom of movement, but which inherently requires more complicated processing[4]. In this way the approach becomes connected to a form of visual-servoing.

Image based visual-servoing is a relatively recent approach to control, characterised by closing the control loop around visual inputs [Shirai and Inoue 1973], [Sanderson and Weiss 1986]. This is part of a novel trend to use noncontact sensors inside the servo-loops themselves, while initially their use was limited to providing data for higher level decisions, that involved mainly pattern recognition problems[Espiau *et al.* 1992]. The reported applications in robotics, target achieving the proper position and orientation of end-effector in respect to an object. The common technique is 'eye-in-the-hand', the camera being mounted on the end-effector. Some dynamic effects present in this approach, such as the perspective gain effect, manipulator vibration, and latency from servo-error to action are discussed in [Corke 1992]. In the approach proposed in this

[3]Impedance is defined as the relation between applied force and the velocity.

[4]Note that camera-guidance as proposed later in this chapter, there is no force information communicated and also no force feedback, which normally in teleoperation increases the task completion time. The information refers to the trajectory, or the succession of movements necessary to successfully accomplish a manipulation task.

thesis visual servoing is used by a robot arm as a mean of tracking/imitating a master arm. The cameras are not fixed 'in-the-hand', but are positioned as to take a 'bird's eye' top view, and thus the only important dynamic effect is the time delay between image acquisition and the placement of the arm in the desired position. As for a real-time vision system used in conjunction with a robot, an excellent example is the ping-pong player robot, initially designed for testing a high-speed camera system [Anderson 1988].

Some robots have used vision-based systems for task learning from practice. For example, a robot was programmed to juggle a single ball by batting it upwards with a large paddle [Atkeson 1990]. The robot used a real-time binary vision system to track the ball and measure its performance. A model of performance errors was built at the task level during practice and used to refine task-level commands. A recent thesis by Schneider [Schneider 1995] deals with robot skill learning by intelligent experimentation and proposes some new algorithms for accelerating learning from practice. In these systems learning is based on an individual effort and is not the result of a skill transfer from a coach (Schneider uses a kind of 'virtual coaches' - particular algorithms that each improve some part of controller's performance).

Teaching and learning bear a close relationship, and it is important to develop not only good learning abilities, but also good teaching schemes, which allow the skill transfer to be coordinated by the human. Coaching is thus very important in developing motor skills. Some important aspects of coaching, and in general of motor skill transfer to humans, are presented in Appendix 7 [5].

From a human's point of view, the ideal way of communicating with robots would be natural language. An example of a system that learns from natural language instruction is Instructo-Soar [Huffman and Laird 1994]. The system starts with a small set of

[5]Their placement in an Appendix is because they do not directly affect the results presented here, however, they are of particular importance for developing robot apprentices that learn in similar ways that people do, and could be very useful in future work continuing the approach presented in this thesis.

primitive operators and learns completely new procedures from sequences of interactive instruction. At the first execution of the procedure, everything that changed from the initial state to the final state during execution is associated with the goal of the procedure. In following executions, the system recalled instructions learned by rote and explained to itself how each contributed to achieving the goal.

Natural language instruction of future robots should be done using spoken language. A precursor of future spoken language interfaced robots is the Speech Activated Manipulator (SAM) [Brown *et al.* 1992]. SAM lives in a complex world, has different types of sensors and communicates with the human teacher via spoken natural language. Through combined human-machine and machine-world interactions SAM substantially reduced the amount of knowledge and skill needed by a human operator.

In the process of skill acquisition, the movement can initially be performed at a lower speed than desired, and then subsequently repeated at increasingly higher speeds, until the desired one is reached. The practice control strategy presented in [Sanger 1994] could be used to mathematically formalise such an approach. Instead of simply storing the motor sequences, specialized structures should be deviced to learn the motor patterns. The learning of motor patterns can be addressed in relation to the learning of a complex temporal sequence, for which a neural network solution was proposed in [Wang and Arbib 1990], [Wang and Arbib 1993]. The storing and generation of visually acquired two dimensional trajectories by a motor program generator was described by Eckmiller [Eckmiller 1990]. He suggests a way in which a robot with NN modules can learn to draw a visually monitored pattern. The example given is a typical writing trajectory - the letter 'b' - and the corresponding hand trajectory in 2D.

The acquisition of motor skills by a system which has learned its sensory-motor coordination on its own is a largely unexplored area. It is also an important step towards an integrated approach to robot motor learning. The remainder of this chapter is intended

to offer a possible scenario on how this may be achieved.

5.2 Learning arm movements from a human instructor: an approach based on imitation

Learning as a single entity experience has received appropriate interest in the robotic, machine learning and artificial intelligence communities. Much less studied has been learning from similar entities. Half a century ago, Turing wrote 'The isolated man does not develop any intellectual power. It is necessary for him to be immersed in an environment of *other men*, whose techniques he absorbs...'. One particular thing humans transfer to each other is skills.

Skills (of a qualified operator, sportsman, etc) can be broadly divided into two large categories: planning skills, i.e. the know-how expertise, and motor control skills, i.e. the ability acquired after performing a movement many times. Accordingly, the mapping between process characteristics and actions can be divided in a mapping between process characteristics and desired actions (which determine the planning skills), and the mapping between desired actions and performed ones (which corresponds to motor control skills). The former are related to strategies at a higher level, while the later refer to dexterity and motor abilities (see Fig. 5.2). In the case of arm coordination the mapping between desired and actual performance is subject to a representation in which a motor controller maps the desired performance into commands, and the arm plays the role of the controlled plant, mapping commands to actual performance. The motor controller also performs a coordinate transformation, from a sensory coordinate system to a motor coordinate system.

A transfer of motor skills refers mainly to the transfer of the mapping between desired

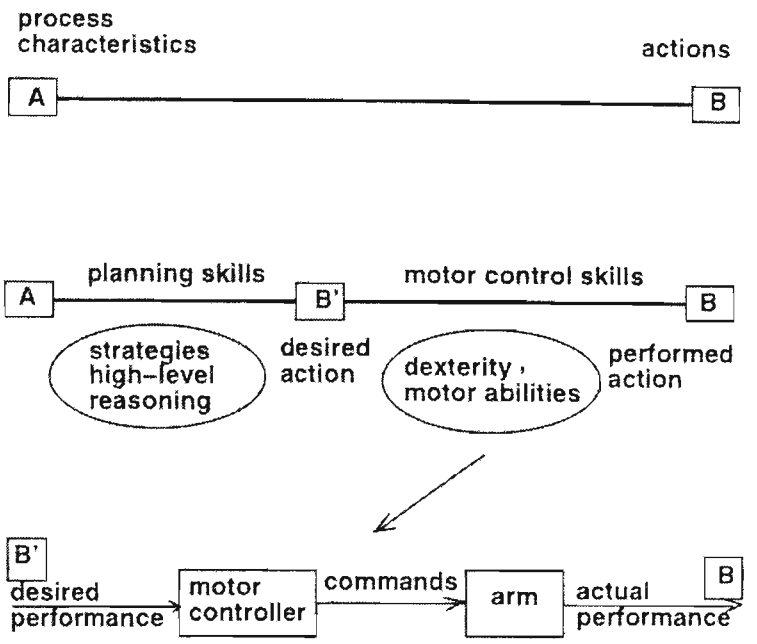


Figure 5.1: Planning skills and motor skills

actions and performed ones. If the robot can 'see' the desired movement performed by a human, and can imitate it, then there is no need for the robot to be very intelligent and figure out for itself 'how' to perform the movement. There is no need for task understanding, path planning, etc; the human offers the solution for motion planning and all that is needed is to imitate human's movement (and repeat as necessary) [6]. Teaching by showing is also for the human the simplest way of instructing a robot.

Imitation allows the robot to get the solution of a movement problem in terms of its own internal representation of motor commands. Ideally this should be followed by some associative processes, the robot learning that it was useful to perform certain movements (shown by human) in a certain context. Two alternative ways in which the robot can imitate the human were tested in simple experiments. In one alternative the robot looked alternatively at the human arm and at his own arm, and tried to reproduce

[6] A very interesting experiment communicated recently [Nagell *et al.* 1993] shows a major difference in the way imitation is performed by humans and chimpanzees respectively. On a tool demonstration task, the chimpanzees retain the general functional relations in the task and the results obtained, but not the actual methods of tool use demonstrated. On the contrary, human children were reproducing demonstrator's actual methods of tool use. Interesting enough, it is the human children who 'ape' and not the chimps !

human's arm posture by canceling a positional error between the two[7]. The vision system has to continuously switch focus between instructor's arm and own arm as to react on differences, and therefore fast movements could be difficult to track in this way. Another way of imitating is by watching the teacher's arm only[8]. In order to imitate, i.e. to place its own arm in postures similar to those of the human's arm, the robot must have an *eye-arm coordination* which associates the images of the human arm in different postures to the commands to own arm.

Two questions naturally arise at this stage. First, why eye-arm coordination and not eye-hand coordination (the form of visuo-motor coordination which has been attempted so far in robotics research)? Second, how can one learn the eye-arm coordination which associates images of human arm to commands of own arm? These questions are answered in the following.

5.2.1 Eye-arm coordination

For redundant manipulators (including here the human arm and anthropomorphic robot arms) the associated inverse kinematics problem is underconstrained, admitting more than one solution (i.e. more than one set of joint values can place the hand at a specific point in space) and some alternatives to cope with this were mentioned in Section 4.1.1. However, none of these alternatives is acceptable if the given task requires specific arm postures, as imposed by obstacles in the environment, or by the task itself. This is exemplified in Fig. 5.2, in which the learned solution (Posture 1) for placing the hand in a particular point is unacceptable due to an obstacle, while a posture shown by

[7]This could be regarded as a classic 'tracking' problem in a control perspective. For a desired 'reference' configuration of arm and an actual configuration for own arm, the error (difference between the two) is the input value for a controller. In some initial experiments, based on arm contour detection, this solution was tried by using a simple proportional controller.

[8]This control is open loop, with no information about robot's own arm coming from the cameras (a model of the robot's own arm can be used to predict the positions).

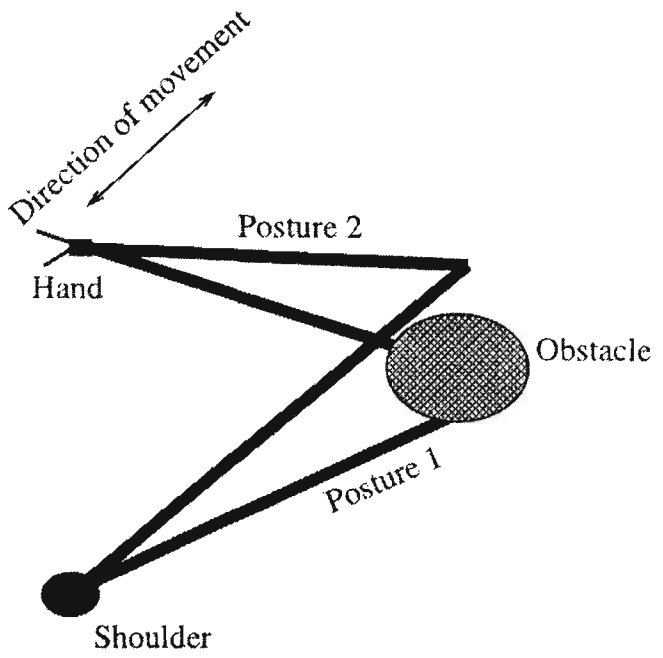


Figure 5.2: Two possible arm postures for performing same hand movement, the first one forbidden by an obstacle

an instructor (Posture 2) provides a feasible (and in general quasi-optimal) alternative. Eye-arm coordination is thus necessary for skill transfer to redundant manipulators, determining successive positions of the arm and not of the hand only, and adds to other models of coordination as shown in Fig. 5.3.

In many situations which do not pose external spatial constraints, humans find eye-hand coordination more efficient for manipulation tasks. The choice of a solution for the control of the redundant human arm is determined by internal constrains of energy minimization, smoothness of movement, comfort (biomechanics studies show that positions around the middle of the permitted range are preferred), etc. In tasks that require learning to move a hand held object, only the trajectory of the instructor's hand is watched for, and eye-hand coordination appears sufficient. However for learning skilled movements, in working environments that contain obstacles or other spatial constrains, it is necessary to watch the movement of the whole arm, and this is the aspect treated here. Future systems could benefit from a combination of eye-hand and eye-arm coordination.

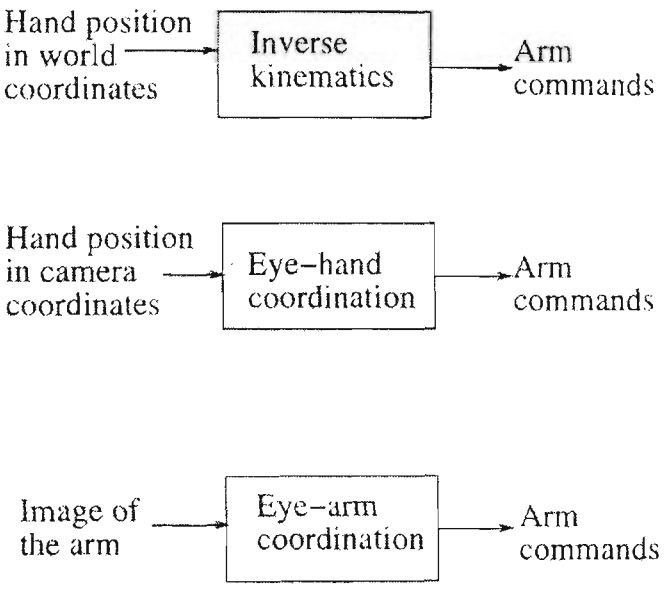


Figure 5.3: Models of arm coordination

Eye-arm coordination is addressed here as a direct mapping between the configuration of the whole arm, as seen by the camera, and joint commands. In this respect it is more related to ALVINN (Automated Land Vehicle In an Neural Network) used in NAVLAB [Pomerleau 1993] than to any work in sensory-motor coordination [9]. ALVINN is a three layer feedforward neural network trained to associate images of the road to commands of the steering wheel. Examples for training the net are obtained 'on the fly' while driving, or from computer generated situations which cover situations rarely encountered in correct driving. In an analogy to ALVINN, the arm in the image corresponds to the street as seen by the camera, while the commands to joints have as correspondent commands to the steering wheel. Three major differences distinguish the approach presented in the following from ALVINN. First, there is more than one output variable to be controlled (here shoulder and elbow angles - and z displacement for 3D tests). Second, after the successful use of classic NN trained by batch backpropagation, it is investigated the use of a fuzzy NN, which performs incremental learning at the presentation of

[9]One can attempt to directly extended the procedure used in eye-hand coordination, i.e. finding centers of hand, lower-arm, upper-arm, but this may mean heavy preprocessing and a careful correlation between different parts. Also possible is to find the skeleton of the arm, and to use this information for mapping. My first attempts were oriented in this direction but the image processing burden and the dependence of the results on carefully controlled laboratory conditions determined the search for better solutions.

each example. The last, and most important difference is the method of providing the examples necessary for supervised training, which is presented in the following section.

5.2.2 The human imitates the robot

How can the robot learn what commands to give to its own arm in order to produce a posture similar to that of the human arm? In building associative models, the associations were generally between commands and results of commands *through the same system*. This idea of associating actions with determined perceptions was applied for example, in INFANT [Kuperstein 1991] in the context of learning the eye-hand (own hand) coordination. In the context of learning autonomous driving with ALVINN, the images of the street were associated to certain rotations of the wheel, the effect of direct human action. The human performance of the task offers directly the input-output data set, which can be used in supervised training for deriving an associative model. This method appears much more difficult to apply to skill transfer at manipulators. To quote ALVINN's builder 'The same techniques would not be readily applicable to domains such as the controlling individual joints of a robot arm...since in these tasks the correct response is difficult to determine' (p. 175 in [Pomerleau 1993]).

The solution proposed here is that *the human imitates the robot*, positioning his own arm in postures similar to the ones produced by the robot. The robot gives a certain command set to its joints, and as a result its arm takes a certain posture. The human tries to place his arm in a posture similar to the robot's. Then it validates the image of his arm seen by the robot's eye which becomes example for training, being associated with the robot command set[10]. The method is subject to human error in the estimation

[10]One can draw an analogy to this technique, relating it to the fact noticed by child psychologists that quite often parents happily reproduce the first attempts of speech of a baby, although these are not proper words of a language; this provides the child with some feedback, which some researchers consider important in learning. (The teacher imitates and reinforces imitation. The child learns that imitation is

of identical or most similar posture. The two arms are not identical, and the metric according to which the two positions are similar is not formally defined, the assessment being subjective. Accordingly, the method can be best evaluated based on some 'posture similarity' criterion.

Thus, in the approach proposed here, the main steps in developing robot apprentices that learn from shown movements are considered:

1. learning of a visuo-motor model of arm coordination,
2. imitation of the teaching arm,
3. correlation of the solution with the task and repetition of the movement to optimize parameters and develop a motor pattern.

Only the first two aspects receive treatment in the following. The human imitates the robot and this offers a solution for developing an association between images of the human arm and the commands that the robot gives to position its own arm in similar postures. In the second stage, the robot imitates the human and thus receives a model of the movement it needs to learn (see Fig. 5.4).

5.2.3 Experimental framework

A series of experiments were conducted in order to validate the fesibility of the proposed approach. To compensate the lack of a true anthropomorphic manipulator (but still maintain real world conditions) two types of experiments were conducted.

---

nice/rewarding.)

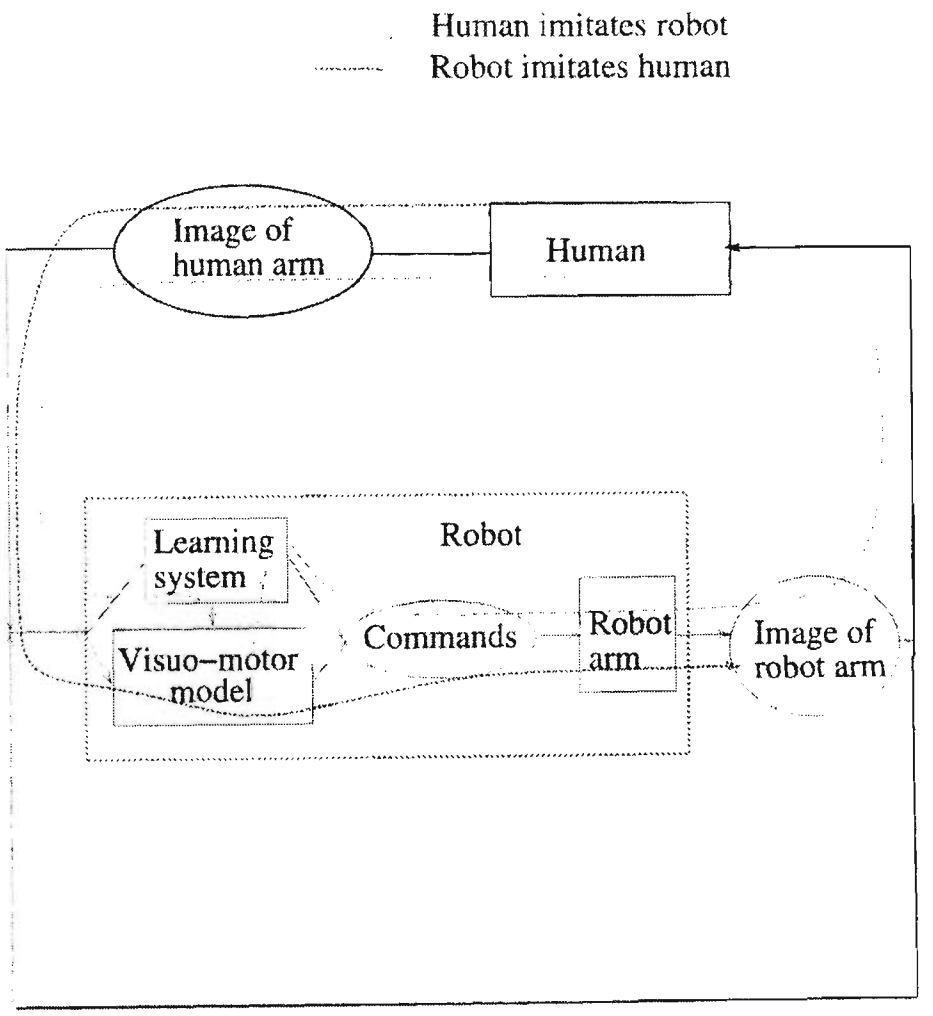


Figure 5.4: Imitation by human and imitation by robot

A first type of experiments involved a robot imitating human arm movements performed in a horizontal plane. Viewed from above, and while performing in a horizontal plane, the robot used (RTX, which is an UMI made SCARA robot) was anthropomorphic (see Fig. 5.7). The experiments investigated the effect of non-identical arms, variations in human arm appearance as determined by folds and color of clothing, objects in the environment, effect of changes in lighting, the necessary resolution of the 'eye', and compared classic neural and fuzzy neural models of eye-arm coordination. The robustness at variations of instructor's arm appearance is very important as the folds on cloths produce a varying contour of the arm, and also different cloths may have different color, and in the end one wants the robot to learn to follow any instructor whose arm stays within a reasonable variation in appearance compared to a 'standard' arm, and not only a particular instructor.

The full setup used in the first set of experiments is presented in Fig. 5.5. A photo of the laboratory, while using this setup is shown in Fig. 5.6. The two cameras mounted on the ceiling do not appear in the photo. One camera takes images of the robot arm, and the other takes images of the human arm (Fig. 5.7). Which particular camera is used by the robot for learning the visuo-motor coordination or for imitation depends on the particular experiment. The video monitors were installed to help the human instructor to more precisely estimate similar postures when imitating the robot arm. The robot was controlled from the PC via a serial interface.

The second type of experiments were targeted at investigating how the approach extends to the 3D performance. Two identical looking robots (RTX) were used, one learning to imitate the other. The human operators controls the master robot via a computer. This time the camera was placed at the approximate position of the human eye, gazing at an oblique angle to the master arm (only one camera was used), as illustrated in the drawing in Fig. 5.8 and the photo in Fig. 5.9. One image as seen by the camera is shown in Fig. 5.10.

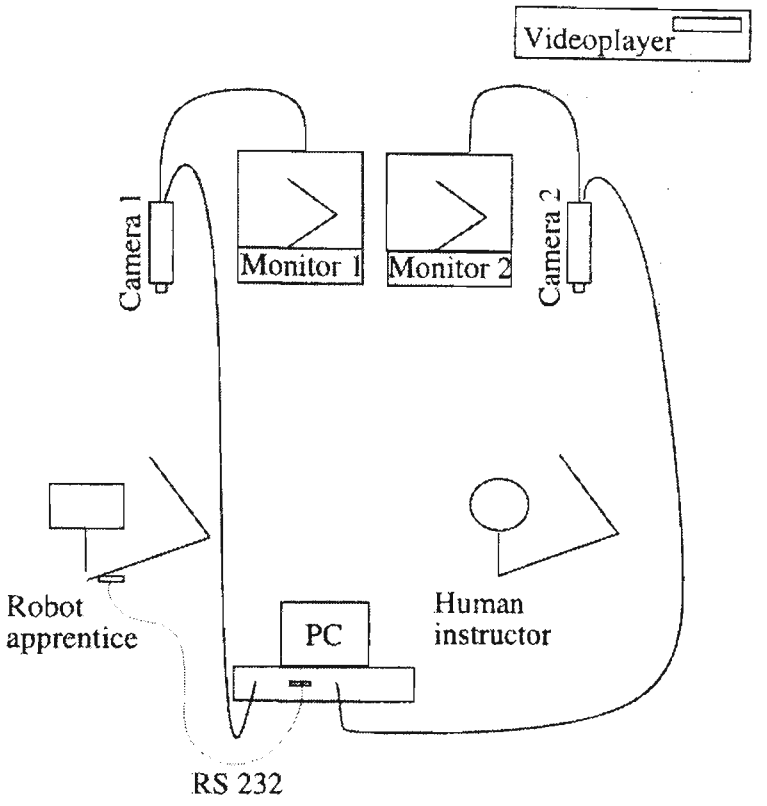


Figure 5.5: Laboratory setup for experiments of first type

5.3 Summary

This Chapter has reviewed work related to an integrated approach to robot motor learning. It also has introduced a perspective on developing robot apprentices, robots that first learn to visually coordinate their arms and then imitate human arms in their performance of a task. It was proposed and argued here that, for learning skilled movements, eye-arm coordination is more appropriate than eye-hand coordination. A technique for providing training examples for learning eye-arm coordination was also proposed, in which the human imitates the robot. The final part presented the experimental setup in which the approach was tested. The next Chapter presents experimental results of using the proposed technique for learning neural models of eye-arm coordination.

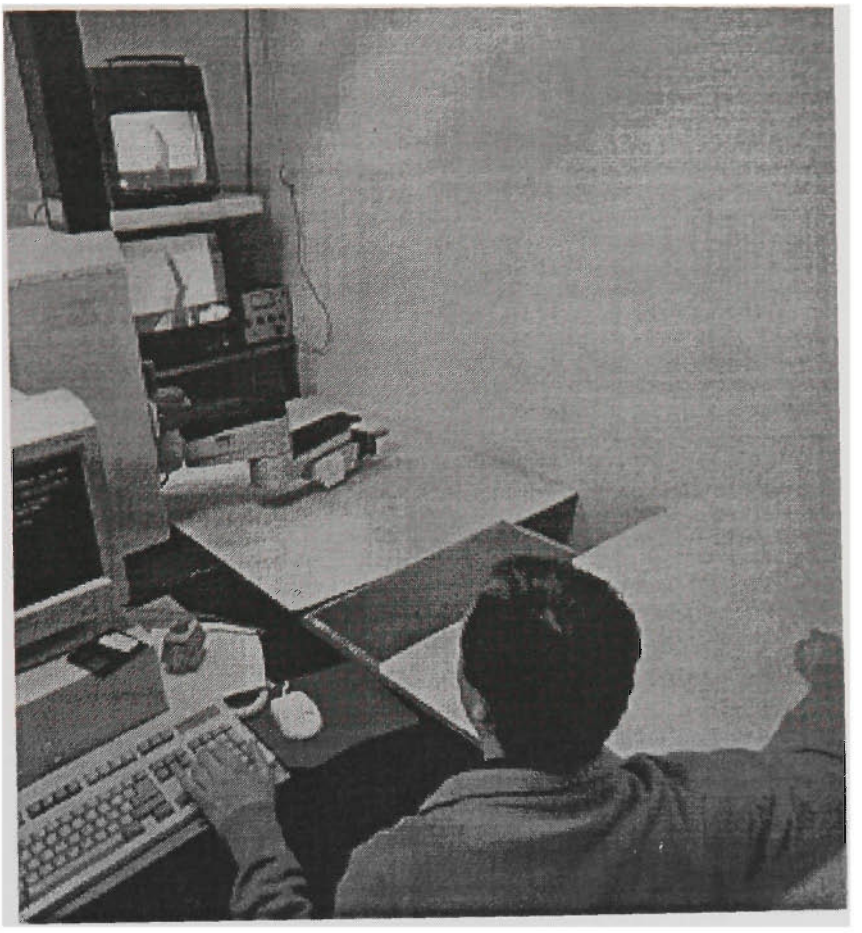


Figure 5.6: Photo from the lab

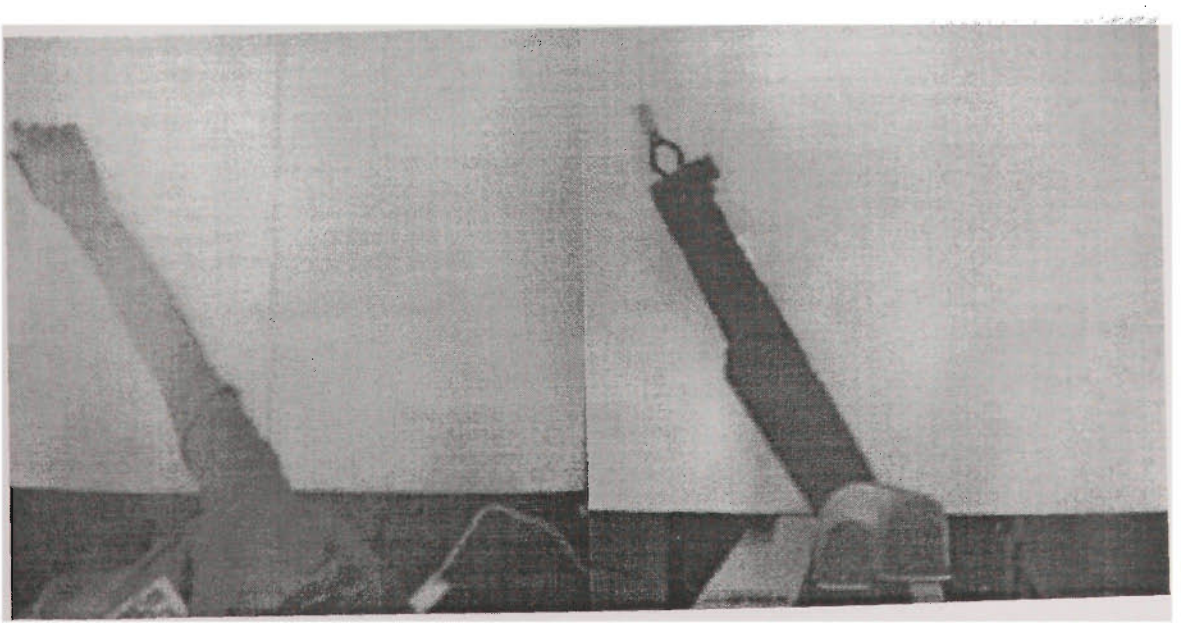


Figure 5.7: Images taken by the two cameras for the teacher and apprentice arm

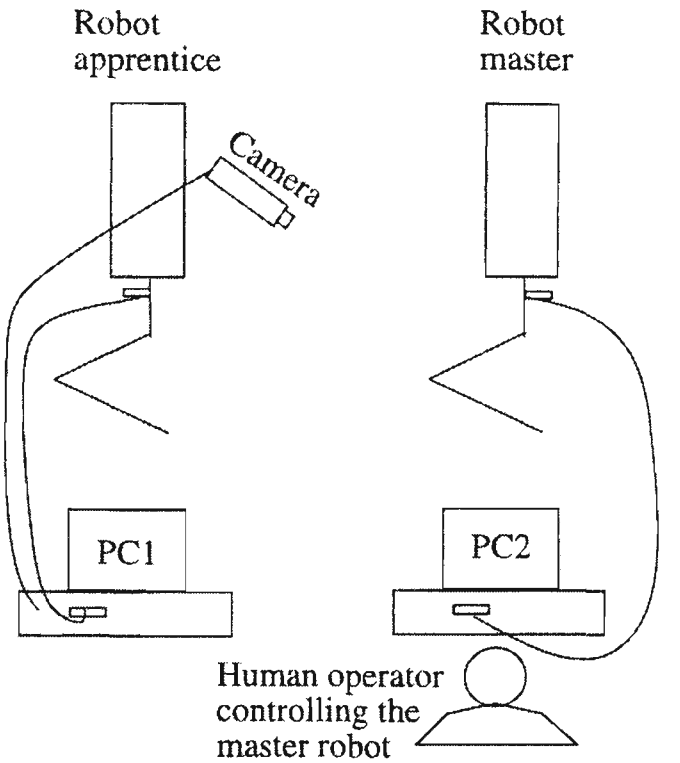


Figure 5.8: Laboratory setup for experiments of second type

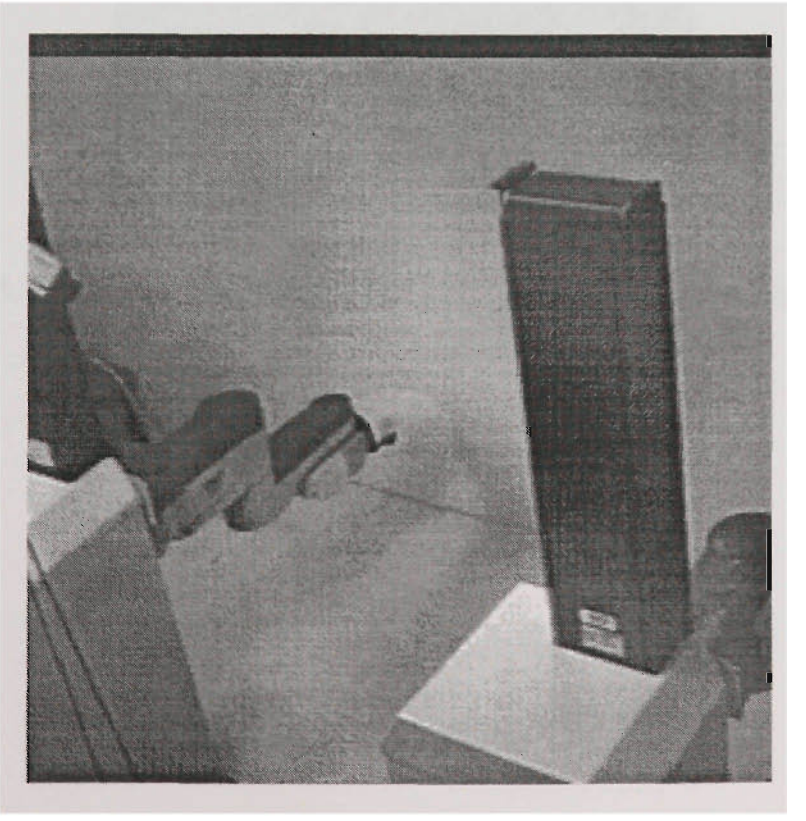


Figure 5.9: The two robots side-by-side. Camera on top left of the image

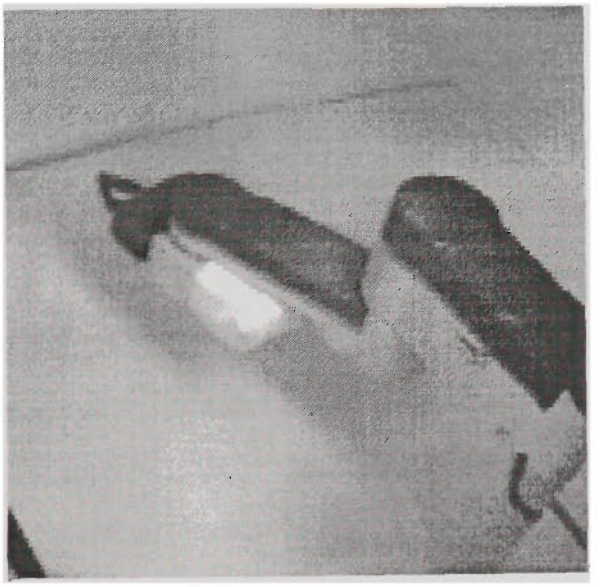


Figure 5.10: Image of master arm as seen by the 'eye' of the apprentice

Chapter 6

Learning neural models of eye-arm coordination

This chapter describes experiments with a neural based robot that learns to imitate the movements of a master arm, illustrating the feasibility of the approach proposed in Chapter 5. It is shown that only one neuron per joint is sufficient for learning the eye-arm coordination. The neurons can be classic[1] or fuzzy. Classic neurons were trained using off-line gradient descent, while fuzzy neurons were trained on-line, incrementally, using algorithms that solve fuzzy relational equations. The fuzzy neural model is transparent and has a direct interpretation, which makes it a better choice than the 'black box' classic neural model. Fuzzy neural models of eye-arm coordination learned from one arm can be used for tracking other arms of similar appearance. Also, fuzzy neural models are less influenced by additional objects that appear in the image of the workspace.

[1]The classic neuron model considered is the 'sum-product-logsig' model presented in Appendix G.

6.1 Inputs, outputs, and the choice of a neural model

6.1.1 Visuo-motor mapping

The purpose of developing a visuo-motor coordination is to allow the robot to place its arm in a position similar to that of the master arm. Thus, a mapping between images of the master arm and commands to the robot arm must be learned. During the learning of the visuo-motor model the visual inputs could be from the robot's own arm, from the arm to follow, or from another teaching arm (these variations are discussed in detail later in this chapter). During the imitation of human arm movements, the visual inputs are images of the human arm. To reflect this visuo-motor coordination, a model can be considered as in Fig. 6.1, reflecting the mapping between visual inputs and joint motor commands.

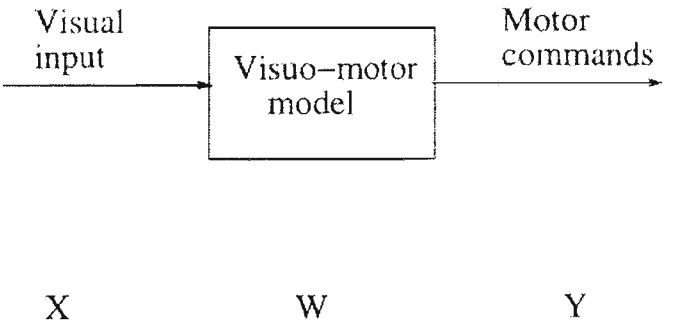


Figure 6.1: A model of visuo-motor coordination

6.1.2 Inputs: images at low resolution

The inputs to the model, denoted by X in Fig.6.1, are low resolution images originating in the images obtained from video cameras. Fig. 6.2 depicts a typical view of the arm, the image at low resolution, and the same image in a presentation which indicates the levels of grey of various regions in the image.

Initial experiments were performed using low sampling, by selecting each 16-th pixel in a 200x256 input image, to form a low resolution image. This did not prove robust enough and resulted in coarse movement. For example, if the arm contour had a close proximity to a sampling point, a one pixel shift in the high resolution image produced a change in the low resolution image. At the other extreme, for the arm contour relatively remote from a sampling point, a several pixels shift in the high resolution image had no effect on the low resolution image. To avoid this non-uniform sensitivity, the following solution was adopted. The low resolution images were obtained by averaging regions of 16x16 neighbouring pixels, and their average was considered the contribution of the region to the low resolution input. This lead to a 12x16 image, with a total of 192 inputs per image (see Fig. 6.2). A higher resolution (of 24x32, i.e. 7668 inputs per image) was also tested and its effect on system performance is discussed later in this chapter [2]. The intensity values were normalised (as the fuzzy models require inputs in the [0,1] interval), with 256 grey levels between white (0) and black (1).

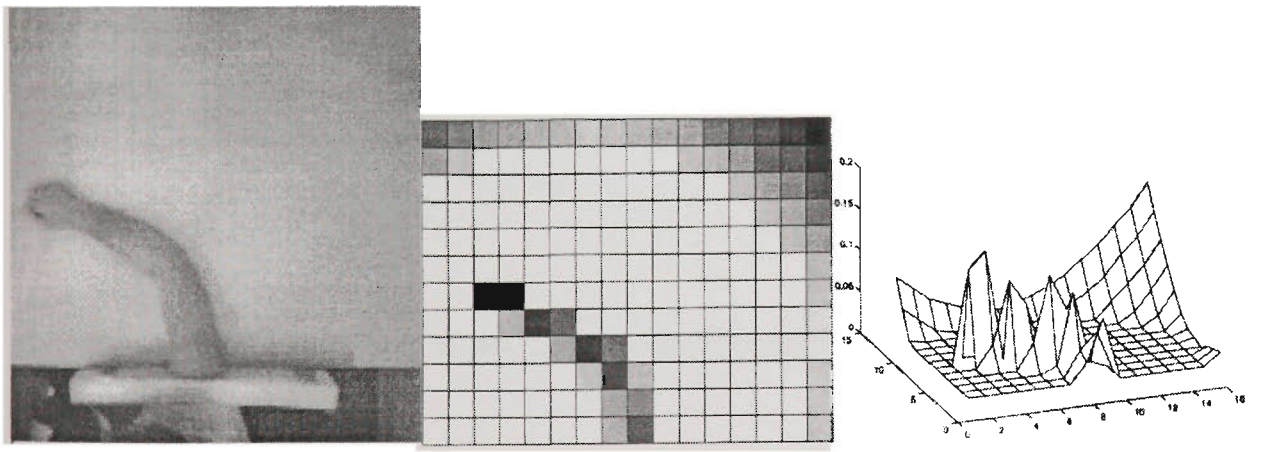


Figure 6.2: Image taken by the video camera and image at low resolution

[2]NAVLAB uses a 30x32 low resolution image derived by averaging a low fraction (around 3%) of pixels in areas of size 16x16 in the high resolution image.

6.1.3 Outputs: control commands to joint motors

The output variables, denoted by Y in Fig 6.1, were associated with shoulder and elbow joint angles (Fig. 6.3). The 'home' position for shoulder was considered on a side, forming an angle of 90 degrees with the frontal direction (for the right arm in a horizontal plane). The maximum range of the shoulder angle varied from 0 to 180, which corresponded to the movement of the arm from front to back; the interval [0,180] was mapped into the [0,1] interval. The elbow angle was considered between the upper-arm and the lower-arm, with a maximum of 180 degrees for the fully extended arm. The maximal range of variation of [0,180] was mapped to [0,1].

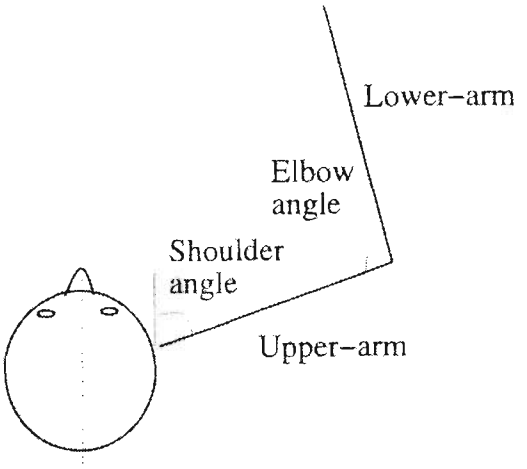


Figure 6.3: Arm skeleton showing shoulder and elbow angles

6.1.4 Mathematical models and identification from examples

The visuo-motor mapping of Fig. 6.1 can be expressed as a function Y = f(X). For neural models the expression of the function depends on the architecture of the network and the type of neurons used. For example, for a layered structure of fuzzy neurons the output Y can be obtained by a repeated S-T composition, i.e.

$$Y = X \circ W = X \circ W_1 \circ, \ldots, \circ W_n$$

where $W_1,...,W_n$ represent the weight matrices of the neural layer. The experiments performed have shown good results using the simplest possible structure with only one neuron per joint. More complicated structures which may give even better models were not considered for investigation because the limitations of the approach did not come from this direction. The limitations of the practical implementation are derived from a set of assumption detailed in Chapter 7, such as, for example, the use of a fixed direction of viewing the arm. To cope with non-simplified situations, including variable viewing positions and changing backgrounds, more complicated neural structures would be needed.

The two neurons associated with the joints generate a graded output in the [0,1] interval, and produce a command signal to shoulder and elbow motor joints (Fig. 6.4).

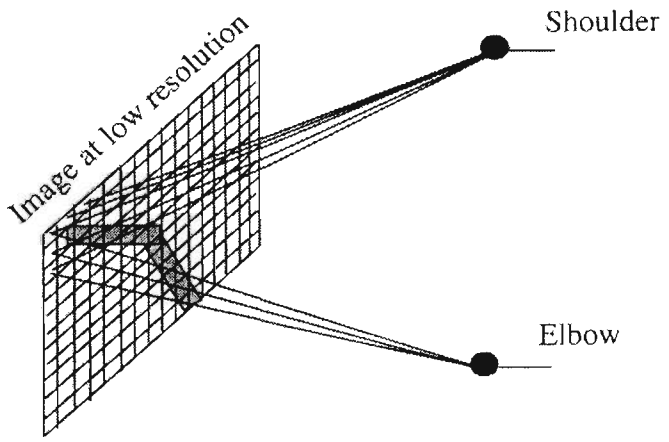


Figure 6.4: Shoulder and elbow neurons that map images to joint commands

Model identification from training examples consists of finding W, for given X and Y pairs. The procedure for generating training examples was that proposed in Chapter 5, according to which the human imitates the robot. The image-command pairs were selected to (approximately) uniformly cover the workspace. A total of 97 image-command pairs was collected, and separated in a training set (88 pairs) and a test set (9 pairs), the number of pairs in the test being taken to be about 10 percent of the training set [3]. The order in which the examples were presented did not influence learning. However,

[3]The system of 88 training pairs is a system of 88 equations with 192 unknowns.

as the human had to imitate the robot, it was easier to have minimal arm posture changes in consecutive examples, so that the human could track them easier. It is predictable that an increased number of training examples, which determine a better sampling of the workspace, would increase the accuracy of the model (up to a limit which is determined by the method of generating examples). This was confirmed by the experiments. All the tests were initially performed at the lowest image resolution image and the lowest number of examples, which was kept as a reference for comparison. The performance of models with classic and fuzzy neurons is presented in the following.

6.2 A model based on classic neurons

The classic neuron model employed in the tests was the sum-product-logsig presented in Appendix G. The learning method was batch GD with momentum and an adaptive learning rate (trainbpx in Matlab [Demuth and Beale 1994]). The plots of the convergence during training are illustrated in Fig. 6.5.

The model was evaluated against the target values from the training set, the results being illustrated in Fig. 6.6, and on the test set using images which the robot had not seen before, the results being illustrated in Fig. 6.7 [4]. The performance of the neural model is considered good. As the test examples were part of the total number of examples selected to approximately uniformly cover the workspace, and did not appear in the training set, the test regions were poorly covered by training. Doubling the number of examples lead to an increased accuracy of the model, as seen in Fig. 6.8. Increasing the number of training examples beyond some value would not continue to improve the approximation power of the model as the examples used in learning were prone to errors

[4]Simulations have shown that using a compressed output i.e. [0.25, 0.75] instead of [0,1] (thus avoiding the region in the neighbourhood of 0 and 1, see the logsig characteristic in Appendix G) the error decreases faster, the accuracy after 8000 steps having increased about 6 times for the training set and about 3.5 times for the test set.

resulted from what the human perceives to be 'similar' posture positions.

The analysis of the neural weights did not show any readily interpretable distribution. Fig. 6.9 shows the graphic representation of the weights of the elbow neuron, organized as a 12x16 array, similar to the 12x16 image inputs upon which it acts as a weighting element (each weight modulates an input cell of the low resolution image).

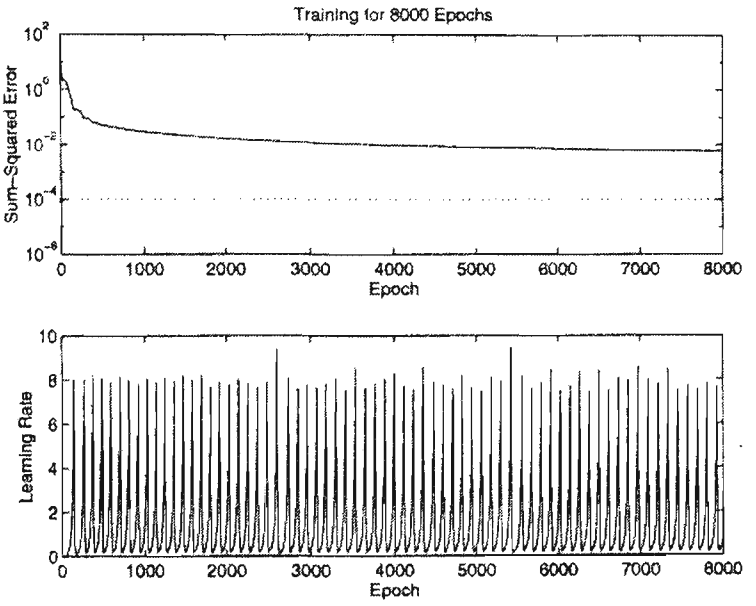


Figure 6.5: Convergence of the GD algorithm

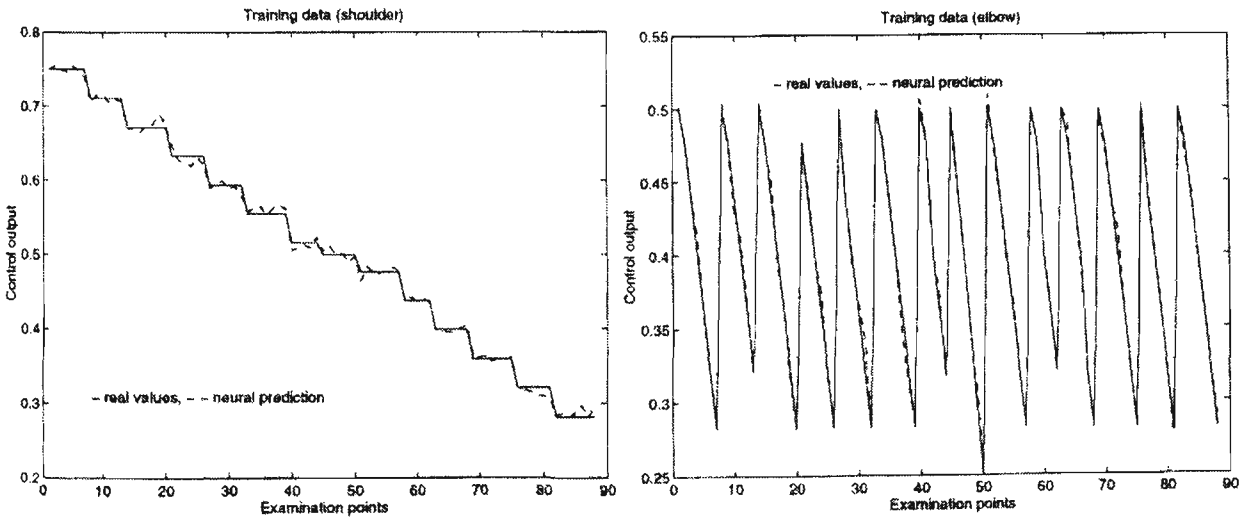


Figure 6.6: Evaluation of the classic neural model on training data

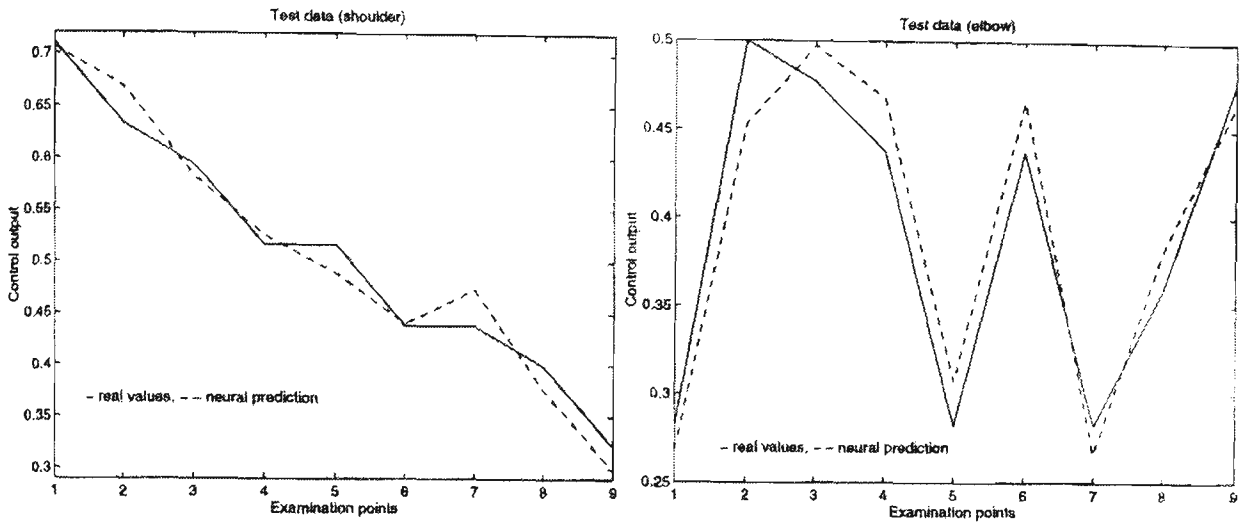


Figure 6.7: Evaluation of the classic neural model on test data

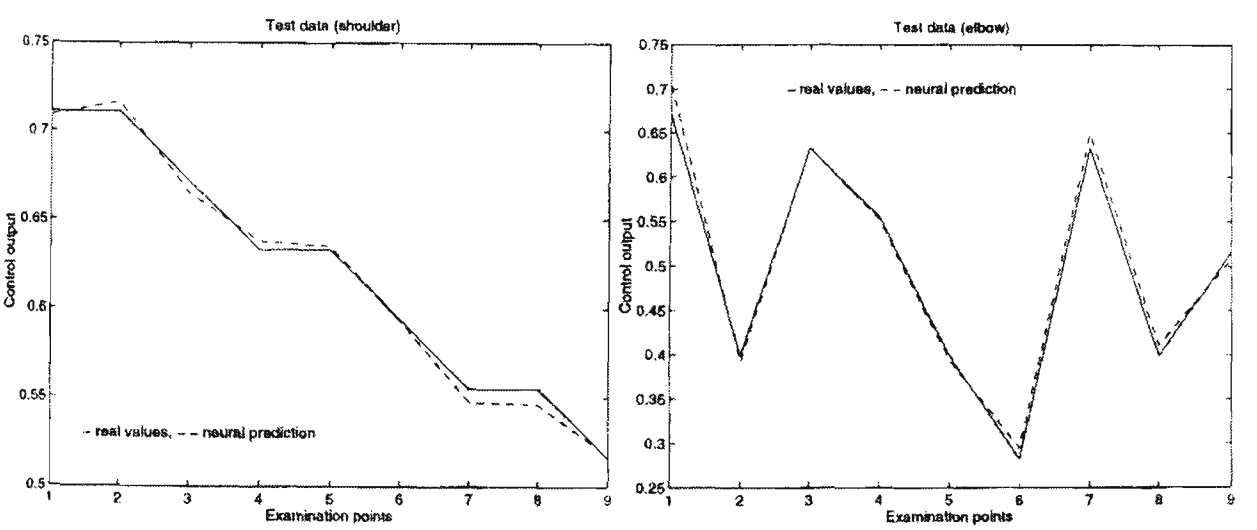


Figure 6.8: Evaluation (on test data) of the classic neural model trained with double number of examples

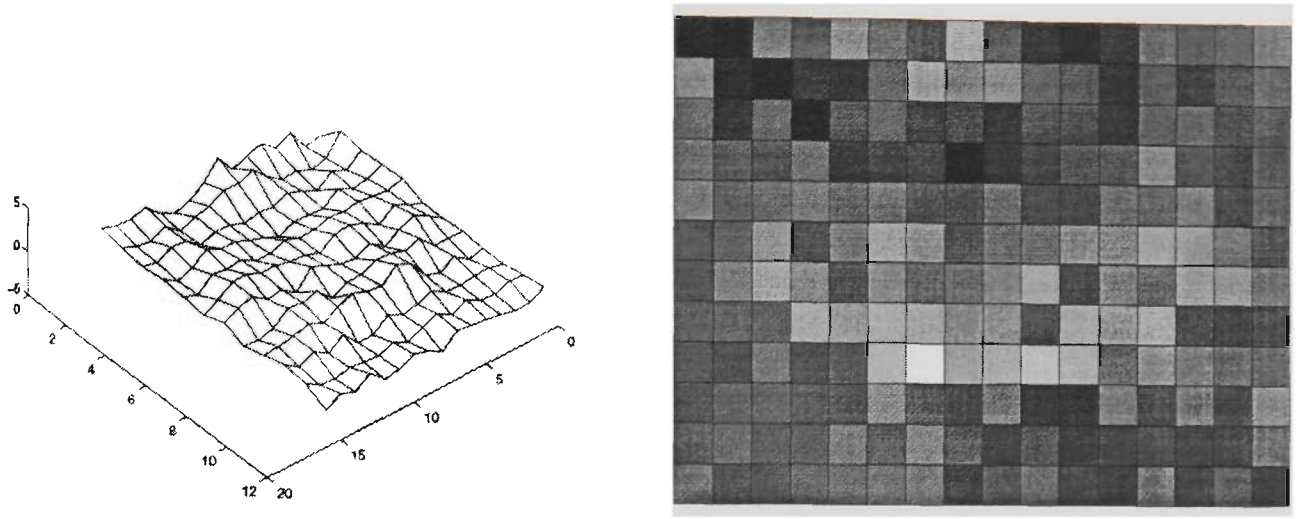


Figure 6.9: Weightspace for elbow node after training by GD

6.3 Fuzzy neural models

This section investigates fuzzy neural models of eye-arm coordination. The learning algorithms proposed here are inspired from analytic resolution methods of FRE, and allow on-line learning with a single pass through training data.

6.3.1 A necessary condition of solvability of a system of MAX-MIN FRE

The problem of training MAX-MIN motor control neurons from input-output examples is equivalent to the resolution of a MAX-MIN FRE. The conditions of solvability for MAX-MIN FRE (Eqs. 2.7, 2.8) indicate that, for the given training set (the same set used for training the classic neurons), the system of FRE does not have solutions. A first step to render the system solvable, is to scale down the output, below the maximum values of inputs (which for my uncovered arm gave a grey level of about 0.22, and for the robot arm gave a level of about 0.5). This condition guarantees a solution for each individual equation, however, does not guarantee that the system of equations has a solution.

Numerical experiments indicated the following necessary condition of solvability of a system of MAX-MIN FRE. Consider the fuzzy relations X (of size pxm), W (of size mxn) and Y (of size pxn), and indices $i \in 1,\ldots,m, j \in 1,\ldots,n$, and $p \in 1,\ldots,k$.

Theorem. A necessary condition of solvability for a system of MAX-MIN FRE given by $X \circ W = Y$ is

$$\underset{k}{MIN}(\underset{j}{MIN}(Y(k,j))) > \underset{k}{MAX}(\underset{i}{MIN}(X(k,i))). \tag{6.1}$$

This excludes the case when for all k,

$$Y(k,j) = m < \underset{k}{MAX}(\underset{i}{MIN}(X(k,i))), \tag{6.2}$$

which is trivial.

Proof: The theorem states that, if any of the outputs is smaller than the minimum value of the inputs, then the system does not admit solutions. Each column of the output can be treated separately and the results assembled at the end. Assume that the condition (6.1) is false and still a solution exists which doesn't lead to the trivial result (6.2). Consider the columns for which the output has a value m, less than the smallest input. Then by α-composition (2.11), the solution obtained has all its elements equal to m. This being lower than the inputs, all the outputs by MAX-MIN composition are equal to m, which is (6.2). The assumption that (6.1) is false lead to a contradiction, so (6.1) must be true.

From the combination of (6.1) with (2.7), results a necessary condition of solvability for a system of MAX-MIN FRE. The condition is to have, for each output column j (which also represents here the output of neuron j),

$$\underset{k}{MAX}(\underset{i}{MIN}(X(k,i))) < \underset{k}{MIN}(Y(k)) < \underset{k}{MAX}(Y(k)) < \underset{k}{MIN}(\underset{i}{MAX}(X(k,i))). \tag{6.3}$$

where i is the index of the input column and k the index of row. For the visuo-motor model, j indicates the output of the neuron, i indicates the pixel in the image, and k indicates the example. In practice one can scale the input or output levels to satisfy this condition. Here the output was compressed, and the operation was given the name 'Compressed Output As a Threshold' (COAT for short), as the outputs act as a separating threshold between the maximal values and the minimal values of the input. In this problem where the inputs are grey-scale images, the maximal values are informative dark values indicating the arm, and the minimal values indicating the level of background noise. For solvability the outputs need to be greater than the maximum noise and smaller than the minimum information level. The situation is illustrated in Fig. 6.10.

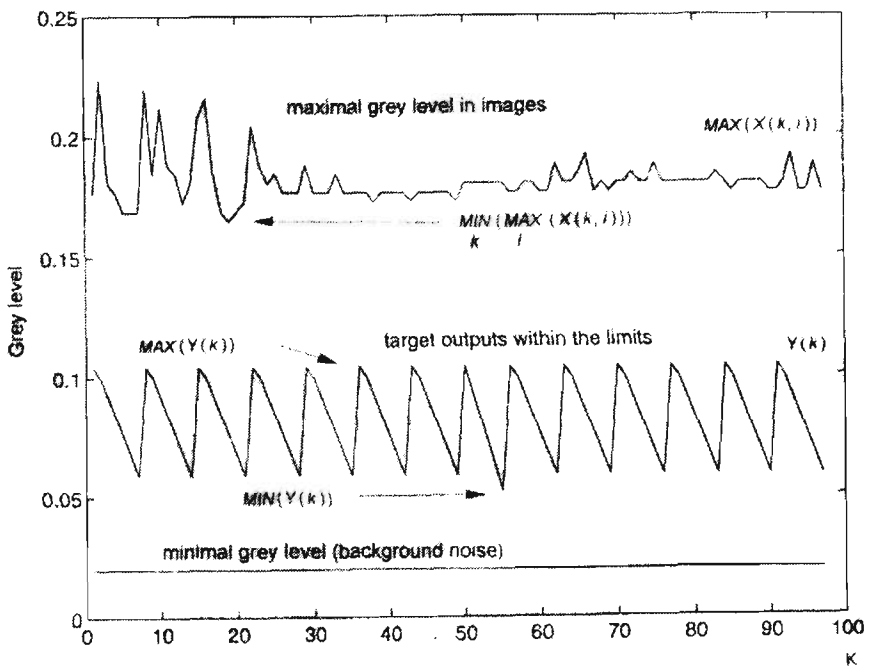


Figure 6.10: The position of the output determines the solvability of a system of MAX-MIN FRE

The (approximate) solution obtained in this case by α-composition is illustrated for shoulder neuron in Fig. 6.11.

The output of the MAX-MIN neurons having the weights calculated by α-composition is illustrated in Fig. 6.12 for the training set and in Fig. 6.13 and for the test set.

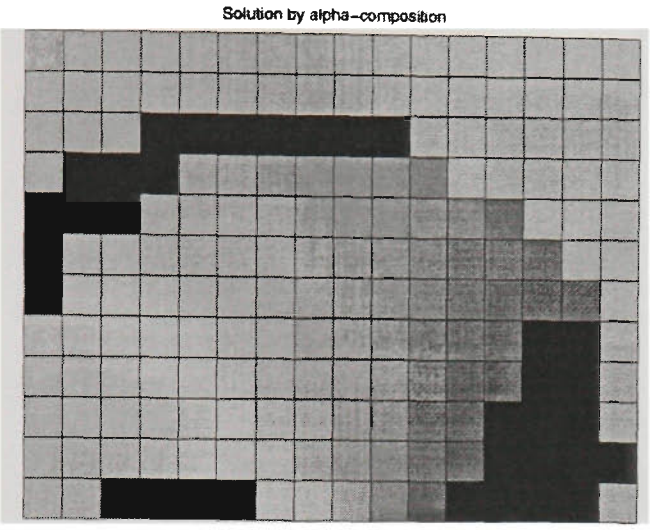


Figure 6.11: Weight matrix: solution by α-composition

The evaluations against target values indicate a satisfactory result, considering that it required only one pass through data, as compared to 8000 passes through data, as were needed for the classic model to achieve the performance shown in Fig. 6.6 and Fig. 6.7.

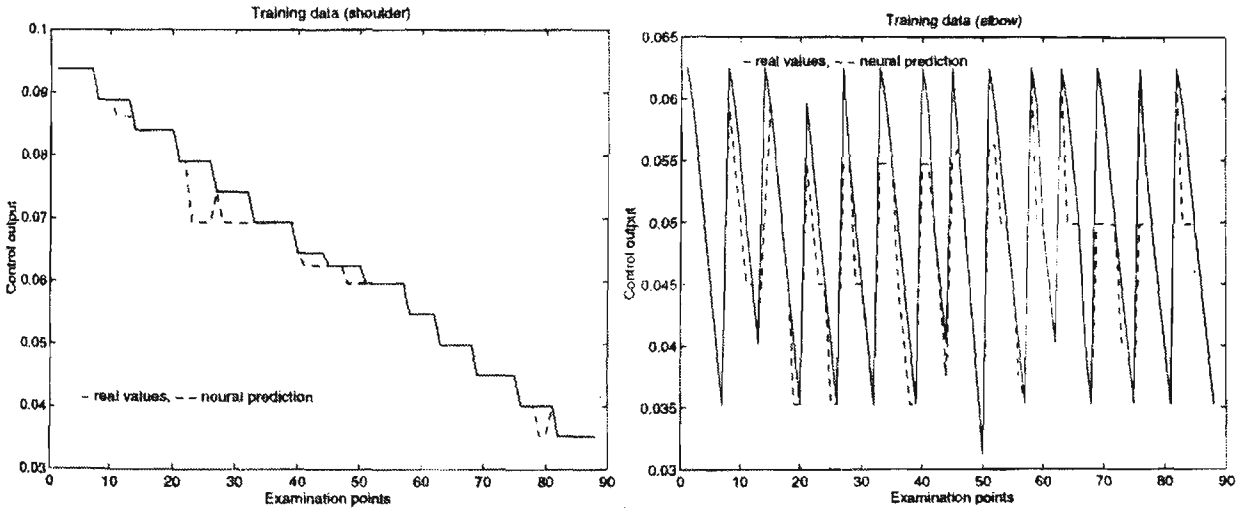


Figure 6.12: MAX-MIN model evaluation on the training data

6.3.2 The αm-composition

Very likely, the solution obtained by α-composition includes some 1's, represented as black regions in Fig. 6.11. In trying to understand the distributed representations shaped

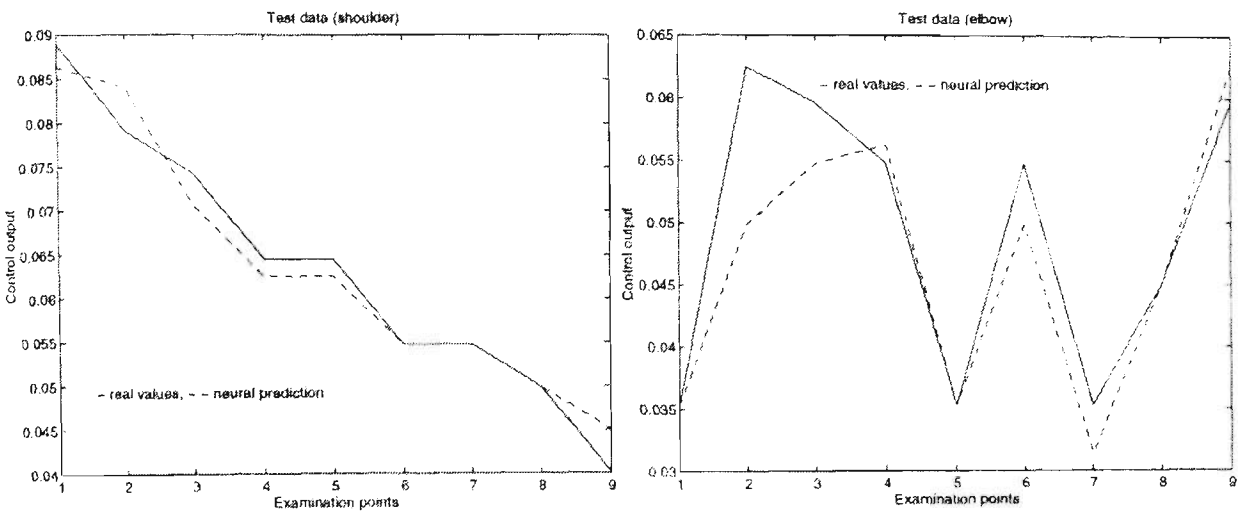


Figure 6.13: MAX-MIN model evaluation on the test data

in the neural weights, 1's are like a cover hiding interesting structures. One can remove this cover by performing a change in the α-composition. This leads to a more easily interpretable weight matrix and offers an explanation of the processing by MAX-MIN neurons.

The *minimized α-composition* (αm for short) is obtained by replacing the 1's in the α-composition given by

$$X_k(i)\alpha Y_k(i) = \begin{cases} 1 & \text{if } X_k(i) \leq Y_k(i), \\ Y_k(i) & \text{if } X_k(i) > Y_k(i), \end{cases}$$

in the way indicated by Equ. (6.4)

$$X_k(i)\alpha m Y_k(i) = \begin{cases} \underset{k}{MAX}(Y_k(i)) & \text{if } X_k(i) \leq Y_k(i), \\ Y_k(i) & \text{if } X_k(i) > Y_k(i). \end{cases} \tag{6.4}$$

For example

$$\begin{pmatrix} 0.3 & 0.7 \\ 0.5 & 0.1 \end{pmatrix} \circ \begin{pmatrix} w_{11} \\ w_{21} \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.9 \end{pmatrix} \quad W_\alpha = \begin{pmatrix} 1 \\ 0.4 \end{pmatrix} \quad W_{\alpha m} = \begin{pmatrix} 0.9 \\ 0.4 \end{pmatrix}$$

If the system of MAX-MIN FRE admits solutions, then the αm composition gives a solution.

Applying the αm - composition to the the set of image-commands pairs leads to the solution illustrated in Fig.6.11. Fig. 6.14 differs from Fig. 6.11 in the intensity of the maximal grey level, which corresponds to the maximal elements in the matrices. Both the α-composition and the αm-composition lead to the the same output of MAX-MIN neurons as illustrated in Fig. 6.6 and Fig. 6.7.

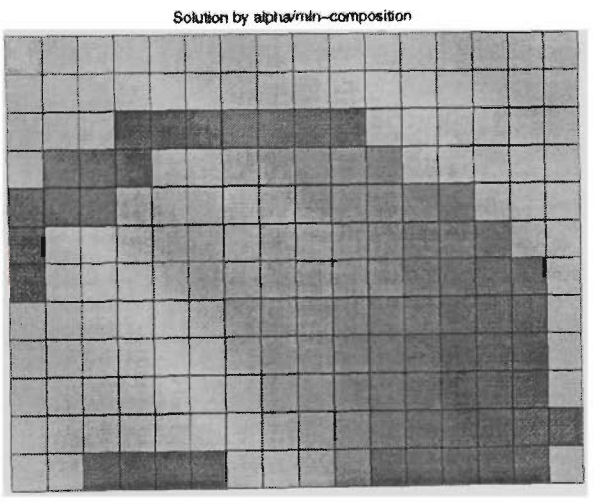


Figure 6.14: Weight matrix: solution by αm-composition

Moreover, the same output response is obtainable by replacing 1's with 0's in the solution obtained by α-composition, defining the *α0-composition*. The weight matrix of α0-composition is shown in Fig. 6.15. Unfortunately, unlike αm-composition, the α0-composition does not provide a general method of resolution for MAX-MIN FRE, i.e. it may the case that the system admits solutions, but the result obtained by α0-composition is not a solution.

The αm-composition constrains the solution to be within the same limits as the output (6.3):

$$\underset{k}{MAX}(\underset{i}{MIN}(X(k,i))) < MIN(W) < MAX(W) < \underset{k}{MIN}(\underset{i}{MAX}(X(k,i))). \qquad (6.5)$$

Placed between the information carrying level and the noise level, the weights act as

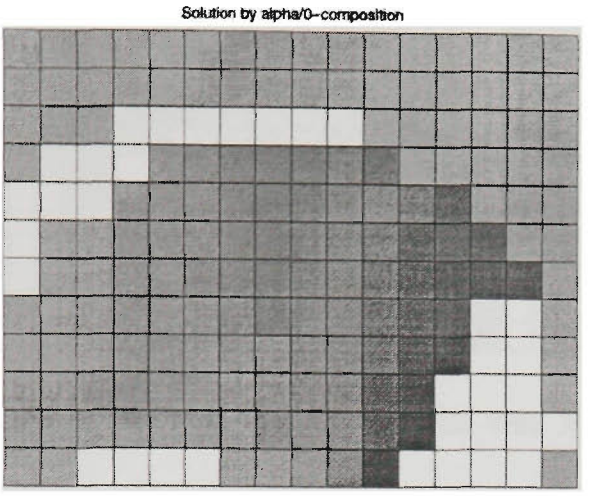


Figure 6.15: Weight matrix: solution by $\alpha 0$-composition

a modulating threshold, as suggested in Fig. 6.16. Fig. 6.16 also indicates how the output of the MAX-MIN neurons is determined. Fig. 6.17 indicates how the output is determined for the shoulder neuron. The weights are the same as in Fig. 6.14, but the grey-level was decreased to establish a common scale with the input image. Fig. 6.16 can be understood as a transversal section in the superimposed input and weight matrix in Fig. 6.17.

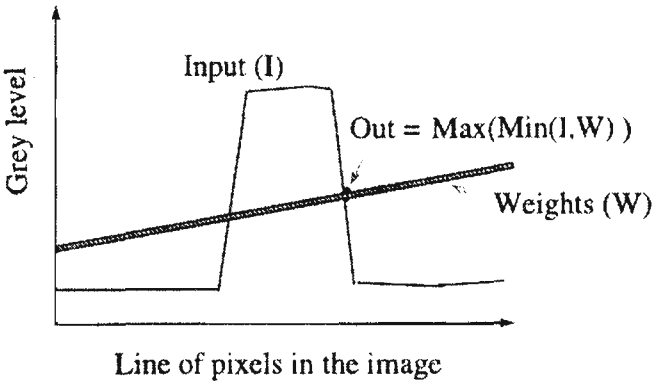


Figure 6.16: Output of MAX-MIN neurons as a point on the modulating threshold

The neural representations show the role of weights acting as a filter on input images. Some of the weights have been generated by the images of the arm, and this profiles an area within the envelope of the arm. The high level of noise in input images has lead to the weights represented in the upper part of the figures. Some of the weights in between the envelope and the noisy regions are 'don't care' weights, without influence on the

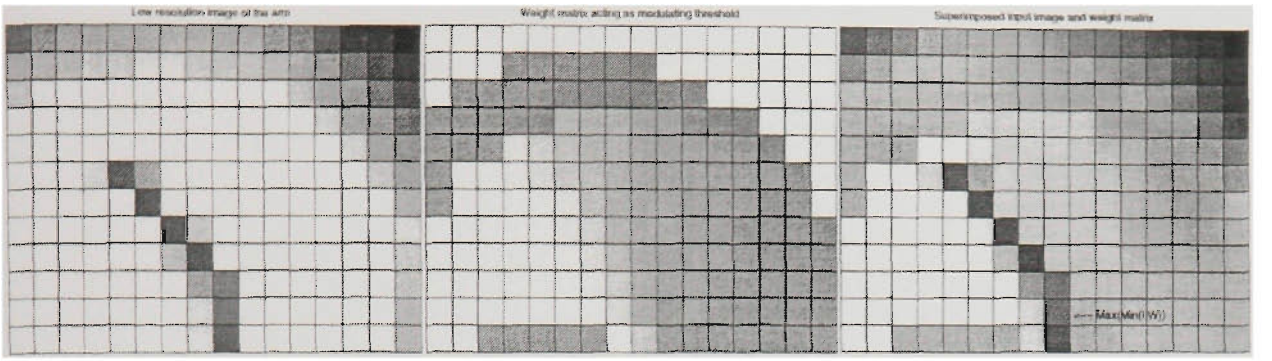


Figure 6.17: Input image of the arm, weights and the way MAX-MIN composition determines the output

output. These 'don't care' weights form the black region in Fig. 6.11, or the white region in Fig. 6.15.

6.3.3 An incremental learning algorithm

This section presents an algorithm which produces a solution very close to the ones discussed, and has an immediate interpretation. The algorithm aims to get the weights to cluster as filters shaped to favour some features in input images. Specifically the features addressed are those that determine the membership of the input pattern to an output class. The weights have associated the value of the output class they characterize.

The algorithm is called 'Maximize if Bigger, Minimize if Same (MBMS)', and includes a mechanism which reduces the noninformative weights. The algorithm uses four arrays of the same size as the image input array: the weight array, the reference array, the counter array and the filter array. The algorithm is presented in Fig. 6.18 and the corespondent MATLAB file is given in Appendix. Terms *darker* and *same* should be given an approximate meaning, possibly as fuzzy relations. In this test *darker* was considered as a level higher by 0.04 than the one compared with. Weights that change because they belong to different classes end up with lowest value associated with a class. The algorithm builds the weights incrementally. Instances during learning are illustrated

Initialize filter matrix to 1's and the other matrices to 0's

For all examples and for all input pixels

If the input pixel is darker than the reference then it becomes reference, and its weight is given the corresponding output value. Increment a counter for pixel change.

Else if the input is the same as the reference then the weight is given the minimum value between the previous weight and the current output value.

Else increment the counter for pixel change.

For all elements of the counter matrix

If counter is less than 2 reset to 0 the associated weight and filter element (there was no informative change in the input images).

If counter equals the number of examples reset the associated weight and filter element to 0.

Figure 6.18: The MBMS algorithm

in Fig. 6.19.

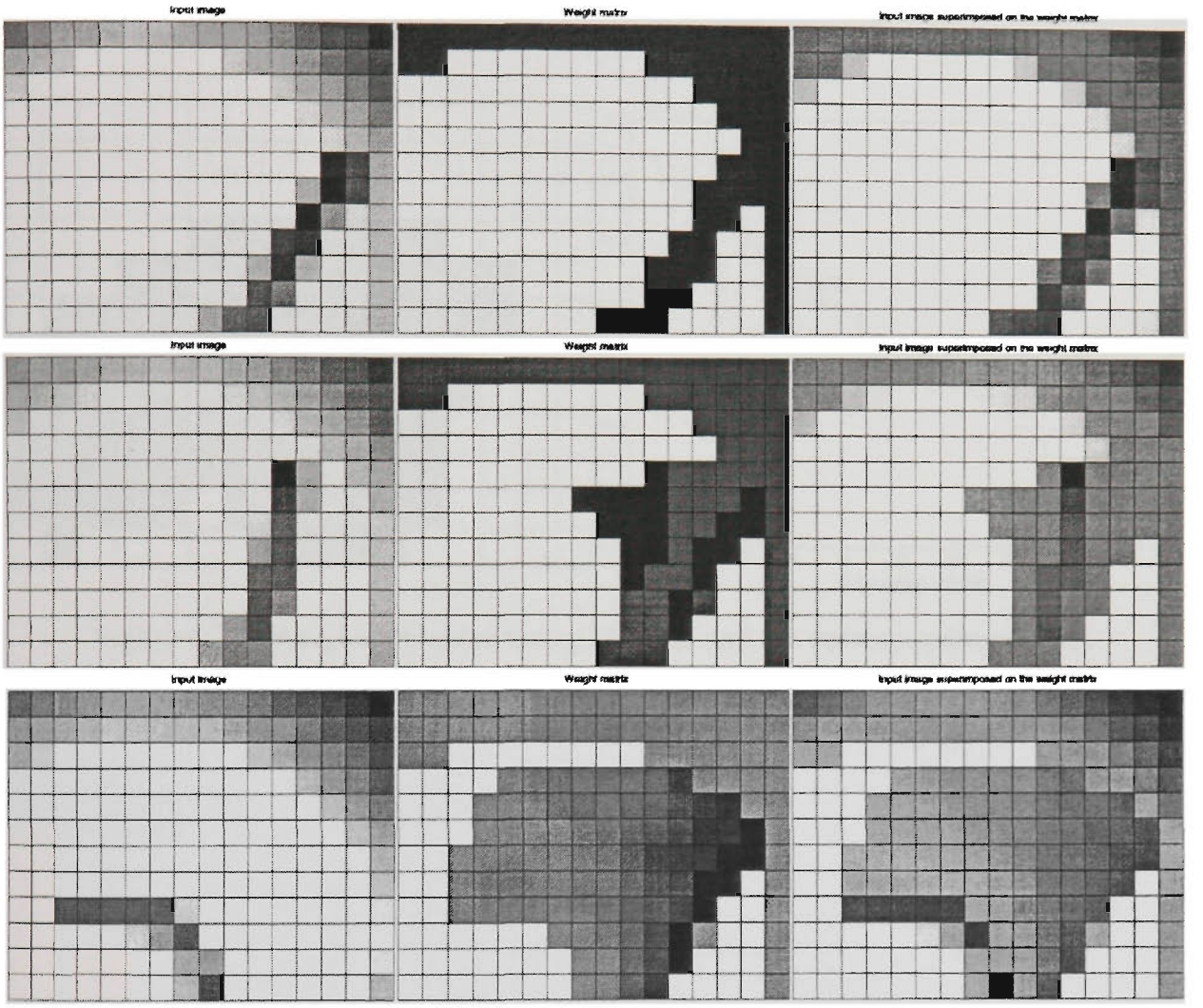


Figure 6.19: Instants during learning by MBMS. Input image of the arm, weights and arm on weights.

The solutions obtained by MBMS and the output it produces are presented in Fig. 6.20. The outputs have a bigger approximation error than that obtained by α-composition. However, of special interest is the filter obtained from the counter.

It is interesting to observe that the solutions obtained by α-composition filtered by the filter resulted from applying MBMS (Fig. 6.21) lead to the same output as obtained by α-composition alone, which was illustrated in Fig. 6.6 and Fig. 6.7. The filter reflects a maximal envelope of the arm and the filtered solution makes the system insensitive to modifications of the image outside the arm's envelope.

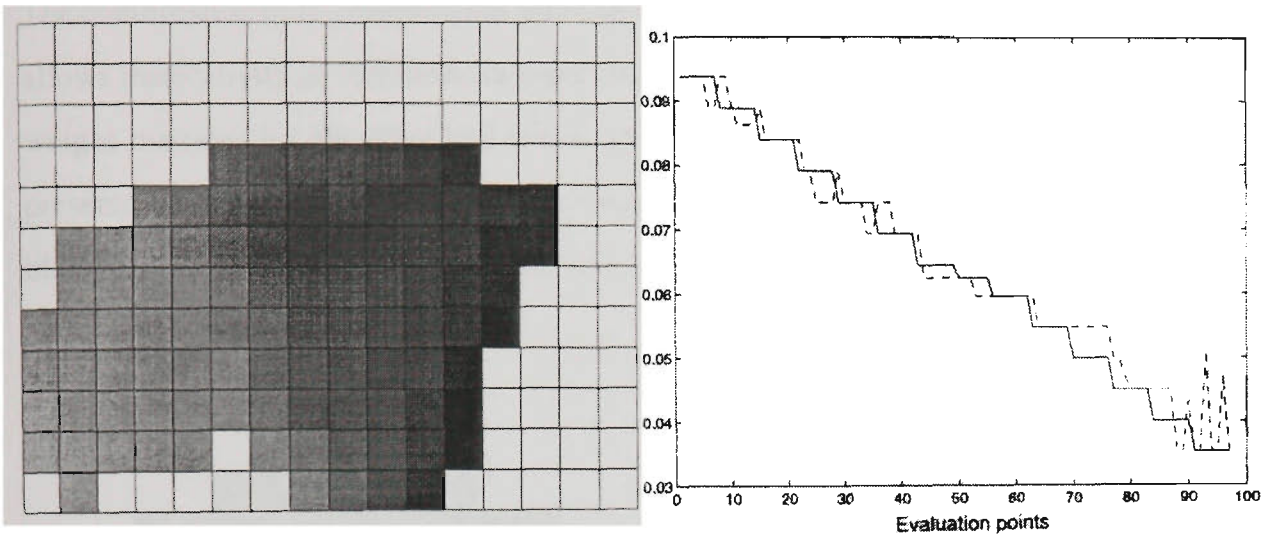


Figure 6.20: Weights obtained using MBMS, and the corresponding output

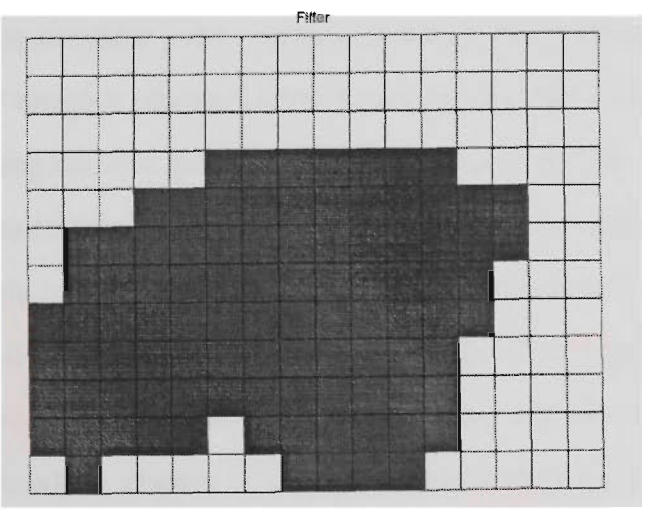


Figure 6.21: Filter from MBMS: 1's are black, 0's are white

6.3.4 Interpreting the structure in the weights.

The combination of α-composition solution with the filter resulted from applying MBMS allows insensitivity to variations in input images which are outside arm's envelope. The weight matrices for shoulder and elbow are illustrated in Figs. 6.22, 6.23. Fig. 6.24 presents the image of the arm superimposed on the final weights (the arm is presented black for better contrast) and the final weights.

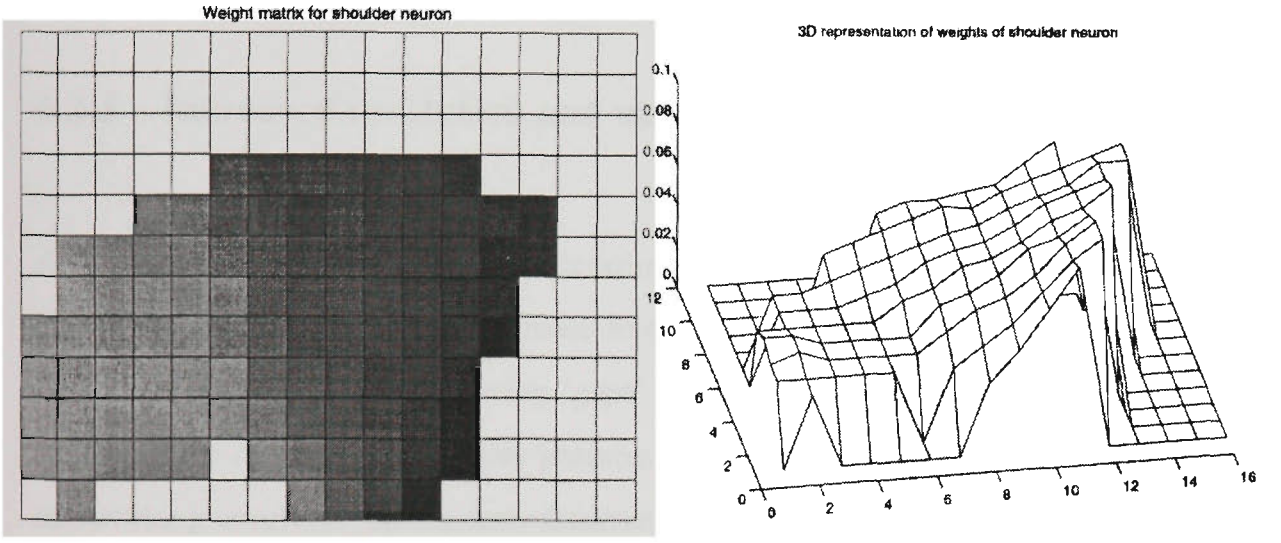


Figure 6.22: Weights of shoulder neuron

Higher values of the shoulder joint commands which make the arm move to the right are reflected in higher weights (darker in the figure) on the right. For the elbow, higher commands for extending the arm reflect in darker values along the radial direction. For the shoulder, each class of same grey level is disposed along a radial direction, the levels decreasing as the angle at shoulder joint varies from right to left. For the elbow, the classes are circular with higher values of the weights placed at greater distance from the shoulder joint and also at the rightmost extremity. This can be seen in the simplified drawings of Fig. 6.25, where classes (C_i) in input space have associated the numerical outputs indicated, and an ordering exists of input classes, along output values.

One can consider that the weights modulate input classes. Thus the system can be described by rules of the form 'If input is in X then the output is Y', where X is a

2D fuzzy set and Y is a singleton. This can also be seen as a form of Takagi-Sugeno reasoning.

An object in the image has no influence on the output, if the arm is darker than it, or if the object is outside the arm envelope. Note also that MBMS eliminates any feature which is common in all examples. Classic NN algorithms usually learn such features first.

6.3.5 Increased resolution and number of training examples.

An increased image resolution, and accordingly an increased size of the weight matrix, should be accompanied by an increase in the number of examples used for training. Otherwise, there is a risk of having 'gaps' in the weight matrix, which appear if the superposition of surfaces of all arm postures used in training does not completely cover the working envelope. In the tests illustrated here, the resolution was increased from 12x16 to 24x32, and the number of examples was increased from 88 to 140. The solutions obtained using the increased resolution and increased number of examples is shown in Fig. 6.26. The results illustrate the increased approximation power of the high resolution model.

6.3.6 Limitations of MAX-MIN neural models.

1. The MAX-MIN neural model presented here does not allow for interpolation. The outputs can only take values from the set of output values used for training, which are moulded in the weights. The number of outputs is also limited by the resolution utilized in the input images, as this dictates the number of elements in the weight matrix. For example, the errors for the elbow output are larger at high values of elbow output. This

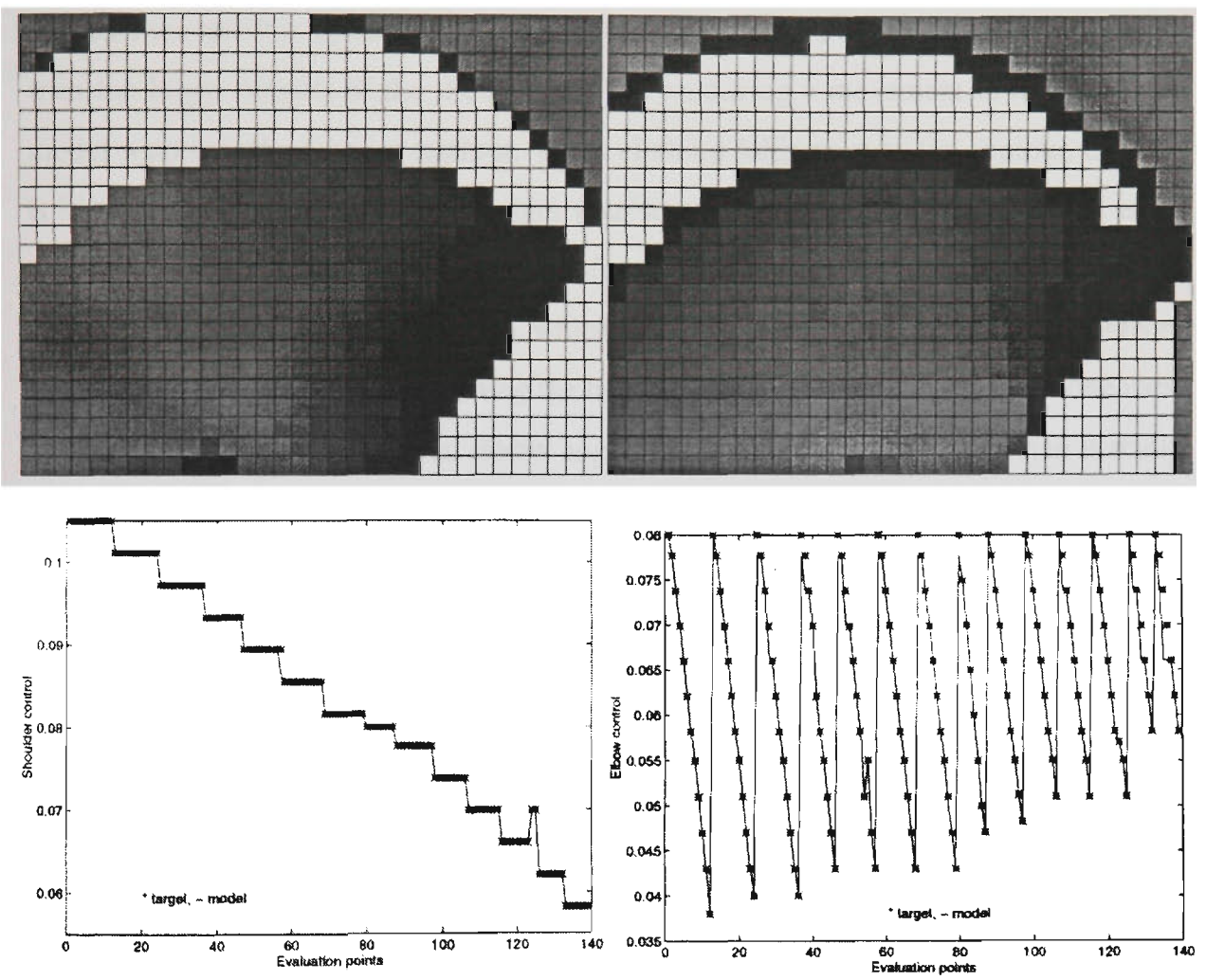


Figure 6.26: Model evaluation for high resolution input

is because at high values of the elbow angle a variation of this angle has little effect on changing an output class, as illustrated in Fig. 6.25. This effect is very strong at low resolution, as illustrated in Fig. 6.12, but can also be noticed at high resolutions as illustrated in Fig. 6.26.

2. The model is not working based on images of the surface of the arm. For the shoulder the result is determined by the contour of the arm. For the chosen arrangement with the larger outputs of the shoulder neuron producing a movement to the right, the model works on the exterior contour. When clothing folds exist on the contour, the arm is evaluated as being more to the right than it really is. The magnitude of the error depends on the size of the 'bump' that deforms the contour. For the elbow, the model works on the tip extremity of the arm, which 'covers' circles closer or further away from the circle center (Fig. 6.25). Opening the hand, or using a longer arm than the one used in training, affects the evaluated elbow angle, the angle appearing bigger than it really is.

6.3.7 MAX-T neural model

The good approximation power of classic neurons is favoured by the use of interpolation combined with the use of grey levels for the arm. A grey pixel in the image of low resolution is obtained by an average of an area in the image of high resolution, and an indicator on how much dark (from the arm, which is darker than the background) is present in that area. Using black and white pixels only, this information would have been eliminated. MAX-MIN models can not interpolate, however MAX-T models can. The maximal solution for a MAX-T FRE is calculated in general using (2.21). When T is the fundamental t-norm (3.6) with $0 < s < \infty$, $s \neq 1$, the solution is given by the

6.4 Model verification by robot control

Arm postures. The neural models determined by training were used by the robot in tracking movements of the master arm. The qualitative evaluation consisted of subjective assessments of how close the posture of the robot's arm was resembling the posture of the human's arm. The postures looked very similar as can be seen in Fig. 6.28, which shows a series of images taken during the tests.

Hand trajectories. If perfect models would be available, and the wrist fixed, identical postures of identically looking arms would guarantee identical trajectories of the end-effectors. This is usually not the case. Arm movements may lead to hand trajectories insufficiently precise for many tasks concerned with end-effector trajectory. For effective usage, eye-arm coordination should be combined with eye-hand coordination.

6.5 Learning from arms with a different appearance

Learning to 'recognize' only the arm used for training can be a strong limitation. However, the tested neural models have shown a reasonably good robustness to variations in the appearance of the human arm. Several tests were conducted, which are briefly discussed in the following. The arms used in this set of experiments are shown in Fig. 6.29.

6.5.1 Learning from an arm and extending the model to other arms

In this test, the robot learned the eye-arm coordination by watching the arm 'skin' shown in Fig. 6.29. It was then tested on the all the arms shown in Fig. 6.29. The test results

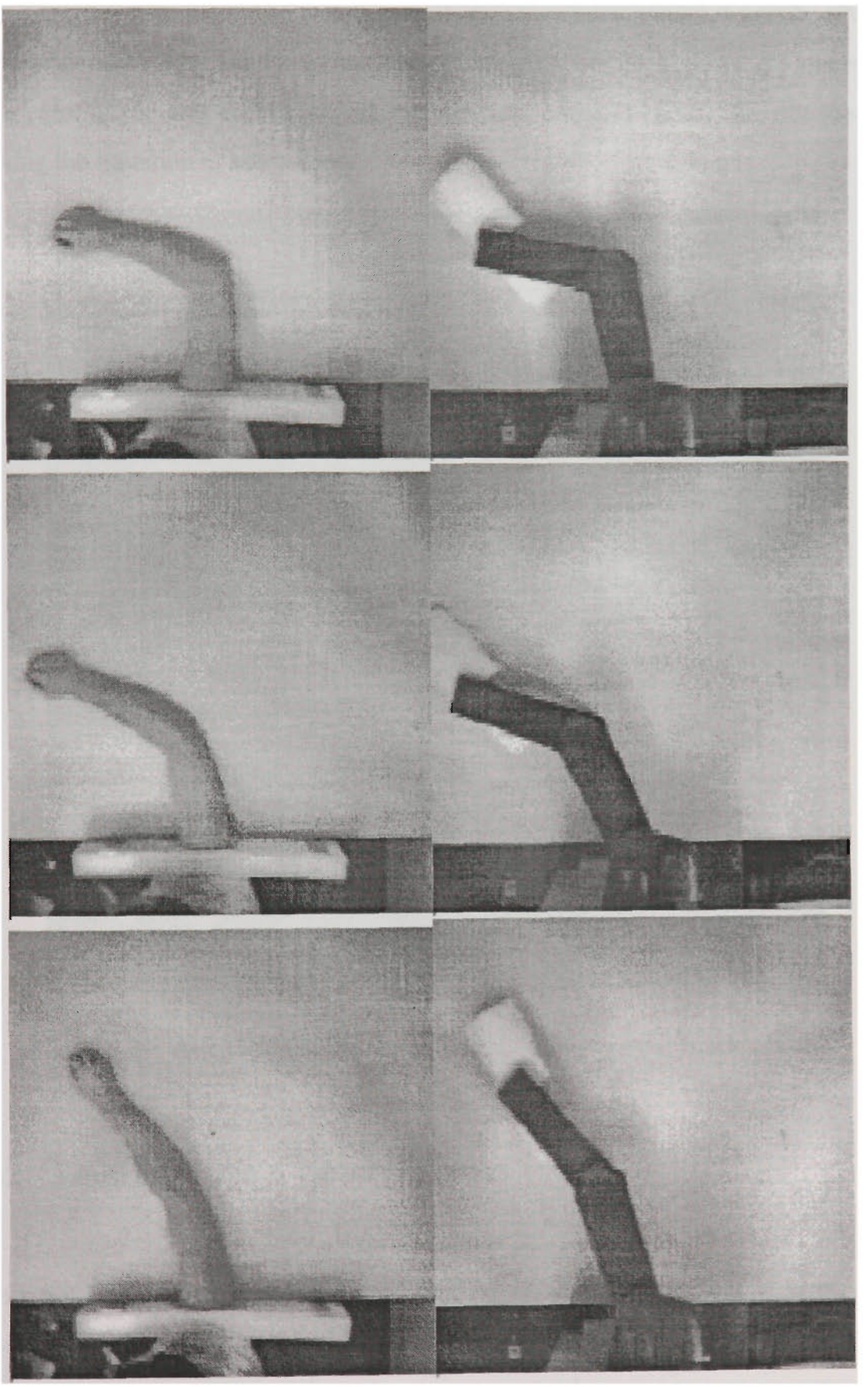


Figure 6.28: Robot moving after the human arm

are presented in Fig. 6.30. The results indicate that the classic model is greatly affected by the color of the arm. On the contrary, the fuzzy (MAX-MIN) neurons are insensitive to the color of the arm and the model learned from one arm can be directly used for allowing the imitation of other arms.

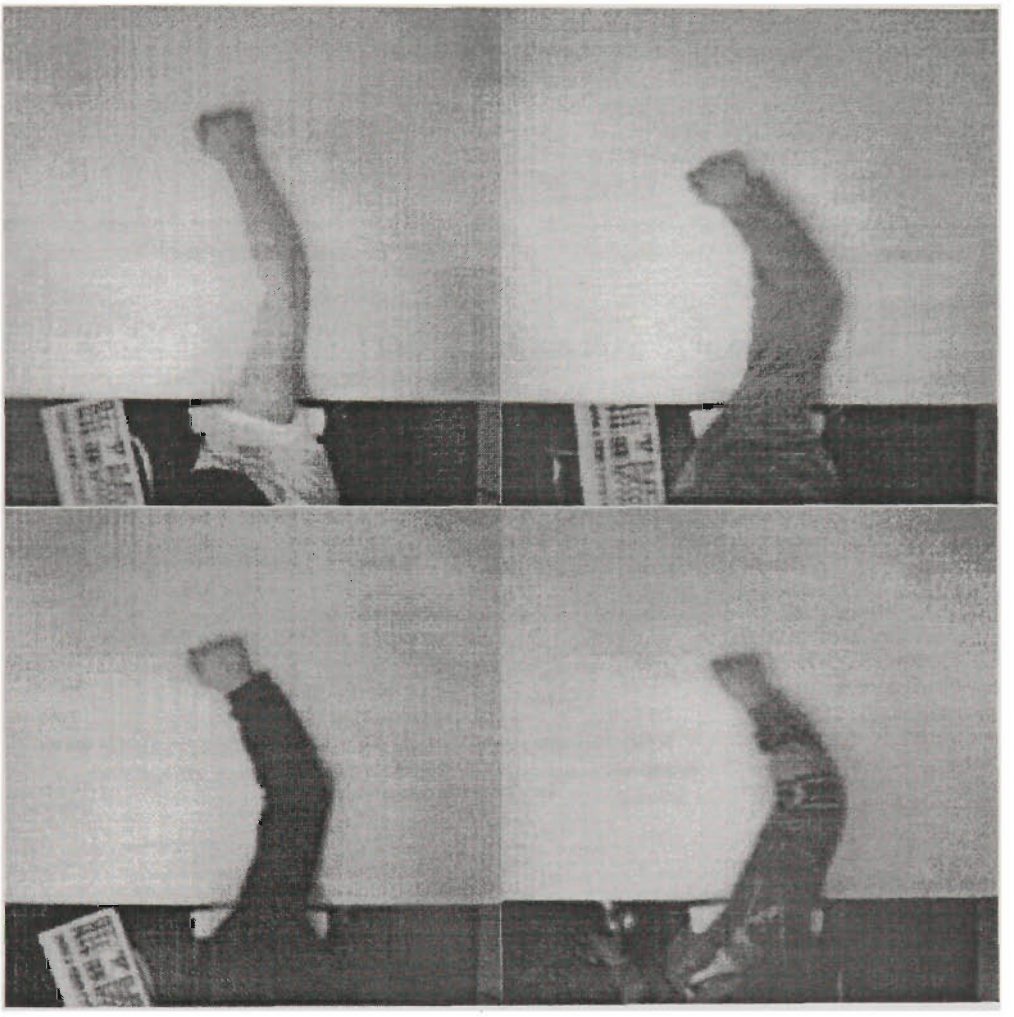


Figure 6.29: Arms used in the experiments: 'skin', 'grey', 'black', 'arm2'

6.5.2 Learning from a variety of arms

In this test, the data used in supervised training came from a variety of arms, more precisely from the arms 'grey', 'black', and 'arm2' shown in Fig. 6.29. The model obtained by learning was tested on the arms used for learning and on an arm which hasn't been seen before (the 'skin' arm) the results being illustrated in Fig. 6.31.

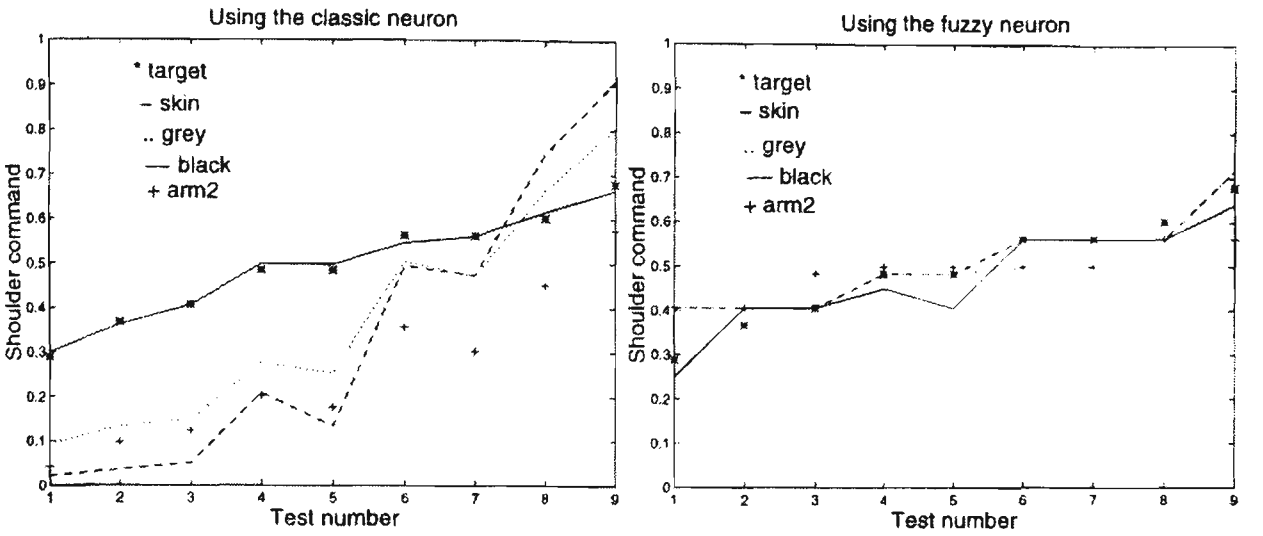


Figure 6.30: Model evaluation for the case of learning from one arm and testing on arms of different appearance

Training on a variety of arms improves the performance of the classic neural model, while it didn't affect the fuzzy neural model.

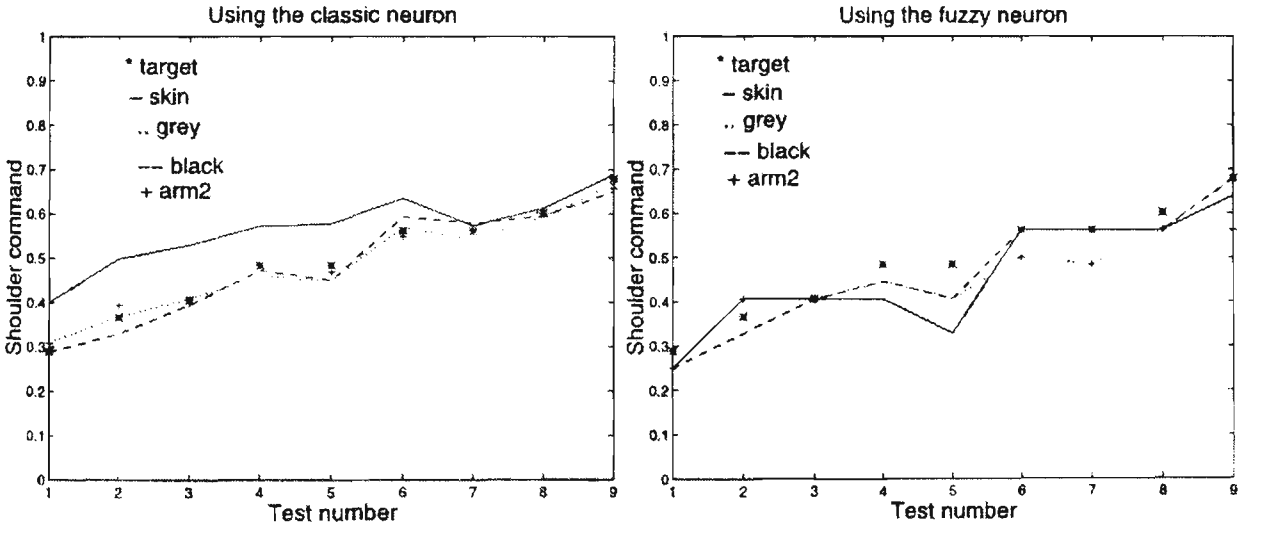


Figure 6.31: Model evaluation for the case of learning from a variety of arms

6.5.3 Learning from its own arm and extending the model to other arms

An important property of the fuzzy neural model is that allows the transfer of the gained knowledge over another user. The robot can learn on its own arm, and than can track a different arm, which can vary in color or contours. This is true for arms that appear to the robot eye as being of similar length, while the arms of different length need adjustment by scaling by a constant. The robot can learn also from images of low resolution and then use the determined weight space as a good first approximation for an increased resolution solution.

The weight matrices obtained in learning a model of the robot's own arm and a model of human's arm are illustrated in Figs 6.32, 6.33. They show the same representation, however, their similarity is limited by two factors. Firstly, there is a difference in the appearance of the two arms. Secondly, an estimation error is always present when the human imitates the robot. These sources of error produce for example the dark points in the center-left of human elbow in Fig. 6.33. Such errors should be reduced when a more precise positioning occurs, and also at a higher resolution of covering the workspace with training examples. A modification of those dark values to match the neighbors, has reduced the error obtained for these regions of the space.

6.6 Comparing classic and fuzzy neural models

In this study on learning eye-arm coordination, fuzzy neural models using t-norm based activation functions present some advantages over models using neurons based on sum-product-sigmoid operations. The advantages are:

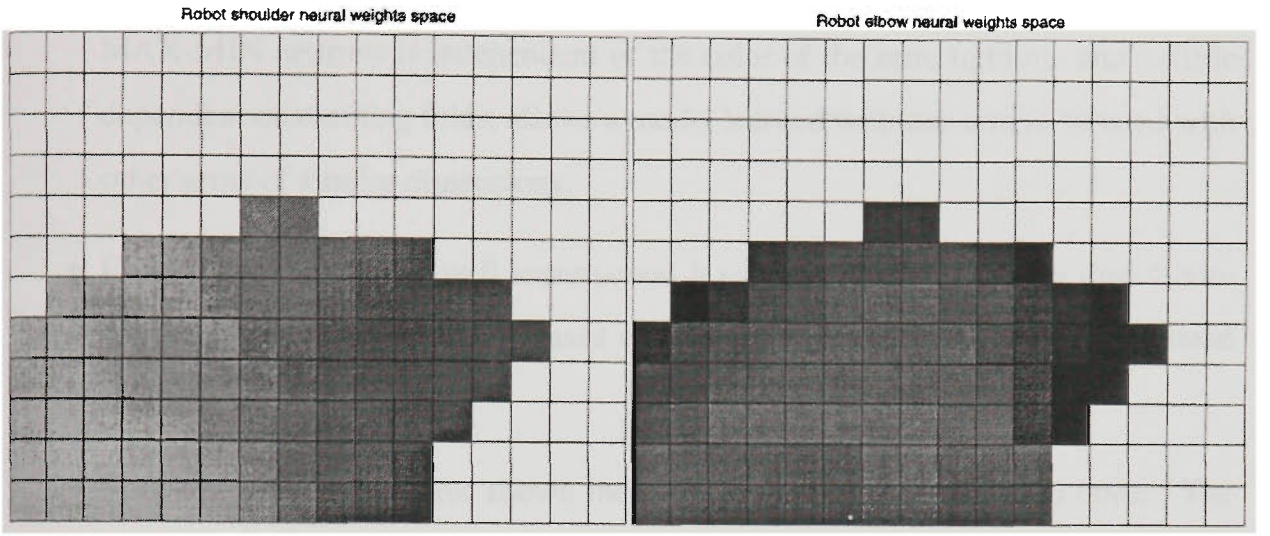


Figure 6.32: Robot shoulder and elbow neural weights

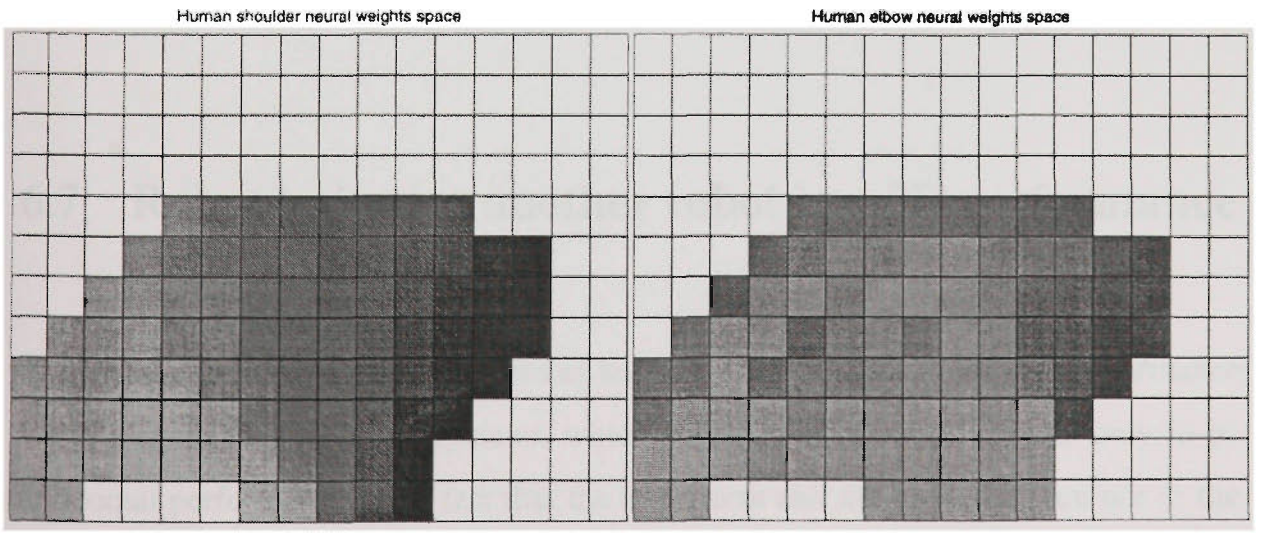


Figure 6.33: Human shoulder and elbow neural weights

- The weight space is directly interpretable by humans. The weights organized as filters took the shape of a right-left downhill slope for the shoulder neuron, and of a semicircular stadium shape for the elbow neurons. This organization, which for MAX-MIN neurons is independent of the color of the arm, lighting, and is little dependent on clothing folds, allows a model learned with one arm to be used with other arms of similar dimensions.

- Fuzzy neurons support well incremental learning. This comes as a direct consequence that learning can be based on analytical methods of solving associated FRE.

- Fuzzy neural models have shown increased robustness to structured noise. The filters enable the robot to be highly insensitive to other objects in the image.

It should be noted however, that in general the classical model has shown an increased approximation power. This is likely due to the fact that fuzzy neural models are limited to solutions in the [0,1] interval, while the weights of classic neurons can take on any real value; moreover, the fuzzy models presented in this chapter used only excitatory inputs.

6.7 Robot imitating another robot in a 3D performance

The main reason for limiting the tests of learning from humans to the 2D performance was the availability of a manipulator, which appeared as anthropomorphic only in its horizontal performance. The fact that the upper-arm and the lower-arm are not in the same horizontal plane, as it can be seen in Fig. 6.35, required a top view of the arm, if similarity with the human's arm was sought. The vertical movement of the SCARA type arm used consists of a vertical translation of the shoulder along the vertical axis, unlike the up-down movement of the human arm around a fixed point at the shoulder joint. For

the demonstration of learning and representation properties of fuzzy neural networks on a real problem, the tests performed in 2D were sufficient. However, for developing real world robot apprentices that learn by imitation, demonstrating the 3D case is essential.

The learning approach proposed in Chapter 5 is general, and equally well applies to 3D, as long as different arm commands determine different visual images of the arm. For extending the imitation of a human arm to 3D, an anthropomorphic arm having similar appearance and similar degrees of freedom must be used (and this was not available). However, the general idea is to imitate a similar arm, and the 3D tests were performed using 2 identical robots, one acting as a master and one acting as an apprentice, as seen in Fig. 6.34. The 3D tests performed have shown a similar level of performance as for the 2D case.

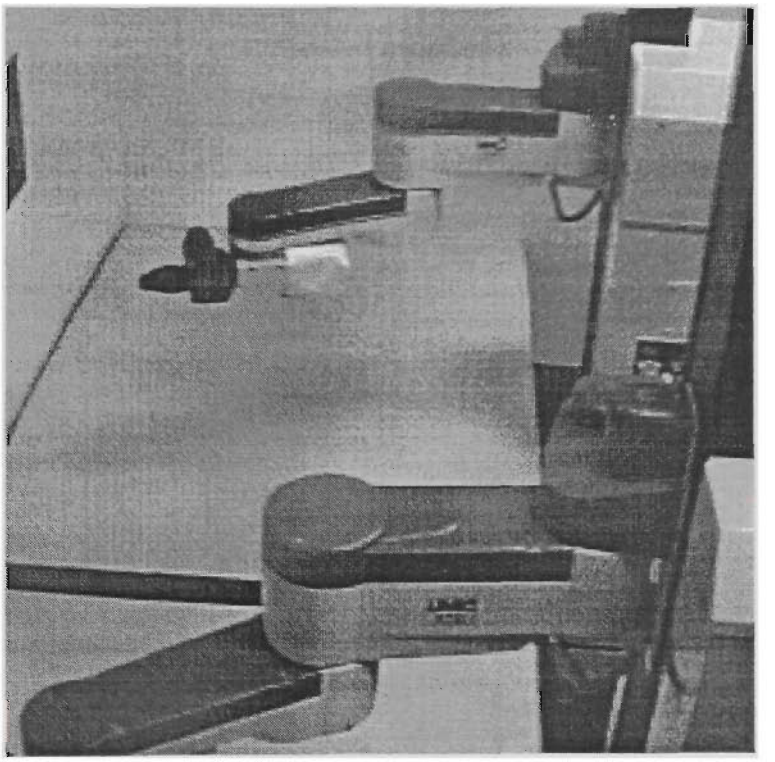


Figure 6.34: Apprentice (closer to viewer) follows master

Some images during training, as seen by the single camera, are illustrated in Fig. 6.35. Each of these images associates with a pair of motor commands for the shoulder, elbow, and z-axis motor neuron. The tests were done with the low resolution vision of

12x16 and 175 training examples, uniformly covering the range of possible displacement. Training lead to a good model which, when presented with input arm images, accurately predicted the correct arm control. A scene during imitation is shown in Fig. 6.34. Fig. 6.36 shows the response given by the model to a test set of images which the robot had not seen before. These results are better than most of the 2D evaluations previously tested, because of the more careful choice of training examples and because the number of training examples was approximately double than used in the 2D tests. This results were obtained using classic neurons trained by gradient descent, which at this low resolution performed better than the fuzzy models.

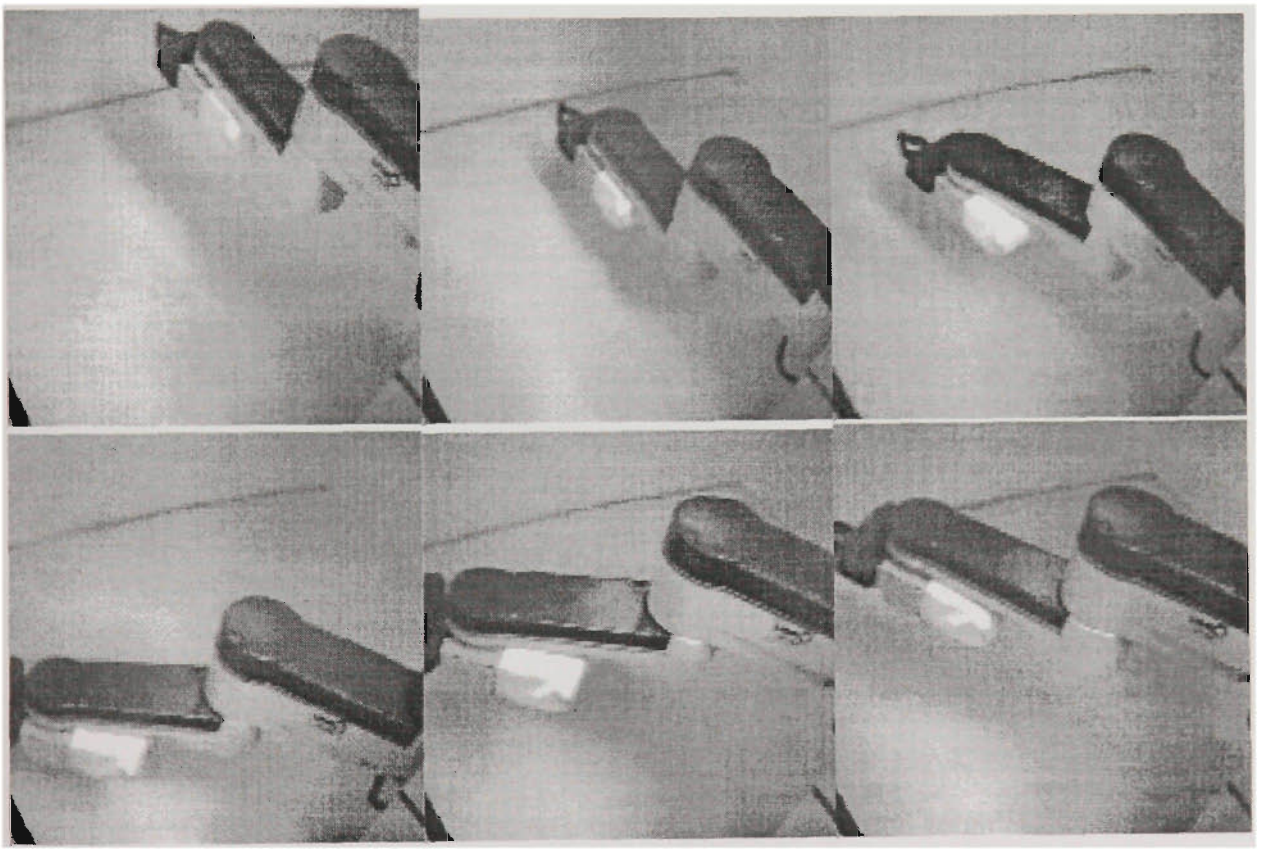


Figure 6.35: Images of the arm as seen by the robot 'eye' during a 3D performance

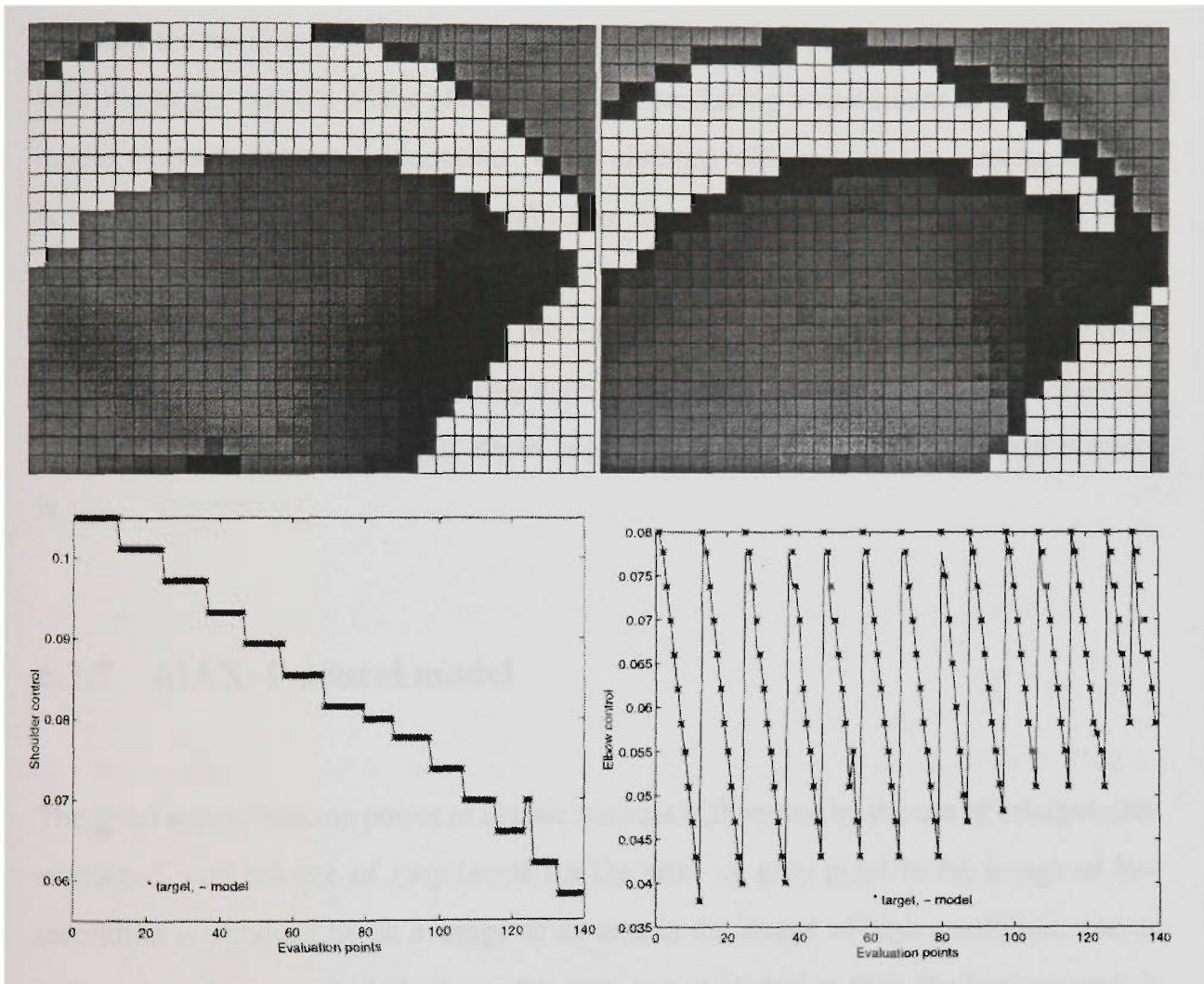


Figure 6.26: Model evaluation for high resolution input

as the knowledge coded in the weight space is directly interpretable.

The models of eye-arm coordination were validated in 2D tests in which the robot imitated the movements of a human arm. The models were also demonstrated for the 3D case, in which the robot imitated another robot of identical appearance.

Chapter 7

Conclusion

Learning from a teacher versus learning by exploration. The proposed approach addressing the development of robot apprentices is influenced by the ideas promoted by Brooks on robots which are situated, embodied, and progressively acquire more complex behaviors [Brooks 1991]. The majority of systems influenced by these ideas concentrate on exploration (and selection mechanisms), which is the principal mean of acquiring new behaviors. I support the idea that in an attempt to build robot apprentices, which can learn behaviors from a teacher, the emphasis should be on teaching-learning mechanisms rather than on exploration. It was observed in natural systems that the longer the period of immaturity of a species, the greater its ability to adapt to a changing environment [Bruner 1972]. Thus rather than launching itself in an exhaustive explorational adventure in the rapacious world, the robot can *learn from a 'parent'*, a system with similar appearance and with knowledge on how to perform tasks in the environment. The parent could have had acquired the knowledge from its own exploration or could have had learned itself from a teacher .

7.1 Contributions

This thesis has addressed aspects of fuzzy neural networks and aspects of motor learning in anthropomorphic robots. Two main claims have been made. The first claim is that anthropomorphic robots can learn the eye-arm coordination which enables them to learn arm movements by imitating arm movements of a human instructor. To enable this, neural models of eye-arm coordination have been proposed, and a technique was deviced for generating training examples for learning in this models. The approach was successfully demonstrated by simulations and by practical tests with a robot that imitated 2D movements of a human arm and 3D movements of an identically built robot.

The second claim of this thesis is that fuzzy neural networks have advantages over classic neural networks in building learning and control structures for such robots. To enable the use of fuzzy neural models, theoretical and practical aspects of learning in relational structures have been investigated, adapting existing results and contributing new constructions such as novel neuron models and learning mechanisms. The advantages shown for the fuzzy models are related to their transparency, portability, robustness and the mechanism of learning. The transparency is related to the shape of the neural weight matrix, which acts as a filter on the inputs, and which allows the prediction of the output for any input situation. The portability refers to the fact that a model obtained by learning from one arm can be used to respond to images of other arms. The robustness reflects the filtering property of the weights, which are relatively insensitive to variations of lighting, color of clothing worn by the human arm, or presence of objects outside the arm envelope, or inside the envelope if the arm is darker than the objects. Learning by fuzzy models has the advantage of being on-line, incremental, and corresponding to analytical methods of resolution of associated fuzzy relational equations.

The following is a brief summary of contributions made in this thesis.

- It has brought together results which can be considered the basis of a theory of fuzzy neural systems. Included here are the relational approach to system modelling, the theory of fuzzy relational equations (FRE) and the model of fuzzy logic neuron.

- It has shown that the S-T composition brings advantages to system modelling mainly by extending the class of problems subject to relational modelling.

- A classification of fuzzy reasoning methods has been proposed, which reflects directly in alternatives of neural implementation.

- Triangular norms have been compared on the basis of their suitability for implementing synaptic/somatic neural operators. The criteria used for this comparison favoured neural learning and adaptation abilities.

- The fundamental fuzzy neuron (FFN) has been defined, based on the fundamental t-norms, which were selected as optimal for implementing fuzzy neurons (in software and hardware).

- Gradient descent equations for FFN networks have been determined, providing a method of learning in such structures, as well as offering a numerical method of resolution of S-T FRE.

- It has been shown that synaptic modification can be used as an additional parameter for adaptation in FFN, which is not possible for classic neurons.

- It has been exemplified how learning in FFN leads to the resolution of a variety of FRE, and how it leads to better modelling when combined with synaptic/somatic adaptation. This combined optimality of weights and synaptic/somatic parameters is equivalent to a system identification method which addresses an optimal fuzzy relation - fuzzy composition pair.

- The 'rules in weights' representation has been demonstrated, which differs from the 'rules in neurons' representation commonly emphasized in neuro-fuzzy systems.

The 'rules in weights' representation offers transparency to fuzzy neural models, which is an important advantage over classic neural networks often described as 'black boxes'[1].

- The fuzzy neuron with shared weights (FNSW) has been defined, allowing the direct implementation of sampled multi-input fuzzy systems which perform distributed associative fuzzy reasoning.
- It has been shown how multi-input systems can be identified using FNSW.
- The 'rules in weights' representation has also been shown for multi-input systems, where 'slices' of neural space represent distributed rule tables.
- It has been shown that any boolean function can be implemented using only one FNSW, and it was argued in favor of FNSW as a general computational element.
- It has been proposed to build anthropomorphic robot apprentices that learn by imitating human movements.
- It has been argued that eye-arm coordination is necessary for vision-based motor skill acquisition.
- A method of generating training examples for building an eye-arm coordination has been proposed. The basic idea is that initially the human should imitate the robot, to allow the correlation of images of the human arm with commands to the robot arm.
- A neural model of eye-arm coordination has been proposed and demonstrated with both classic and fuzzy neurons.
- It has been shown how results from the theory of fuzzy relational equations can be applied in the practical case of learning eye-arm coordination. A necessary condition for the resolution of a system of FRE has been provided indicating a practical scheme for creating solvable models.

[1]The extent to which this transparency applies to other problems is yet to be investigated.

- The αm-composition has been defined, providing a solution for a system of MAX-MIN FRE; in the particular situation considered in the study, the solution was easier to interpret than the maximal solution of MAX-MIN FRE obtained by α-composition.

- A learning algorithm (MBMS) has been proposed, which offers an approximate solution for MAX-MIN FRE in the context of learning from images. The algorithm gives a solution and a filter. The solution was not as good as the one obtained by α-composition. However, applying the filter to the solution obtained by α-composition leads to a solution of increased robustness and transparency.

- It has been shown that both the increased resolution of input images and the increased number of training examples, lead to better fuzzy neural models.

- Better interpolation has been noted when using grey-scale images for the arm than for simple binary images (obtained using a thresholding technique to isolate the arm from the background).

- It has been shown that 3D movements of a robot arm could be imitated by another robot which perceived them using a single camera.

Some of the listed contributions have a high degree of generality, while others may be strictly connected to the particular application, and their extension in other situations has still to be investigated. All the results demonstrated for fuzzy neural networks are general. The delicate point is the transparency and ease of interpretation of the 'rules in weights' interpretation, which although shown for simple examples and for the 2D case of eye-arm coordination model, needs further study so that they may be fully characterized. In particular, for systems that admit a multitude of solutions, a question that needs to be answered is how to determine the solution whose representation makes most sense for humans (in the case of eye-arm coordination the technique found useful was to minimize the solution, starting with α-composition, continuing with αm and α0-composition and filtering by MBMS filter). One issue which was not made explicit

is that the choice of maximal elbow values associated with the extended arm posture was important for the model obtained (the choice of minimal values would not have lead to the same model). This suggests that even before choosing a solution for the fuzzy relational equation, the formulation of the fuzzy relational model itself needs careful consideration in the search for the interpretable fuzzy neural models.

The approach for the acquisition of motor skills by imitation can be applied to the extent that the teaching arm and the apprentice arm are similar. Learning associations with the method in which the teacher imitates the apprentice is more general that the context of learning arm movements from similar creatures; in fact the teacher and apprentice do *not* need to look alike at all. The purpose of teacher imitation was to provide a visual pattern associated with a command given to the robot arm. In a different context, this pattern can be of completely different nature, or even a word. For instance, when the robot generates a command to move to the left, the image of a cat can be shown, or the word 'left' said, and when the robot generates a command to move to the right, the picture of a fish can be presented, or the word 'right' said. This would 'condition' the robot to move to the left whenever it sees the cat, or hears 'left', and move to the right whenever it sees the fish, or hears 'right'. The possibility of conditioning was demonstrated for example in the robot MAVIN [Baloch and Waxman 1991], which associates reflex motor behaviors with certain learned objects, demonstrating classical conditioning paradigms [2].

[2]Mobile Adaptive VIsual Navigator (MAVIN) [Baloch and Waxman 1991] is a mobile robot based on dynamical neural networks, which takes as inputs various patterns of light (corresponding to 3D objects) and learns objects invariant to location, size, orientation, angle of gaze and aspect on the visual field. It also learns to associate motor behaviors to visual objects demonstrating behavioral conditioning.

7.2 Future work

Fuzzy neural models. More investigation is needed in fuzzy neural models. It has been suggested in this thesis that fuzzy neural solutions can have advantages over classic neural ones, although the extent to which the advantages extend to other situations needs to be clarified. In the application discussed, the models did not benefit from the full range of capabilities that fuzzy neurons have, such as using inhibitory inputs or shared weights. A very useful contribution would be an analytical method of resolution of S-T FRE.

Variable direction of gaze and changing environments. There are two main problems which need to be solved before the approach to learning by imitation proposed in this thesis can become feasible outside the laboratory. The first problem derives from the fact that, in the tests performed, a fixed direction of gaze (a fixed position of the eye in relation to the arm) was considered for learning and the *same* direction was used for interpretation of the input images. To some extent, humans also learn mappings of fixed orientation; the easiest way to control a telemanipulator is to face the same direction (as illustrated in Figs 1.1, 1.2). This is also true in learning arm movements from a coach as in martial arts or fencing. However, humans have a high degree of adaptation to variations from the learned position. In order to allow the robot to imitate while watching from a variety of positions, the immediate technique would be to include a variety of positions in the set of training examples. This can be done by having the camera take arm images of a given posture as seen from different points in space and using the examples in supervised training. The structure with one neuron per joint is much too simple for this case; an alternative would be to allocate processing to two neural structures, a first neural structure that performs a rotation/transformation of the image to a given pattern (which has for example the shoulder centred in the bottom of the image), and a second neural structure that does the mapping as already exemplified. A mechanism that may

help is to impose a kind of initial 'calibration' through a couple of simple 'put your arm like mine' posture agreements. This would help the robot to perform the rotation of incoming images to suit the internal model. Image processing algorithms can probably play a role.

The second problem comes from learning in changing environments, more precisely, with various backgrounds behind the teaching arm. It is assumed that no object obstructs the arm, which would be quite unusual for human motor skill acquisition as well. It was seen that fuzzy neural networks can develop representations that allow them to ignore objects outside an envelope limited filter. This was possible with noisy [3] *static* backgrounds (even when the noise intensity was higher than that of the signal), which do not vary from one example to another. One can possibly restrict the learning of eye-arm coordination to a static environment, while the imitation is performed in an environment with a variable background outside the arm envelope. However, for variable noisy patterns *inside* the envelope, the one neuron model is powerless. The way to cope with this situation may be to first recognize and separate the arm from the background, and only then to provide the preprocessed input to a neural structure that maps it to a motor command.

Correlating eye-arm coordination, eye-hand coordination and object manipulation. The focus in this thesis was on learning eye-arm coordination, because it was considered important for learning from an instructor and because it was not treated before in the literature. However, for achieving full potential, eye-arm coordination should be combined with eye-hand coordination. The two should also correlate with the object to be manipulated and the task. For example in learning how to use a hammer, the focus should not be on the correct trajectory only, but also on whether the hammer hits the nail properly or not! Once the trajectories are determined, the motor sequence for producing them should be transformed in motor patterns, reducing the amount of stored

[3]Here *noise* includes everything except the arm and pure 'white' background.

information required, and optimizing the movement for some chosen indices. Fully anthropomorphic arms need to be developed to gain the most of the approach proposed in this thesis.

Language acquisition in correlation with movements. An interesting direction of research would be to integrate the development of visuo-motor coordination with language acquisition. This can be seen as a modification of the Miniature Language Acquisition task [Feldman *et al.* 1990] (proposed for language descriptions of visual scenes) to aim at language acquisition in correlation with movements. The task would be to build a robot that, when trained on a sequence of motion motor commands - motion linguistic descriptions - motion visual perception pairs, will learn to 'understand' the language, where understanding is measured by the system's ability to correctly perform movements according to their descriptions. Words from a possible vocabulary could refer, for example, to actors (*robot, teacher*), arm parts (*shoulder, elbow, wrist, arm, forearm*), actions (*move, stop, continue*), direction (*up, down, left, right*), distance (*close, far*) or speed (*fast, slow*). During learning, the words are gradually offered to the robot in the context of a movement that has the particular characteristic, thus establishing correspondent associations. The learned mappings are seen as reversible, in the sense that the robot can describe linguistically a movement it performs or sees, or can respond to the linguistic command by performing its own movement.

Learning to imitate. Instead of being programmed to imitate, the robot can learn to do so[4]. Imitation can be developed as the consequence of an evolutionary process or by reinforcement, for example as proposed in [Stoica 1994]. By reinforcing the accidental coincidence or by favorizing coincidence of actions, e.g. by human imitation, a robot may be controlled into developing imitation.

[4]A distinction is made here between learning to imitate and learning how to imitate, the later being addressed in this thesis.

Developmental robotics. In the recent years new directions in robotics research have emerged. These include, for example, behavior-based robotics, perceptual robotics, cognitive robotics and evolutionary robotics. An increasing number of researchers use robotics as a vehicle for the exploration of theories in disciplines dealing with living organisms. For example the name of a popular conference 'From animals to animats' reflects such a tendency. The benefits are mutual as robots become more versatile and incorporate solutions from nature. In the same way as the solution to the 'symbol grounding' problem is expected to come from robots acting, perceiving the world and building their own representations, it may be the case that an efficient ability of 'being taught' is not programmable but would need a developmental process, with robots building representations of their own actions, perceptions, and interactions with humans, in a way similar to child development. Investigation in this area could lead to a new direction of robotics research, possibly called *developmental robotics*, aiming at building robots based on mechanisms similar to human cognitive and motor development.

Appendices

Appendix A

Some properties of MAX-MIN, MAX-T and S-T compositions.

A summary of some properties of MAX-MIN, MAX-T and S-T composition is presented in Table A.1. E is a matrix of 1's, I is the identity matrix and H is a matrix of 0's.

Table A.1: Some properties of MAX-T and S-T compositions

No.	Property	MAX-MIN	MAX-T	S-T
1	$H < A < E$	T	T	T
2	$A^m = A \circ A \circ .. A$ (m times) If A fuzzy matrix, so is A^m	T	T	T
3	$A \circ H = H \circ A = H$	T	T	T
	$A \circ I = I \circ A = A$	T	T	T
4	$A^{m+n} = A^m \circ A^n$	T	T	F
	$A^{mn} = (A^m)^n = (A^n)^m$	T	T	F
5	$A \circ (B \circ C) = (A \circ B) \circ C$	T	T	F
6	$A < B, C < D, A \circ C < B \circ D$	T	T	T
7	$A \circ B \neq B \circ A$	T	T	T
8	$A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$	T	T	F
9	$A \circ (B \cap C) \leq (A \circ B) \cap (A \circ C)$	T	F	F
10	$B \leq C \Rightarrow A \circ B \leq A \circ C$	T	T	T

Appendix B

Comparison of neuro-fuzzy models

Citations in Table 1.1: [1] = [Lee and Lee 1975], [2] = [Takagi and Hayashi 1991], [3] = [Takagi *et al.* 1992], [4] = [Horikawa *et al.* 1992], [5] = [Jang 1992], [6] = [Pedrycz and Rocha 1993], [7] = [Gupta 1992], [8] = [Keller and Krishnapuram 1992], [9] = [Keller and Tahani 1992], [10] = [Pedrycz 1991], [11] = [Blanco *et al.* 1994], [12] = [Yamakawa *et al.* 1992], [13] = [Uchino and Kubo 1994].

Table B.1: Comparison of some neuro-fuzzy models

Authors	Architecture	Type of neuron	Significance of neurons	Significance of weights	Learning	Destination
Lee [1]	fully connected, one common input	non-fuzzy, arithmetic	basic element for a state machine	all 1's, full transmission of signal	no	synthesis of fuzzy automata
Takagi [2] or [3] Horikawa in [4] Jang [5]	multilayer, layers map stages of processing in FS	non-fuzzy, either sum-product-sigmoid or bell-shape-product [5]	depends on layer contribution of each rule	parameters for tuning	GD (BKP)	configure, implement FS with adaptive charact., automatic extraction of memb.func and rules
Pedrycz, in [6] Gupta in [7]	multilayer, maps the logic of the problem to a logic oriented net	fuzzy (OR, AND,MATCH) in some work based on a parametric t-norm (Hamacher)	aggregative + reference n. logic processor some represent contribution of rules	parameter	GD, other sugested	knowledge based networks
Keller in [8], [9]	multilayer inputs outputs -discretized I/O space	fuzzy, of different types, each implementing a Yager operator [8] classic neuron [9]	logical processors implementing a specific type of logic operator [8] 2. processing element [9]	all are 1's full transmission of logic value [8] 2. parameter [9]	global search for param of logic operator [8] 2.BKP [9]	networks for fuzzy logic inference
Pedrycz [10] Blanco [11]	one layer	non-fuzzy or max-min fuzzy	processing element	parameters elements of FRE	GD, GD/GA in GAREL	solve FRE from I-O data
Yamakawa in [12] Uchino in [13]	neo-fuzzy neuron	arithmetic summing non-linear synapses performing Sugeno type of inference	neuron is a complete fuzzy system	singleton outs for one variable	modified GD	identification of nonlinear dynamic systems
This thesis	one layer for fuzzy reasoning, same mapping as Keller, discretized I/O space	fuzzy s-t (similar with Pedrycz's OR and Gupta's neuron), calculated with fundamental t-norms	elementary processor for fuzzy reasoning	shape distributed fuzzy rules, fuzzy relation	hebbian, GD, MBMS COAT	find best model (find fuzzy relation and composition)

Appendix C

Example of modelling by S-T composition

The following example illustrates a situation, in which the S-T composition offers better modelling than the MAX-MIN composition. The displaying technique is that of 'Chernoff faces'[1]. In Fig. C.1 the children A and B resemble in some degree to their brothers C and D, which in turn resemble in some degree to the parents. A and B resemble their parents more than what can be inferred using MAX-MIN. Each similar ear, nose or mouth are given 1 point, while the eyes are given 4 points. The resemblance is defined as (number of points)/10. The fuzzy relations are:

$$Q_{A,B-to-C,D} = \begin{pmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{pmatrix} \quad R_{C,D-to-F,M} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.7 \end{pmatrix} \quad P_{A,BtoF,M} = \begin{pmatrix} 0.8 & 0.6 \\ 0.5 & 0.9 \end{pmatrix}$$

In the last relation there are elements with higher values than in the first two relations. As stated by condition 2.7 this situation can not be modeled with MAX-T composition (The use of MIN-S restricts in the opposite direction).

[1]Chernoff faces ([Chernoff 1973] referred to in [Krzanowski 1993]) are a display technique proposed for monitoring systems of many variables. As facial patterns are easily perceived by humans, the shape of eyes, ears, nose and mouth are correlated with different values of the variables.

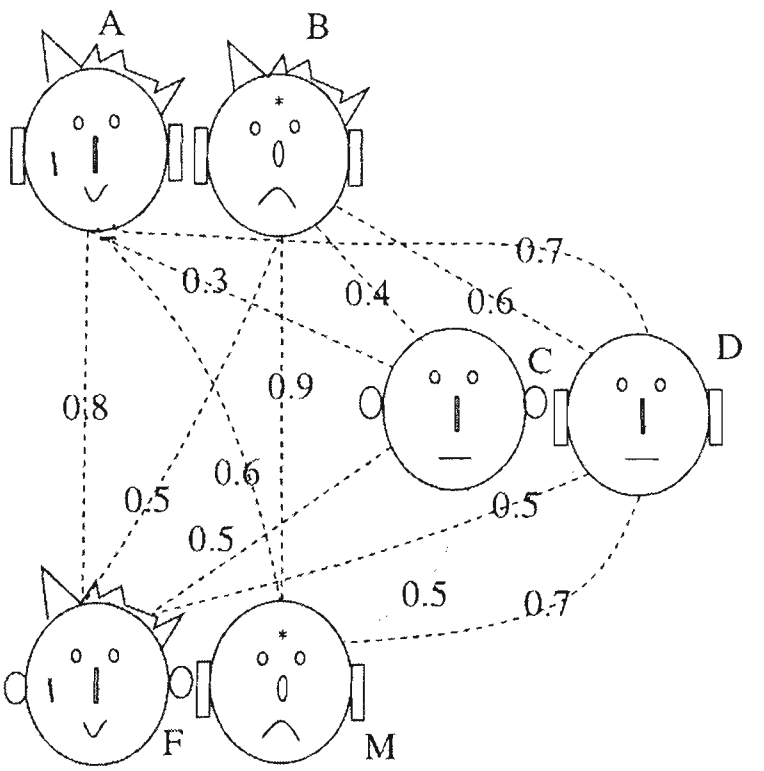


Figure C.1: Chernoff faces illustrating resemblance relations

Appendix D

Fuzzy Hebbian learning

In many cases the result of fuzzy hebbian learning can be a good initial approximation for a FRE solution, which can be further tuned by using GD. Fuzzy hebbian learning supports incremental learning.

Classic Hebb law refers to changes in synaptic strength r_{ij} as depending on simultaneous firing of x_i, y_j. In the context of logic neurons, the law can be interpreted as $r_{ij} = r_{ij}$ OR (x_i AND y_j). Incremental learning starting with a null connection strength:

$$r_{ij}^0 = 0 \tag{D.1}$$

after the first presentation,

$$r_{ij}^1 = r_{ij}^0 S(x_i^1 T y_j^1) = S(r_{ij}^0, T(x_i^1, y_j^1)) = T(x_i^1, y_j^1) \tag{D.2}$$

after the second

$$r_{ij}^2 = S(r_{ij}^1, T(x_i^2, y_j^2)) = S(T(x_i^1, y_j^1), T(x_i^2, y_j^2)) \tag{D.3}$$

and the third,

$$r_{ij}^3 = S(r_{ij}^2, T(x_i^3, y_j^3)) = S(S(T(x_i^1, y_j^1), T(x_i^2, y_j^2)), T(x_i^3, y_j^3)) \tag{D.4}$$

which after using Equ. (2.17) this generalizes to

$$r_{ij}^p = \underset{k=1}{\overset{p}{S}}(T(x_i^k, y_j^k)) \tag{D.5}$$

The vectorial form can be written as

$$R = S(T(X^T, Y)) = X^T \circ Y \tag{D.6}$$

where $\circ$ can be any S-T composition. Observe that hebbian learning can be driven by the same type of fuzzy neurons, whose output in this case determines the weights of other fuzzy neurons.

Hebbian learning is strongly affected by the choice of synaptic and somatic parameters. Large values of the somatic parameter s_S and small values of the synaptic parameter

s_T lead to the fastest increase in the weights, in effect it often leads to saturation. The surface shaped by R is non-decreasing at the presentation of each new learning example. To reduce this surface a decay effect may be considered, or a GD technique can be employed for finer shaping of the weights.

The solution obtained by hebbian learning is close to the solution of the FRE. Fig. D.1 shows the shape of the solution obtained by hebbian learning, which can be compared with the solution of FRE illustrated in Fig 3.8.

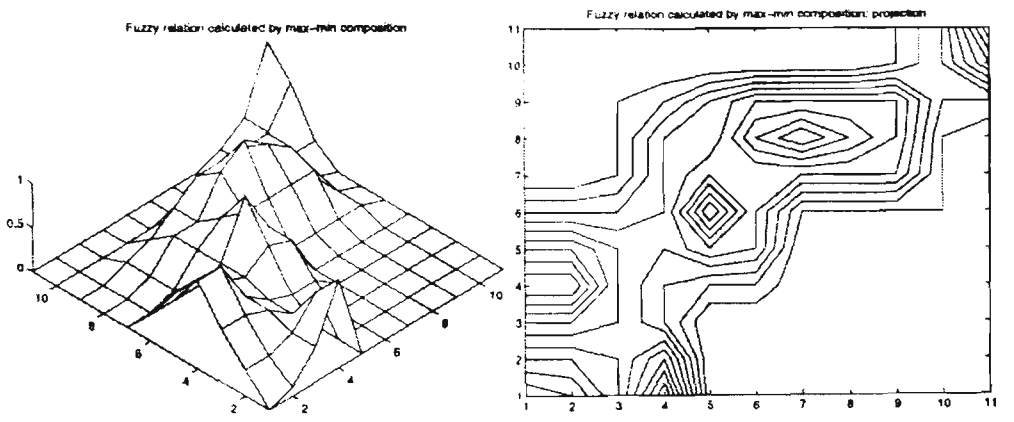


Figure D.1: Fuzzy relation determined by hebbian learning

Appendix E

Training pair for learning fuzzy distributed representations

Training pairs formed by the discretised representations of mapping sets in Example from Section 3.5.

The generated data is:

In_1 = (1 0.7 0.3 0 0 0 0 0 0 0 0); Out_1 = (1 1 0.5 0.3 0 0 0 0 0 0 0);

In_2 = (0 0.3 0.6 1 0.7 0.3 0 0 0 0 0); Out_2 = (0 0.1 0.4 1 0 0 0 0 0 0 0);

In_3 = (0 0 0 0.2 0.6 1 0.6 0.4 0.1 0 0); Out_3 = (0 0 0.1 0.4 1 0.5 0 0 0 0 0) ;

In_4 = (0 0 0 0 0 0.1 0.5 1 0.6 0 0); Out_4 = (0 0 0.1 0.2 0.4 0.8 1 0.8 0.6 0.1 0);

In_5 = (0 0 0 0 0 0 0 0 0.2 0.7 1); Out_5 = (0 0 0 0 0 0 0 0 0.1 0.3 1);

Appendix F

Application of FFNN to FRE resolution: working examples

Example 1: An exact solution

The following MAX-MIN FRE (from [Blanco *et al.* 1994]) admits an exact solution

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 0.6 & 0.5 & 0.8 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.9 & 0.6 & 0.4 \\ 0.1 & 0.1 & 0.9 & 0.8 & 0.5 \\ 0.9 & 0.2 & 0.9 & 0.1 & 0.5 \\ 0.4 & 0.5 & 0.3 & 0.8 & 0.9 \end{pmatrix}$$

Example 2: An approximate solution

The problem is from [Pedrycz 1990*a*], and was also used as a test case in [Negoita *et al.* 1994]. The system $XoW = Y$ does not admit exact solutions and an approximate solution is sought.

$$X = \begin{pmatrix} 1.0 & 0.4 & 0.3 & 0.1 \\ 0.4 & 1.0 & 0.6 & 0.4 \\ 0.5 & 0.5 & 0.8 & 0.9 \\ 0.6 & 0.4 & 0.7 & 1.0 \\ 0.9 & 0.6 & 0.4 & 0.3 \\ 0.5 & 0.6 & 0.8 & 1.0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0.5 & 0.5 & 0.7 & 1.0 \\ 0.3 & 0.3 & 1.0 & 0.9 \\ 0.4 & 0.2 & 0.2 & 0.6 \\ 0.5 & 0.6 & 1.0 & 0.3 \\ 0.4 & 0.3 & 0.5 & 0.7 \\ 1.0 & 0.6 & 0.3 & 0.2 \end{pmatrix}$$

The search was done for a S-T network with s = 0.000001. The results obtained by GD are compared with results from [Pedrycz 1990*a*] and more recent results from [Negoita *et al.* 1994]. Criteria considered for comparison include maximum error per pair (M), maximum error per matrix (MM), sum of absolute errors (SAE) (fuzzy Hamming distance between target and obtained outputs, which is the performance index used in [Pedrycz 1990*a*], [Negoita *et al.* 1994]), and the sum of squared errors (SSE) which is the performance index in the GD search.

Notations: P - refers to results obtained in [Pedrycz 1990*a*], N - results obtained in [Negoita *et al.* 1994], G - results obtained here by GD. E - error matrix expressing the distance between target outputs and those obtained by S-T composition with determined solutions W. Values used in training: number of iterations = 200, s = 0.000001 (Y_G is obtained by proper MAX-MIN).

The results are:

$$W_P = \begin{pmatrix} 0.4 & 0.3 & 0.7 & 0.7 \\ 0.4 & 0.3 & 1.0 & 0.9 \\ 0.4 & 0.3 & 1.0 & 0.6 \\ 0.5 & 0.6 & 1.0 & 0.6 \end{pmatrix} \quad W_N = \begin{pmatrix} 0.40 & 0.37 & 0.7 & 1.0 \\ 0.37 & 0.30 & 1.0 & 0.9 \\ 0.35 & 0.0 & 0.25 & 0.37 \\ 0.5 & 0.6 & 0.25 & 0.6 \end{pmatrix} \quad W_G = \begin{pmatrix} 0.437 & 0.406 & 0.661 & 0.913 \\ 0.109 & 0.174 & 0.999 & 0.897 \\ 0.165 & 0.142 & 0.111 & 0.251 \\ 0.632 & 0.409 & 0.223 & 0.250 \end{pmatrix}$$

$$Y_P = \begin{pmatrix} 0.4 & 0.3 & 0.7 & 0.7 \\ 0.4 & 0.4 & 1.0 & 0.9 \\ 0.5 & 0.6 & 0.9 & 0.6 \\ 0.5 & 0.6 & 1.0 & 0.6 \\ 0.4 & 0.3 & 0.7 & 0.7 \\ 0.5 & 0.6 & 1.0 & 0.6 \end{pmatrix} \quad Y_N = \begin{pmatrix} 0.4 & 0.3 & 0.7 & 1.0 \\ 0.4 & 0.4 & 1.0 & 0.9 \\ 0.5 & 0.6 & 0.5 & 0.6 \\ 0.5 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.37 & 0.7 & 0.9 \\ 0.5 & 0.6 & 0.6 & 0.6 \end{pmatrix} \quad Y_G = \begin{pmatrix} 0.437 & 0.406 & 0.661 & 0.913 \\ 0.400 & 0.400 & 0.999 & 0.897 \\ 0.632 & 0.409 & 0.500 & 0.500 \\ 0.632 & 0.409 & 0.600 & 0.600 \\ 0.437 & 0.406 & 0.661 & 0.900 \\ 0.632 & 0.409 & 0.600 & 0.600 \end{pmatrix}$$

$M_P = [0.3, 0.1, 0.7, 0.3, 0.2, 0.7]$,

$M_N = [0.13, 0.1, 0.4, 0.4, 0.2, 0.5]$,

$M_G = [0.094, 0.1, 0.3, 0.4, 0.2, 0.4]$

$MM_P = 0.7$,	$MM_N = 0.5$,	$MM_G = 0.4$
$SAE_P = 4.1$,	$SAE_N = 3.6$,	$SAE_G = 4.1126$
$SSE_P = 1.85$,	$SSE_N = 1.1418$,	$SSE_G = 1.0436$

Example 3: Adaptive composition gives a better approximate solution for Example 2

A combined search for W and s on the data set used in Example 2, gives after 25 steps a weight matrix and a logic (s = 12970000), for which the sum of squared errors is $SE2_{gd} = 0.93$. This is a better approximation than obtained for MAX-MIN composition, for which $SE2_{gd} = 1.04$.

Example 4: The simultaneous identification of the fuzzy relation and the composition

The focus is on a system for which it is known that an exact solution exists, as the output Y is determined by the S-T composition (s = 10) of input X with W.

$$X = \begin{pmatrix} 0.21 & 0.52 & 0.52 \\ 0.04 & 0.67 & 0.09 \\ 0.67 & 0.00 & 0.65 \\ 0.67 & 0.38 & 0.41 \\ 0.93 & 0.06 & 0.70 \\ 0.38 & 0.41 & 0.91 \\ 0.51 & 0.68 & 0.76 \\ 0.83 & 0.58 & 0.26 \\ 0.03 & 0.93 & 0.04 \\ 0.05 & 0.84 & 0.73 \end{pmatrix} \quad W = \begin{pmatrix} 0.32 & 0.99 & 0.98 \\ 0.63 & 0.36 & 0.72 \\ 0.75 & 0.24 & 0.75 \end{pmatrix} \quad Y = \begin{pmatrix} 0.58 & 0.38 & 0.72 \\ 0.41 & 0.22 & 0.50 \\ 0.56 & 0.72 & 0.86 \\ 0.54 & 0.73 & 0.87 \\ 0.69 & 0.94 & 0.97 \\ 0.80 & 0.59 & 0.91 \\ 0.81 & 0.72 & 0.94 \\ 0.61 & 0.87 & 0.94 \\ 0.58 & 0.34 & 0.68 \\ 0.82 & 0.42 & 0.87 \end{pmatrix}$$

Appendix G

A classic neuron model

The classic neuron model referred in this thesis is a perceptron modulated by a squashing sigmoid.

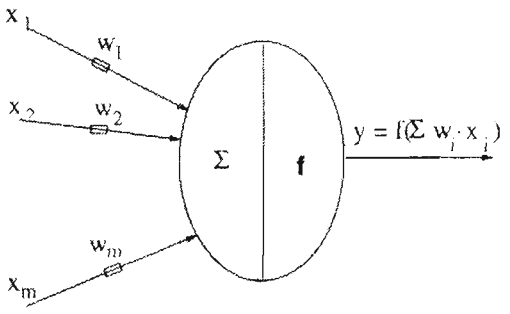


Figure G.1: Classic neuron model

The neuron performs a summation of weighted inputs,

$$a = \sum w_i \cdot x_i \tag{G.1}$$

to which applies a nonlinear sigmoid function

$$y = f(a) = \frac{\gamma}{1 + e^{-\beta a}} \tag{G.2}$$

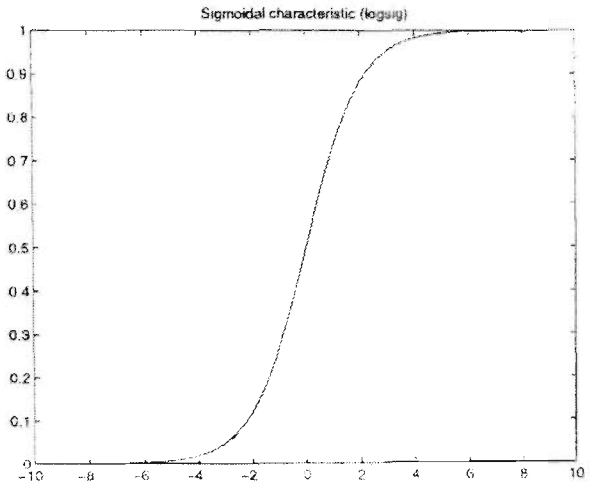


Figure G.2: Logsig characteristic

With $\gamma = 1$ the function takes values in [0, 1] interval. The function **logsig** (logarithmic-sigmoid) implemented in Matlab, has $\gamma = 1$ and $\beta = 1$.

Appendix H

A coach's perspective on teaching motor skills

Coaching is very important in developing motor skills. Some important aspects of teaching humans how to move are itemised in the following [1].

- Movements are learned in increasing order of their complexity, starting with simple movements and progressing toward complex movements, which usually are presented in terms of simple movements (Progressive).
- A complex movement can be presented as an ordered sequence of simple movements (Order).
- When simple movements are introduced, it is mentioned in what context the respective movement will be useful (Utility).
- Movements (especially complex ones) are presented in 3-D perspective (front and side profile). Note that those who show the movements do not always have the

[1]The list is the result of personal observations and not the result of a scientific study (which couldn't be found as such in the literature). Its purpose was to help creating skill transfer scenarios inspired after the human experience.

same length of the arm as those who learn, but the learner has the capability to grasp the essential (Spatiality).

- Significant points of the trajectory (prototypes) are pointed out with the hand or indicated in words (Prototypes).

- The attention is concentrated on one movement at a time (Attention).

- Words help in a variety of situations such as in remembering and generalising what one sees, in naming the elementary movements, in describing the complex ones and in pointing where to concentrate attention (e.g. in what you see). Descriptive fuzzy statements often accompany the exemplified movement ('Push your right hand slower and withdraw it quickly, like this'). 'The angle between the chest and the shoulder is 90 degrees' (Language).

- The teacher has the key role in feedback. A jumper made the following comments referring to his couch: 'Mark is my eyes, I don't know what I am doing wrong'. The coach points out important points of the trajectory 'prototypes for the trajectory', and indicates wrong trajectories (Feedback).

- Wrong movements can be shown, and explained why they are wrong. This is a case of examples associated with a negative reinforcement (Negative examples).

- The actions are divided into small elements, and one needs to systematically evaluate the athlete performance, focus concentration on one thing a time, and decide where the most fundamental lies. It is important not to give many changes at once. (Strategy).

- After a movement is learned it becomes a reflex action. The cognitive phase is no longer involved. 'Fencing is also a mental game. Once a fencer has practised the various movements until he is physically able to carry out a plan without having to think about how the various parts of the body must move, he finds that the real excitement lies in outthinking and outwitting his opponent. You must quickly analyse your adversary's style and then plan your strategy accordingly. You must set traps for your opponent while being careful to avoid those set by him.' [Bower and Mori 1986] (Reflex).

Appendix I

TINMAN's behaviours

TINMAN'S SOFTWARE IMPLEMENTATION

FUNCTIONAL FLOWCHART

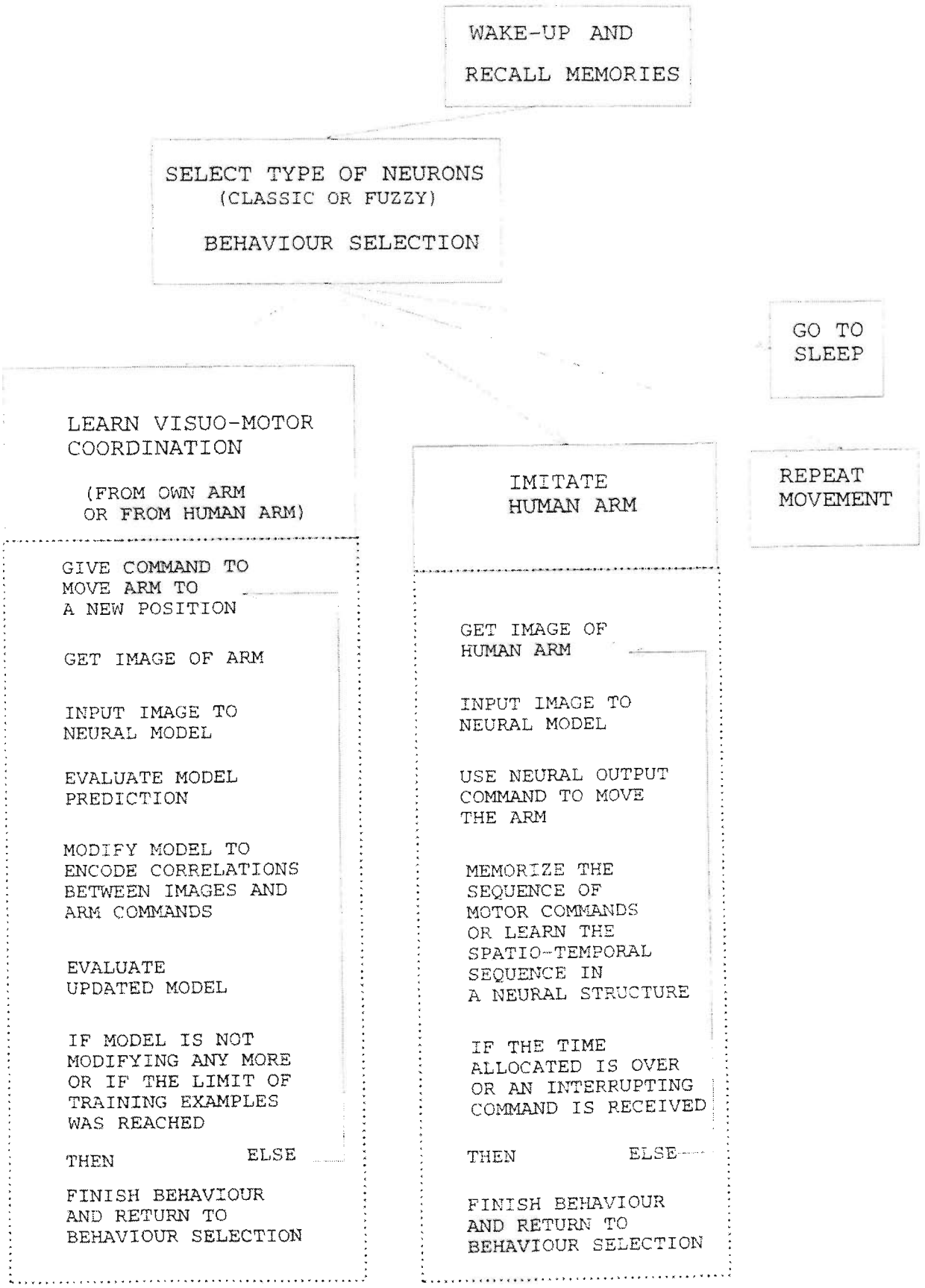


Figure I.1: TINMAN's behaviours

Bibliography

ANDERSON R. L. (1988). *A robot ping-pong player: experiment in real time intelligent control*. MIT Press.

ANDREWS R., J. DIEDERICH AND A. B. TICKLE (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. Technical report. Neurocomputing Research Center, Queensland University of Technology. Box 2434 GPO Brisbane 4001, Queensland, Australia.

ASADA H. AND S. LIU (1990). Acquisition of task performance skills from a human expert for teaching a machining robot. in *Proc. of the 1990 American Control Conference*. pp. 2827–2832.

ASADA H. AND S. LIU (1991). Transfer of human skills to neural net robot controllers. in *Proc. of 1991 IEEE International Conference on Robotics and Automation*. IEEE. IEEE. pp. 2442–2448.

ATKESON C. G. (1990). Memory-based techniques for task-level learning in robots and smart machines. in *Proc of the 1990 American Control Conference, San Diego, CA*. pp. 2815–2820.

BALOCH A. A. AND A. M. WAXMAN (1991). Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot mavin. *Neural Networks* Vol. 4, pp. 271–302.

BELMANS P. (1990). An approach to elemental task learning. in *SPIE Vol. 1293 Applications of Artificial Intelligence VIII*. pp. 634–644.

BLANCO A., M. DELGADO AND I. REQUENA (1994). Solving fuzzy relational equations by max-min neural networks. in *Proc of the IEEE Conference on Fuzzy Logic, World Congress on Compuational Intelligence, Orlando, Florida*. pp. 1737–1742.

BOUR L. AND M. LAMOTTE (1988). Equations des relations floues avec la composition conorme-norme triangulaires. *BUSEFAL* Vol. 34, pp. 86–94.

BOWER M. AND T. MORI (1986). *Fencing*. Dubuque, Iowa, W. C. Brown Co.

BROOKS R. A. (1991). New approaches to robotic science. *Science* Vol. 253, pp. 1227–1232.

BROWN M. K., B. M. BUNTSCHUH AND J. G. WILPON (1992). Sam: A perceptive spoken language understanding robot. in *IEEE Trans. on Systems, Man, and Cybernetics, Vol. 22, No.6*. IEEE. pp. 1390–1402.

BRUNER J. S. (1972). Nature and uses of imaturity. *American Psychologist* Vol. 27, No. 8, pp. 687–709.

BUTNARIU D. AND E. P. KLEMENT (1993). *Triangular norm-based measures and games with fuzzy coalitions*. Kluwer Academic Publishers.

BUTNARIU D., KLEMENT E.P. AND S. ZAFRANI (1995). On triangular norm-based propositional fuzzy logics. *Fuzzy Sets and Systems* Vol. 69, pp. 241–255.

CHERNOFF H. (1973). the use of faces to represent points in k-dimensional space graphically. *J. Amer. Statist. Assoc.* Vol. 68, pp. 361–368.

CORKE P. (1992). Dynamic effects in high-performance visual servoing. in *Proc IEEE Int Conf on Robotics and Automation*. pp. Vol 2, 1838–43.

DAVIDOR Y. (1991). A genetic algorithm applied to robot trajectory generation. in *Handbook of Genetic Algorithms* (L. Davis, Ed.). Van Nostrand Reinhold. New York. pp. 145–165.

DEMUTH H. AND M. BEALE (1994). *Matlab: Neural Network Toolbox User's Guide*. MathWorks Inc.

DI NOLA A. AND S. SESSA (1983). On the set of solutions of composite fuzzy relational equations. *Fuzzy Sets and Systems* Vol. 9, pp. 275–285.

DI NOLA A., S. SESSA, W. PEDRYCZ AND E. SANCHEZ (1989). *Fuzzy relation equations and their applications to knowledge engineering*. Kluwer Academic Publishers.

DI NOLA A., W. PEDRYCZ, S. SESSA AND E. SANCHEZ (1991). Fuzzy relation equations theory as a basis of fuzzy modelinng: An overview. *Fuzzy Sets and Systems* Vol. 40, pp. 415–429.

DUBOIS D. AND H. PRADE (1988). *Possibility Theory - An Approach to Computerized Processing of Uncertainty*. Plenum, New York.

ECKMILLER R. (1990). The design of intelligent robots as a federation of geometric machines. in *An introduction to neural and electronic networks* (S. F. Zornetzer, J. L. Davis and C. Lau, Eds.). Academic Press.

EDELMAN G. M. (1987). *Neural Darwinism: The theory of neuronal group selection*. Basic Books. New York.

EDELMAN G. M., G. N. JR. REEKE, W. E. GALL, G. TONONI, D. WILLIAMS AND O. SPORNS (1992). Synthetic neural modeling applied to a real world artifact. in *Proc. Natl. Acad. Sci. USA 89*. pp. 7267–7271.

ENGELBERGER J. (1995). Service robots arise. *International Federation of Robotics Newsletter* No. 18, pp. 1.

ESPIAU B., F. CHAUMETTE AND P. RIVES (1992). A new approach to visual servoing in robotics. *IEEE Trans on Robotics and Automation* Vol. 8, No. 3, pp. 313–326.

FELDMAN J. A., G. LAKOFF, A. STOLCKE AND S. H. WEBER (1990). Miniature language acquisition: A touchstone for cognitive science. in *TR-90-009, Int Comp Sci Inst ICSI*.

FITTS P. AND M. POSNER (1967). *Human performance*. Brooks/Cole Publ. Co.. Belmont CA.

FOULLOY L., S. GALICHET AND E. BENOIT (1994). Fuzzy control with fuzzy state sensors. in *EUFIT'94, Second European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany*. pp. 1156–1160.

FRANK M. J. (1979). On the simultaneous associativity of f(x,y) and x+y-f(x,y). *Aequationes Math.* Vol. 19, pp. 194–226.

GELFAND J., M. FLAX, R. ENDRES, S. LANE AND D. A. HANDELMAN (1992). Acquisition of automatic activity through practice: changes in sensory input. in *Proc. of the Tenth National Conference on Artificial Intelligence AAAI-92*. pp. 189–193.

GOH T. H. WANG P. Z. AND H. C. LUI (1991). Learning algorithm for enhanced fuzzy perceptron. Technical report. National University of Singapore, Inst of Systems Science.

GRAF D. H. AND W. R. LALONDE (1989). Neuroplannerrs for hand/eye coordination. in *Int. Joint Conf. on Neural Networks (IJCNN)*. IEEE. pp. II, 543–548.

GUEZ A., Z. AHMAD AND J. SELINSKY (1992). The application of neural networks to robotics. in *Neural Networks: Current Applications* (P. G. J. Lisboa, Ed.). Chapman and Hall. pp. 111–122.

GUPTA M. M. (1992). Fuzzy logic and neural networks. in *Proc. of Tenth Int Conf on Multiple Criteria Decision Making (TAIPEI'92), vol 3 July 19-24, Japan*. pp. 281–294.

GUPTA M.M. AND J. QI (1991*a*). Design of fuzzy logic controllers based on generalized t-operators. *Fuzzy Sets and Systems* Vol. 40, pp. 473–489.

GUPTA M. M. AND J. QI (1991*b*). Theory of t-norms and fuzzy inference methods. *Fuzzy Sets and Systems* Vol. 40, pp. 431–450.

HARNAD S. (1990). The symbol grounding problem. *Physica D* No. 42, pp. 335–346.

HELLENDOORN H. (1994). Neural-fuzzy: basics and industrial applications. in *Proc of EUFIT'94*.

HIROTA K. AND W PEDRYCZ (1994). OR/AND neuron in modeling fuzzy set connectives. *IEEE Transactions on Fuzzy Systems* Vol. 2, pp. 151-161.

HORIKAWA S., T. FURUHASHI AND Y UCHIKAWA (1992). On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Trans on Neural Networks* Vol. 3, No. 5, pp. 801–806.

HUFFMAN S. B. AND J. E. LAIRD (1994). Acquiring procedural knowledge through tutorial instruction. in *Proc. of the 1994 Knowledge Acquisition for Knowledge-Based Systems Workshop, Banf, Canada* (B. Gaines, Ed.).

JANG J.S.R. (1995). Neuro-fuzzy modeling and control. *Procedings of the IEEE.*

KAUFMANN A. AND M. M. GUPTA (1988). *Fuzzy mathematical models in engineering and management science*. North-Holland.

KELLER J. M. AND H. TAHANI (1992). Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks. *Int'l Journal of Approximate Reasoning* Vol. 6, pp. 221–240.

KELLER J. M. AND R. KRISHNAPURAM (1992). Evidence aggregation networks for fuzzy logic inference. *IEEE Transactions on Neural Networks* Vol. 3, No. 5, pp. 761–769.

KLIR G. J. AND B. YUAN (1994). Approximate solutions of systems of fuzzy relation equations. in *Proc of the IEEE Conference on Fuzzy Logic, World Congress on Compuational Intelligence, Orlando, Florida*. pp. 1452–1457.

KRZANOWSKI W. J. (1993). *Principles of multivariate analysis: a user's perspective.* Oxford University Press.

KUPERSTEIN M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks* Vol. 4, pp. 131-145.

KWAN H. K. AND Y. CAI (1994). A fuzzy neural network and its application to pattern recognition. *IEEE Transactions on Fuzzy Systems* Vol. 2, No. 3. pp. 185–193.

LANE S. H., D. A. HANDELMAN AND J. J. GELFAND (1990). Can robots learn like people do?. in *SPIE Vol. 1294 Applications of Artificial Neural Networks (1990).* pp. 296–309.

LEE C. C. (1990). Fuzzy logic in control systems: fuzzy logic controller - part i and ii. *IEEE Transactions on Systems, Man, and Cybernetics.*

LEE S. C. AND E.T. LEE (1975). Fuzzy neural networks. *Mathematical Biosciences* Vol. 23, pp. 151–177.

LIU S. AND H. ASADA (1992). Transfer of human skills to robots: learning from human demonstrations for building an adaptive control system. in *Proc. of the 1992 American Control Conference*. pp. 2607–2612. qq.

MAMDANI E. H. AND S. ASSILIAN (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Machine Studies* Vol. 7, pp. 1–13.

MARTINEZ T. M, H. J. RITTER AND K. J. SCHULTEN (1989). 3d-neural-net for learning visuomotor-coordination of a robot arm. in *Int. Joint Conf. on Neural Networks (IJCNN 89)*. IEEE. pp. II, 351–356.

MEL B. (1991). A connectionist model may shed light on neural mechanisms for visually guided reaching. *Journal of Cognitive Neuroscience* Vol. 3, No. 3, pp. 274–292. MIT Press.

MENGER K. (1942). Statistical metrics. in *Proc. Nat. Acad. Sci. U.S.A. 28*. pp. 535–537.

MIYAKOSHI M. AND M. SHIMBO (1985). Solutions of composite fuzzy relational equations with triangular norms. *Fuzzy Sets and Systems* Vol. 16, pp. 53–63.

MIZUMOTO M. (1991). Min-max-gravity method versus product-sum-gravity method for fuzzy controllers. in *Fourth IFSA (Engineering)*. pp. 127–130.

MIZUMOTO M. AND H. J. ZIMMERMANN (1982). Comparison of fuzzy reasoning methods. *Fuzzy Sets and Systems* pp. 253–283.

NAGELL K., R.S. OLGUIN AND M. TOMASELLO (1993). Processes of social learning in the tool use of chimpanzees (pan troglodytes) and human children (homo sapiens). *Journal of Comparative Psychology* Vol. 107, No. 2, pp. 174–186.

NEGOITA M. GH., A. AGAPIE AND F. FAGARASAN (1994). Applications of genetic algorithms in solving fuzzy relational equations. in *Proc of the 2nd European Congress on Intelligent Techniques and Soft Computing, EUFIT'94*. pp. 1126–1129.

NOLFI S., D. FLOREANO, O. MIGLINO AND F. MONDADA (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. in *Proc of Artificial Life IV Conference, July 6-8, Cambridge, MA, USA*.

PALM R. (1992). Control of a redundant manipulator using fuzzy rules. *Fuzzy Sets and Systems* Vol. 45, pp. 279–298.

PEDRYCZ W. (1983*a*). Fuzzy relational equations with generalized connectives and their applications. *Fuzzy Sets and Systems* Vol. 10, pp. 185–201.

PEDRYCZ W. (1983*b*). Numerical and applicational aspects of fuzzy relational equations. *Fuzzy Sets and Systems* Vol. 11, pp. 1–18.

PEDRYCZ W. (1990*a*). Algorithms for solving fuzzy relational equations in a probabilistic setting. *Fuzzy Sets and Systems* Vol. 38, pp. 313–327.

PEDRYCZ W. (1990*b*). Relational structures in fuzzy sets and neurocomputation. in *Proc. of the Int. Conf. on Fuzzy Logic and neural Networks, Iizuka, Japan*. pp. 235–238.

PEDRYCZ W. (1991*a*). Fuzzy modelling: fundamentals, construction and evaluation. *Fuzzy Sets and Systems* pp. 1–15.

PEDRYCZ W. (1991*b*). Neurocomputations in relational systems. *IEEE Trans. Pattern Machine Intell* Vol. PAMI-13, No. 3, pp. 289–297.

PEDRYCZ W. (1991*c*). Processing in relational structures: Fuzzy relational equations. *Fuzzy Sets and Systems* Vol. 40, pp. 77–106.

PEDRYCZ W. (1992). Fuzzy neural neworks with reference neurons as pattern classifiers. *IEEE Transactions on Neural Networks* Vol. 3, No. 5, pp. 770–775.

PEDRYCZ W. (1994). Garel: A hybrid genetic learning in fuzzy relational equations. in *Proc of the IEEE Conference on Fuzzy Logic, World Congress on Compuational Intelligence, Orlando, Florida*. pp. 1354–1358.

PEDRYCZ W AND A. F. ROCHA (1993). Fuzzy-set based models of neurons and knowledge-based networks. *IEEE Transactions on Fuzzy Systems* Vol. 1, No. 4, pp. 254–266.

PEDRYCZ W., P.C.F. LAM AND A. F. ROCHA (1995). Distributed fuzzy system modelling. *IEEE Transactions on Systems, Man, and Cybernetics* Vol. 25, No. 5, pp. 769–780.

POMERLEAU D. A. (1993). *Neural network perception for mobile robot guidance*. Kluwer Academic Publishers.

REEKE G. N. AND O. SPORNS (1993). Behaviorally based modeling and computational approaches to neuroscience. *Annual Review of Neuroscience* Vol. 16, pp. 597–623.

RUMELHART D., G. HINTON AND J. L MCCLELLAND (1986). A general framework for distributed processing. in *Parallel Distributed Processing* (D. Rumelhart, J.L. McClelland and the PDP group, Eds.). MIT Press.

SAITO T. AND M. MUKAIDONO (1992). A learning algorithm for max-min network and its application to solve fuzzy relation equations. in *Proc. of the 2nd Int Conf on Fuzzy Logic and Neural Networks IIZUKA'92*. pp. 184–187.

SANCHEZ E. (1976). Resolution of composite fuzzy relation equations. *Information and Control* Vol. 30, pp. 38–48.

SANCHEZ E. (1993). Fuzzy genetic algorithms in soft computing environment. in *Proc Fifth IFSA Congress, Seoul, Korea*. pp. 44–50.

SANDERSON A. C AND L. E. WEISS (1986). Dynamic sensor-based control of robots with visual feedback. in *Proc IEEE Conf Robotics and Automation*.

SANGER T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation* Vol. 10, No. 3, pp. 323–333.

SCHNEIDER J. G. (1995). Robot skill learning through intelligent experimentation. PhD thesis. Computer Science, University of Rochester.

SEARLE J. R. (1980). Minds, brains and science. *Behaviour and Brain Science*.

SESTITO S. AND T. DILLON (1994). *Automated knowledge aquisition*. Prentice Hall, Australia.

SHERIDAN T. B. (1992). *Telerobotics, automation, and human supervisory control*. MIT Press.

SHIRAI Y. AND H. INOUE (1973). Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition* Vol. 5, pp. 99–108.

SMAGT P. VAN DER, F. GROEN AND B. KROSE (1993). Robot hand-eye coordination using neural networks. Technical Report TR CS-93-10. University of Amsterdam, Department of Computer Systems. Kruislaan 403.

STOICA A. (1994). Evolving creatures that can learn by imitation: apprentice behavior and its role in robot motor learning. in *Proc. of Int. Conf on Autonomous Robots and Arificial Life PERAC'94 - From Perception to Action, Switzerland*. pp. 440–443.

TAKAGI H. (1990). Fusion technology of fuzzy theory and neural networks - survey and future directions. in *Proc. of the Int. Conf. on Fuzzy Logic and neural Networks, Iizuka, Japan*. pp. 13–26.

TAKAGI H., N. SUZUKI, T. KODA AND Y. KOJIMA (1992). Neural networks designed on approximate reasoning architecture and their applications. *IEEE Trans on Neural networks* Vol. 3, No. 5, pp. 752–760.

TAMURA S., S. HIGUCHI AND K. TANAKA (1971). Pattern classification based on fuzzy relations. *IEEE Trans. on Systems, Man, and Cybernetics* Vol. SMC-1, pp. 61–66.

THIMM G., P. MOERLAND AND FIESLER (1995). The interchangeability of learning rate and gain in backpropagation networks. *submitted to Neural Computation*.

TOWELL G. AND J. SHAVLIK (1993). The extraction of refined rules from knowledge based neural networks. *Machine Learning* Vol. 131, pp. 71–101.

WALTER J. A. AND K. J. SCHULTEN (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks* Vol. 4, No. 1, pp. 86–95.

WANG D. AND M. A. ARBIB (1990). Complex temporal sequence learning based on short-term memory. *Proc. of the IEEE* Vol. 78, No. 9, pp. 1536–1542. qq.

WANG D. L. AND M. A. ARBIB (1993). Timing and chunking in processing temporal order. in *IEEE Trans. on Systems, Man, and Cybernetics*. IEEE. pp. 993–1009.

WILLIAMS R. J. (1986). The logic of activation functions. in *Parallel Distributed Processing* (D. Rumelhart, J.L. McClelland and the PDP group, Eds.). MIT Press.

Yager, R. R. and Filev, D., Eds. (1994). *Essentials of fuzzy modeling and control*. J. Wiley, New York.

Yager, R. R. and Zadeh, L. A., Eds. (1994). *Fuzzy sets, neural networks, and soft computing*. Van Nostrand Reinhold.

YAMAKAWA T., E. UCHINO, T. MIKI AND H. KUSANAGI (1992). A neo fuzzy neuron and its application to systems identification and prediction of system behavior. in *Proc. 2nd Int Conf on Fuzzy Logic and Neural Networks, IIZUKA'92*. pp. 477–483.

ZADEH L.A. (1965). Fuzzy sets. *Information and Control* Vol. 8, pp. 338–353.

ZADEH L. A. (1971). Similarity relations and fuzzy orderings. *Information Science* Vol. 3, pp. 177–200.

ZADEH L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics* Vol. SMC-3, No. 1, pp. 28–44.

ZIMMERMANN H. J. (1991). *Fuzzy Set Theory and Its Applications*. Kluwer-Nijhoff Publishing.

A listing of main Matlab and C programs used

```
// TINMAN
// written by Adrian Stoica, Victoria University of Technology, adrian@cabsav.vut.edu.au
//
// TINMAN learns eye-arm coordination by associating perceived images
// with commands given to own arm. It then imitates movements of a Master
// arm and learns motor patterns. The program allows the use of several
// learning techniques, for classic and fuzzy neurons.
//
//
//
// Robot camera goes in channel 1, Master camera in channel 2.
//
// This is in project with armp.c, rtx.c , fuzzylib.c, mc400_w.obj.
//
// // // Modification for 3D imitation using two identical robots
//      Modified getcomm("comrev") to getcomm3(comrevz)
//       5 and 6 for 3d (comrevz) sign is changed. 6 runs on Master, 5 runs on TINMAN


#include              <graphics.h>
#include   <math.h>
#include   <stdlib.h>
#include              <string.h>
#include   <stdio.h>
#include   <conio.h>
#include   <dos.h>
#include   "armp.h"
#include              "fuzzyhea.h"
#include              "mischea.h"
#include              "mc400_b.h"

char behavior;
char newchar;
unsigned char huge camera [256][256] ;     // from framegrabber
unsigned int fil1[200];                    // a line change filter system
unsigned int fil2[200];                    // to eliminate up-down shaking effect in images
unsigned char huge big_b[32768];           // image buffer
unsigned int  eye1 [12][16], eye2 [12][16]; // low resolution
unsigned int  cam1 [12][16], cam2 [12][16]; // used for display
int a,b,c,x,y,z;
int cot[145];                      // elbow trace
int umar[145];                     // shoulder trace
int sh, el, zaxis,  sbtot, eltot,ztot;
int counter1[192];                 // counters for MBMS (shoulder)
int counter2[192];                 //                  (elbow)
unsigned long age;                 // keeps track of how "old" TINMAN is (runs). Starts with 0 at birth.
unsigned long learnepoch;          // how many learning epochs
unsigned long timeawaken;          // how many cycles since wake-up
double frac;                       // constant required by MBMS
double WeightNN[3][192];           // memorize weights for classic neurons
double Ainputs[192];               // low resolution input from camera
double reference1[192];            // used by MBMS
double reference2[192];            //
double Ws[192];                    // weights for shoulder fuzzy neuron
double We[192];                    // weights for elbow fuzzy neuron
double Wz[192];                    // weights for z direction neuron
double activation[3];              // temporary output from neurons
double bias[3];                    // bias for classic neurons
double armcommand[175][3];         // shoulder and elbow commands for random flail
double repcommand[175][3];         // stores for playback
double huge eyemems[192][175];     // visual memories for off-line training
double errmot, err1, err2, err3, err4;   // errors
double err1abssum;
double elbowfactor;                // scale factors for motor commands
double shoulderfactor;             //
double zfactor;
double shpos, shposold, elpos, elposold, zpos, zposold, shposmod, elposmod, shposhum,shposrob, elposhum,
elposrob, zposhum, zposrob;
double ss, st;                     // logics for fuzzy neurons

FILE   *weight_file, *age_file, *image_file, *flail_file;
typedef struct
 {
  char red ;
  char green ;
  char blue ;
 } RGB ;

typedef RGB palette [256] ;
  palette       pal ;

  void set_RGB (p)

 char *p ;
 {
  int i ;

  outportb (0x3C8, 0) ;
  for (i = 0 ; i < 192 ; i++)
   outportb (0x3C9, *(p++)) ;
 }

int huge detect_vga256 ()
 {
  return (0) ; /* Return 0=lowest screen resolution ie 320x200) *
                /*       4 = highest ie 1024x768 not all VGA cards *
                 * support these resolutions  *
 }
```

```
void wakeup(void);                  // initializations and get memos from files
void go_sleep(void);                // save memos send arm to home position
void putwnnfuz(char *fname);            // save weights of fuzzy neurons
void putreference(char *fname);         // save reference for fuzzy neurons as used in MBMS
void putcounter(char *fname);           // save reference for fuzzy neurons as used in MBMS
void getreference(char *fname);         // load reference for fuzzy neurons as used in MBMS
void getcounter(char *fname);           // load reference for fuzzy neurons as used in MBMS
void ini(void);                             // graphical initializations
void getage(char *fname);               // gets age from file
void saveage(char *fname);              // saves age in file
void getepoch(char *fname);             // gets learning epoch from file
void saveepoch(char *fname);            // saves learning epoch in file
void getcomm(char *fname);              // gets commands for random flail
void getcomm3(char *fname);
void camselect(int camparam);
void camera2eye(int whicheye);
void getorder(void);
void dispeye(void);
void beep(void);
void getWNN(char *fname);
void getbias(char *fname);
void saveeye(char *fname);
void saveimg(char *fname);
void makeeye(void);
void makeeyehum(void);
void umarcot(char *fname);
void messages(double activation[3], int sh, int el, int zaxis ,double shpos, double elpos, double zpos, double
shposold, double elposold, double zposold);

void  main(void)
 {
  ini();
  wakeup();
  while (!kbhit()  && (behavior != 's'))   //    while no sleep order
   {
    int c192;

//  A set of choices follows. The first is LEARN, self-learning using classic
//  NN
    if (behavior == 'l') // ((l)earn
    {
                        //    Takes a set of positions from memory and moves to them

          activation[0] = armcommand[timeawaken][0];
          activation[1] = armcommand[timeawaken][1];
          shposold = shpos;
          elposold = elpos;
          shpos = (activation[0]-0.5)*shoulderfactor;
          elpos = (activation[1]-0.5)*elbowfactor;
          sh = shpos - shposold;
          el = elpos - elposold;

            MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();
beep();beep();beep();beep(); beep();beep();beep();beep();   getch();

 //get image and put it as vector in buffer images position timeawaken

            // Take image from camera one (robot)
             camselect(1);
             // Put it at low resolution in eye1.
             camera2eye(1);
             makeeye();     dispeye();
            for (c192 = 0; c192<192;c192++)
             eyemems[c192][timeawaken] =  Ainputs[c192];


}

        if (behavior == 't')    //test
{
//take command from buffer commands position timeawaken
          activation[0] = armcommand[timeawaken][0];
          activation[1] = armcommand[timeawaken][1];
          shposold = shpos;
          elposold = elpos;
          shpos = (activation[0]-0.5)*shoulderfactor;
          elpos = (activation[1]-0.5)*elbowfactor;

          sh = shpos - shposold;
          el = elpos - elposold;
//give command to arm
            gotoxy(1,14);
             printf(" activation[0] = %4.3f, activation[1] = %4.3f",activation[0],activation[1]);
            printf("\n shpos = %4.3f, elpos = %4.3f",shpos,elpos);
             printf("\n sh = %d, el = %d",sh,el);
            MOVEARM(sh,el,zaxis);   beep();beep();beep();beep(); getch();

 //get image and put it as vector in buffer images position timeawaken

            // Take image from camera one (robot)
             camselect(1);
             // Put it at low resolution in eye1.
             camera2eye(1);
             makeeye();     dispeye();    // take command from buffer commands position timeawaken

//give command to arm
```

```
//get image and input it to NN to obtain the model-command
 logsigNN(WeightNN,Ainputs,bias,activation);
        gotoxy(1,17);
        shposmod = (activation[0]-0.5)*shoulderfactor;
        elposmod = (activation[1]-0.5)*elbowfactor;
        printf("\n modelactivation[0] = %4.3f, modelactivation[1] = %4.3f",activation[0],activation[1]);
        printf(" \n shposmod = %4.3f, elposmod = %4.3f",shposmod,elposmod);
getch();

//display the given command and the model command
}


        if (behavior == 'f')   //follow
{

          // Take image from camera two (master)
           camselect(2);
          // Put it at low resolution in eye2.
           camera2eye(2);
           makeeyehum();     dispeye();    // take command from buffer commands position timeawaken

// calculate the command to place it there (from home position)
 //get image and input it to NN to obtain the model-command
        logsigNN(WeightNN,Ainputs,bias,activation);
        gotoxy(1,17);
        shposhum = (activation[0]-0.5)*shoulderfactor;
        elposhum = (activation[1]-0.5)*elbowfactor;
        printf(" humactivation[0] = %4.3f, humactivation[1] = %4.3f",activation[0],activation[1]);
        printf(" \n shposmod = %4.3f, elposmod = %4.3f",shposhum,elposhum);
          //          getch();    beep();beep();beep();beep();

//get image from robot
           camselect(1);
          // Put it at low resolution in eye1.
           camera2eye(1);
           makeeye();     dispeye();
//calculate the command to place it there (from home)

 logsigNN(WeightNN,Ainputs,bias,activation);
        gotoxy(1,19);
        shposrob = (activation[0]-0.5)*shoulderfactor;
        elposrob = (activation[1]-0.5)*elbowfactor;
        printf(" robactivation[0] = %4.3f, robactivation[1] = %4.3f",activation[0],activation[1]);
        printf(" \n shposrob = %4.3f, elposrob = %4.3f",shposrob,elposrob);

//calculate the difference between the commands
        sh = shposhum - shposrob;
        el = elposhum - elposrob;
        shtot +=sh;
        eltot +=el;
  // give this command to the robot
          //give command to arm
        printf("\n \n sh:hum-rob = %d, el:hum-rob = %d",sh,el);
        MOVEARM(sh,el,zaxis);


 }
 if (behavior == 'w') //watch human and learn
 {
//take command from buffer commands position timeawaken
        activation[0] = armcommand[timeawaken][0];
        activation[1] = armcommand[timeawaken][1];
        shposold = shpos;
        elposold = elpos;
        shpos = (activation[0]-0.5)*shoulderfactor;
        elpos = -(activation[1]-0.5)*elbowfactor;

        sh = shpos - shposold;
        el = elpos - elposold;
  //give command to arm   (activation[], sh, el, shpos, elpos, shposold, elposold)
          zaxis = 0;
          MOVEARM(sh,el,zaxis);  //  beep();beep();beep();beep();
//   beep();beep();beep();beep();
  beep();beep();beep();beep(); getch();

  //get image and put it as vector in buffer images position timeawaken

          // Take image from camera two (human)
           camselect(2);
          // Put it at low resolution in eye1.
           camera2eye(2);
           makeeyehum();     dispeye();
          for (c192 = 0; c192<192;c192++)
           eyemems[c192][timeawaken] = Ainputs[c192];

 }


        if (behavior == 'h')     //human test
{

          // Take image from camera two (human)
           camselect(2);
          // Put it at low resolution in eye1.
           camera2eye(1);
           makeeye();     dispeye();    // take command from buffer commands position timeawaken
```

```
//give command to arm

//get image and input it to NN to obtain the model-command
 logsigNN(WeightNN,Ainputs,bias,activation);
        gotoxy(1,17);
        shposmod = (activation[0]-0.5)*shoulderfactor;
        elposmod =- (activation[1]-0.5)*elbowfactor;
        printf("\n modelactivation[0] = %4.3f, modelactivation[1] = %4.3f",activation[0],activation[1]);
        printf(" \n shposmod = %4.3f, elposmod = %4.3f",shposmod,elposmod);
        getch();

//display the given command and the model command
}


        if (behavior == 'm')   //masterfollow   one camera
{
          // Take image from camera two (master)
          camselect(2);
          // Put it at low resolution in eye2.
          camera2eye(2);
          makeeyehum();     dispeye();    // take command from buffer commands position timeawaken
        gotoxy(1,16);
        printf(" activation[0] = %4.3f, activation[1] = %4.3f, activation[2] =
%4.3f",activation[0],activation[1],activation[2]);
        printf(" Weight[0][0] = %4.3f, Weight[1][0] = %4.3f, Weight[2][0] =
%4.3f",WeightNN[0][0],WeightNN[1][0],WeightNN[2][0]);
   // calculate the command to place it there (from home position)
    //get image and input it to NN to obtain the model-command
    printf("bias = %4.3f, %4.3f, %4.3f",bias[0],bias[1],bias[2]);
        logsigNN(WeightNN,Ainputs,bias,activation);

//      activation[2] = 0.5;  // in 2D movements keep the vertical motor control constant

        shposhum = (activation[0]-0.5)*shoulderfactor;
        elposhum = -(activation[1]-0.5)*elbowfactor;
//      zposhum  = (activation[2]-0.5)*zfactor;
        repcommand[timeawaken][0] = activation[0];    //this is for repetition!!!!
        repcommand[timeawaken][1] = activation[1];
//      repcommand[timeawaken][2] = activation[2];


        printf(" humactivation[0] = %4.3f, humactivation[1] = %4.3f, humactivation[2] =
%4.3f",activation[0],activation[1],activation[2]);
        printf(" \n shposmod = %4.3f, elposmod = %4.3f, zposmod = %4.3f",shposhum,elposhum,zposhum);
          //        getch();    beep();beep();beep();beep();


   //         robot
        shposrob = shtot*1.0;
        elposrob = eltot*1.0;
//      zposrob  = ztot*1.0;
        printf(" \n shposrob = %4.3f, elposrob = %4.3f, zposrob = %4.3f",shposrob,elposrob,zposrob);
   //calculate the difference between the commands
        sh = shposhum - shposrob;
        el = elposhum - elposrob;
//      zaxis = zposhum - zposrob;
 //     shtot +=sh;
  //    eltot +=el;
   // give this command to the robot
     //give command to arm
      printf("\n \n sh:hum-rob = %d, el hum-rob = %d, zaxis = %d",sh,el,zaxis);
            printf("\n timeawaken = %d",timeawaken);
      cot[timeawaken] = elposrob;
      umar[timeawaken] =shposrob;

// Watchdog
        if (( activation[0] >0.25) && (activation[0] <0.75)&&( activation[1] <0.51) && (activation[1] >0.27))
// &&( activation[2] >0.12) && (activation[2] <0.53)
                {
                zaxis = 0; MOVEARM(sh,el,zaxis);
                shtot +=sh;
                eltot +=el;
//              ztot  +=zaxis;
                 }
                  else  {beep();beep;}
//free!!!!          getch();

   }


        if (behavior == 'r') // repeat what you have buffered
  {
 //take command from buffer commands position timeawaken
        activation[0] = repcommand[timeawaken][0];
        activation[1] = repcommand[timeawaken][1];
        shposold = shpos;
        elposold = elpos;
        shpos = (activation[0]-0.5)*shoulderfactor;
        elpos = (activation[1]-0.5)*elbowfactor;

        sh = shpos - shposold;
        el = elpos - elposold;
   //give command to arm   (activation[], sh, el, shpos, elpos, shposold, elposold)

        zaxis = 0;   MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();
  beep();beep();beep();beep(); beep();beep();beep();beep();
```

```
}

        if (behavior == 'y')   // y save images of master and slave (masterfollow)   one camera
{
          // Take image from camera two (master)
           camselect(2);

           if (timeawaken == 1)

            {
                  for (x=z= 0 ; x < 256; x =x+2) //x++)
                   for (y = 0 ; y < 256 ; y =y+1)
                   big_b[z++] = camera[x][y];
                   saveimg("hum1.img");
                   }

if (timeawaken == 2)

            {
                  for (x=z= 0 ; x < 256; x =x+2) //x++)
                   for (y = 0 ; y < 256 ; y =y+1)
                   big_b[z++] = camera[x][y];
                   saveimg("hum2.img");
                   }


          // Put it at low resolution in eye2.
           camera2eye(2);
           makeeyehum();     dispeye();    // take command from buffer commands position timeawaken

// calculate the command to place it there (from home position)
 //get image and input it to NN to obtain the model-command
        logsigNN(WeightNN,Ainputs,bias,activation);
        gotoxy(1,17);
        shposhum = (activation[0]-0.5)*shoulderfactor;
        elposhum = -(activation[1]-0.5)*elbowfactor;
        printf(" humactivation[0] = %4.3f, humactivation[1] = %4.3f",activation[0],activation[1]);
        printf(" \n shposmod = %4.3f, elposmod = %4.3f",shposhum,elposhum);
          //          getch();    beep();beep();beep();beep();


//          robot

        shposrob = shtot*1.0;
        elposrob = eltot*1.0;
        printf(" \n shposrob = %4.3f, elposrob = %4.3f",shposrob,elposrob);

//calculate the difference between the commands
        sh = shposhum - shposrob;
        el = elposhum - elposrob;
        shtot +=sh;
        eltot +=el;
   // give this command to the robot
            //give command to arm
             printf("\n \n sh:hum-rob = %d, el:hum-rob = %d",sh,el);
                    printf("\n timeawaken = %d",timeawaken);
             cot[timeawaken] = elposrob;
             umar[timeawaken] =shposrob;

zaxis = 0;                MOVEARM(sh,el,zaxis); getch();

             camselect(1);

if (timeawaken == 1)

              {
                    for (x=z= 0 ; x < 256; x =x+2) //x++)
                     for (y = 0 ; y < 256 ; y =y+1)
                     big_b[z++] = camera[x][y];
                     saveimg("rob1.img");
                     }
if (timeawaken == 2)

              {
                    for (x=z= 0 ; x < 256; x =x+2) //x++)
                     for (y = 0 ; y < 256 ; y =y+1)
                     big_b[z++] = camera[x][y];
                     saveimg("rob2.img");
                     }
}

        if ( (behavior == '1') || (behavior == '2')) //watch human and learn
{
//take command from buffer commands position timeawaken
        activation[0] = armcommand[timeawaken][0];
        activation[1] = armcommand[timeawaken][1];
        shposold = shpos;
        elposold = elpos;
        shpos = -(activation[0]-0.5)*shoulderfactor;      //the minus sign is correlated to the use
        elpos = -(activation[1]-0.5)*elbowfactor;         // of a different command set, or
                  // movements to the other direction then initial (eg left as opposite to right)

        sh = shpos - shposold;
        el = elpos - elposold;
 //give command to arm  (activation[], sh, el, shpos, elpos, shposold, elposold)

          MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();
beep();beep();beep();beep();beep();beep();beep();beep();
```

```
//    for (i = 0; i < 192; i++)
//////////////////////    printf(" \n Ws[2] =  %4.3f, We[2] =  %4.3f",Ws[2],We[2]);


   //get image and put it as vector in buffer images position timeawaken

   if (behavior == '1')
            // Take image from camera one (robot)
                  camselect(1);

   else {
            // Take image from camera two (human)
            getch();    camselect(2);
           }
            // Put it at low resolution in eye1.
            camera2eye(2);

            makeeyehum();     dispeye();
           for (c192 = 0; c192<192;c192++)
            eyemems[c192][timeawaken] =  Ainputs[c192];

//  outmbms(0, st, Ws, Ainputs, activation);
//  printf("\n model activation[0] before learning =  %4.3f",activation[0]);

  mbms(Ws,counter1, reference1, Ainputs, activation[0], frac);

  outmbms(0, st, Ws, Ainputs, activation);
  printf("\n model activation[0] after learning =  %4.3f",activation[0]);

//  outmbms(1, st, We, Ainputs, activation);
//  printf("\n model activation[1] before learning =  %4.3f",activation[1]);

  mbms(We, counter2, reference2, Ainputs, activation[1], frac);

  outmbms(1, st, We, Ainputs, activation);
  printf("\n model activation[1] after learning =  %4.3f",activation[1]);

   }


         if (behavior == '3')   //masterfollow   one camera
  {
            // Take image from camera two (master)
             camselect(2);

            // Put it at low resolution in eye2.
            camera2eye(2);
            makeeyehum();     dispeye();    // take command from buffer commands position timeawaken

outmbms(0, st, Ws, Ainputs, activation);

outmbms(1, st, We, Ainputs, activation);

        gotoxy(1,17);
        shposhum = -(activation[0]-0.5)*shoulderfactor;
        elposhum = -(activation[1]-0.5)*elbowfactor;

        repcommand[timeawaken][0] = activation[0];     //this is for repetition!!!!
        repcommand[timeawaken][1] = activation[1];


        printf(" humactivation[0] =  %4.3f, humactivation[1] =  %4.3f",activation[0],activation[1]);
        printf(" \n shposmod =  %4.3f, elposmod =  %4.3f",shposhum,elposhum);

//          robot

        shposrob = shtot*1.0;
        elposrob = eltot*1.0;
        printf(" \n shposrob =  %4.3f, elposrob =  %4.3f",shposrob,elposrob);

//calculate the difference between the commands
        sh = shposhum - shposrob;
        el = elposhum - elposrob;

// give this command to the robot
          //give command to arm
          printf("\n \n sh:hum-rob =  %d, el:hum-rob =  %d",sh,el);
                printf("\n timeawaken =  %d",timeawaken);
          cot[timeawaken] = elposrob;
          umar[timeawaken] =shposrob;

         if( activation[0] >0.28) && (activation[0] <0.75)&&( activation[1] <0.50) && (activation[1] >0.28))
         watchdog
                  {MOVEARM(sh,el,zaxis);
                  shtot += sh;
                  eltot += el;
                   }
                    else  {beep();beep(),beep(); beep(),beep();beep(); beep();beep(),beep();}

}


        if (behavior == '4')  //repeat what you have buffered
{
//take command from buffer commands position timeawaken
        activation[0] = repcommand[timeawaken][0];
        activation[1] = repcommand[timeawaken][1];
```

```
        shposold = shpos;
        elposold = elpos;
        shpos = -(activation[0]-0.5)*shoulderfactor;
        elpos = -(activation[1]-0.5)*elbowfactor;

        sh = shpos - shposold;
        el = elpos - elposold;
  //give command to arm

            MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();
 beep();beep();beep();beep(); beep();beep();beep();beep();

 }


        if (behavior == '5') //Z
 {
//take command from buffer commands position timeawaken
        activation[0] = armcommand[timeawaken][0];
        activation[1] = armcommand[timeawaken][1];
        activation[2] = armcommand[timeawaken][2];
        shposold = shpos;
        elposold = elpos;
        zposold = zpos;
        shpos = (activation[0]-0.5)*shoulderfactor;
        elpos = (activation[1]-0.5)*elbowfactor;
        zpos = (activation[2]-0.5)*zfactor;
        sh = shpos - shposold;
        el = elpos - elposold;
        zaxis = zpos - zposold;
  //give command to arm  (activation[], sh, el, shpos, elpos, shposold, elposold)

            MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();
 beep();beep();beep();beep(); beep();beep();beep();beep();

//    for (i = 0; i < 192; i++)
////////////////////    printf("\n Ws[2] = %4.3f  We[2] = %4.3f",Ws[2],We[2]);


  //get image and put it as vector in buffer images position timeawaken

            // Take image from camera two (human)
            getch();    camselect(2);

             if (timeawaken == 1)

              {
                    for (x=z=0 ; x < 256; x=x+2) //x++)
                     for (y = 0 ; y < 256 ; y=y+1)
                                                  big_b[z++] = camera[x][y];
                                                  saveimg("hum1.img");
                                                  }

                if (timeawaken == 2)

                        {
                              for (x=z= 0 ; x < 256; x=x+2) //x++)
                               for (y = 0 ; y < 256 ; y=y+1)
                               big_b[z++] = camera[x][y];
                               saveimg("hum2.img");
                               }

                          // Put it at low resolution in eye1.
                          camera2eye(2);

                          makeeyehum();     dispeye();
                          for (e192 = 0; e192<192;e192++)
                          eyemems[e192][timeawaken] = Ainputs[e192];

//  outmbms(0, st, Ws, Ainputs, activation);
//  printf("\n model activation[0] before learning = %4.3f",activation[0]);

//**//  mbms(Ws,counter1, reference1, Ainputs, activation[0], frac);

//**//  outmbms(0, st, Ws, Ainputs, activation);
//**//  printf("\n model activation[0] after learning = %4.3f",activation[0]);

//  outmbms(1, st, We, Ainputs, activation);
//  printf("\n model activation[1] before learning = %4.3f",activation[1]);

//**//  mbms(We, counter2, reference2, Ainputs, activation[1], frac);

//**//  outmbms(1, st, We, Ainputs, activation);
//**//  printf("\n model activation[1] after learning = %4.3f",activation[1]);

  }


        if (behavior == '6') // This is a "copy" program running on a second robot
 {
        activation[0] = armcommand[timeawaken][0];
        activation[1] = armcommand[timeawaken][1];
        activation[2] = armcommand[timeawaken][2];
        shposold = shpos;
        elposold = elpos;
        zposold = zpos;
        shpos = (activation[0]-0.5)*shoulderfactor;
        elpos = (activation[1]-0.5)*elbowfactor;
```

```
        zpos = (activation[2]-0.5)*zfactor;
        sh = shpos - shposold;
        el = elpos - elposold;
        zaxis = zpos - zposold;  beep();beep();beep();beep();  beep();beep();beep();beep();
        MOVEARM(sh,el,zaxis);   beep();beep();beep();beep();   beep();beep();beep();beep();
  delay(1000);        getch();

 }


 timeawaken++;
 if (timeawaken == 97)  // here 97 training examples or iteration steps
        {
        if ((behavior == '1') || (behavior == '2')|| (behavior == '5'))
        {
          learnepoch ++;
           for (x = 0 ; x < 192 ; x++)
                   {
                   if (counter1[x] <= learnepoch)
                    { reference1[x] = 0;
                      Ws[x] = 0;
                    }
                   if (counter2[x] <= learnepoch)
                    { reference2[x] = 0;
                      We[x] = 0;
                    }

                   }
         }
        behavior = 's';
        }

 }
 go_steep();
}


void wakeup(void)
{

  char        memoryfile [40];
  int j, i;
  err1abssum = 0.0;
//  frac = 0.04;  printf("frac = "); scanf("%f", &frac); printf("frac = %4.2f",frac);
    frac = 0.06;

printf("  frac : ");
scanf("%lf", &frac);   printf(" that is %of \n",frac);
 activation[0] = 0.0;   activation[1] = 0.0; activation[2] = 0.0;
 ss = 0.01; st =1.001;
 sh = 0; el = 0;  zaxis = 0;    shtot = 0; eltot = 0; ztot = 0;
 shpos = 0.0; shposold = 0.0;
 elpos = 0.0; elposold = 0.0;
 zpos = 0.0;  zposold = 0.0;
timeawaken = 0;
elbowfactor = 4800.0; //2400.0
shoulderfactor = 4800.0;      4800.0
zfactor = 200.0*10; //10;  for RT100 and 5 for  RTX

  getorder();


 clrscr();
          if (behavior == 'j')
 {
  getWNN("w3d.dat");
 getbias("b3d.dat");
 behavior = 'm';
   }
   else
{
getWNN("w3d.dat");
 getbias("b3d.dat");
 }

 getWNN("wgol.dat");
 getbias("bgol.dat");

          if( (behavior == '1') || (behavior == '2') ||(behavior == '3') || (behavior == '4') || (behavior == '5') ||
(behavior == '6') )
           { getcomm3("comrevz.dat");   getWNN("wmufuz.dat"); // getcomm("comrev.dat");
             printf("age %d", age);
   if (age > 0)   { printf("age %d", age); getreference("referenc.dat"); getcounter("counter.dat");
getepoch("epoch.dat");}
           }
          else  getcomm("comrev.dat"); //getcomm("comfileh.dat");


 if (age == 0)
 {
   for (j = 0; j<192; j++)
   { learnepoch = 0;
          reference1[j] = 0; reference2[j] = 0;
          Ws[j] = 0; We[j] = 0;
          counter1[j] = 0; counter2[j] = 0;
   }
 }
```

```
// blink your eyes...
 for (j = 1; j<=16; j++)
  {for (i = 1; i<=12; i++)
    {  eye2[i-1][j-1] = 0;
           eye1[i-1][j-1] = 0;
    }
   }

 for (j = 1; j<=256; j++)
  {for (i = 1; i<=256; i++)
  camera[i-1][j-1] = 'A';
  }


          INIMOT();       //initialize  motors

  for (j = 0 ; j < 480; j++)
  {
   for (i = 0  ; i < 640 ; i++)
          putpixel (i, j, 0) ;
          }

}

void go_sleep(void)
// Organize and save memories.
// Have arm placed in initial position SET_POSITION/ GO_POSITION (in RTX lib)
{

age = age+1; //baby, you are one cycle older!

GOSLEEP();        //move arm back in home position
saveage("age.dat");
saveepoch("epoch.dat");
putwnnfuz("wnnfuz.dat");
putreference("referenc.dat");
putcounter("counter.dat");
saveeye("eyememo.dat");
umarcot("umarcot.dat");
closegraph();

}


void camselect(int camparam)
{
  int i; long sumfil;
   unsigned buff_ofs, buff_seg ;
   sumfil = 0;

   asm MOV AX,0x318
   MC_SETPORT () ;
   if (camparam == 1)  MC_SETCHAN1 () ;
   else MC_SETCHAN2 () ;

   MC_SETRESHIGH ();

   buff_ofs = FP_OFF (camera) ;
   buff_seg = FP_SEG (camera) ;

   asm MOV DI,buff_ofs
    asm MOV AX,buff_seg
//  get first frame
          MC_GRAB8A () ;
                beep();

//////////////////////////
          asm MOV DI,buff_ofs
          asm MOV AX,buff_seg

// get first frame
          MC_GRAB8A ();
          beep();


 for (i = 1; i<200; i++) fil1[i] = camera[i][128];
 //////////////////////////
          asm MOV DI,buff_ofs
          asm MOV AX,buff_seg

// get first frame
          MC_GRAB8A ();
          beep();

 for (i = 1; i<200; i++) fil2[i] = camera[i][128];

 for (i = 1; i<200; i++) sumfil +=  abs(fil2[i]-fil1[i]);
printf("sumfil %ld",sumfil);
 if (sumfil > 280)
 {
          asm MOV DI,buff_ofs
          asm MOV AX,buff_seg
          MC_GRAB8A ();
          beep();
 }
```

```
}


void ini(void)
{
  //Graphic initialization
                        int driver, mode, error_code;
                        int gdriver = DETECT, gmode, errorcode;
                        int i, x, y, color, maxx, maxy, maxcolor, seed;
                        initgraph(&gdriver, &gmode, "");
                                /* read result of initialization */
                        errorcode = graphresult();
                                /* an error occurred */
                        if (errorcode != grOk)
                                 {  printf("Graphics error: %s\n", grapherrormsg(errorcode));
                        printf("Press any key to halt:");
                                 getch();
                                /* terminate with an error code */
                        exit(1);
                                 }

}


void camera2eye(int whicheye)
{
 int i,j,a,b;  double sumarray;

 for (j = 1; j<=16; j++)
   {for (i = 1; i<=12; i++)
     {sumarray = 0.0;
             for (a = 0;a<=15; a++)
                {for (b =0; b<=15; b++)
                 sumarray +=  camera[(i-1)*16+a][(j-1)*16+b];          //average
                 }
              if (whicheye == 1) eye1[i-1][j-1] = min(250, sumarray/256.0 + 50); //soft light
              else  eye2[i-1][j-1] = min(250, sumarray/256.0 + 50);  //software light
     }
    }

 }

 void makeeye(void)
  {
  int i,j, k;
  k = 1;
  for (i = 1; i<=12; i++)
    {for (j = 1; j<=16; j++)
      {Ainputs[k-1] =1.0 - (eye1[i-1][j-1])/255.0;  dark values are informative andd get higher real value
       k = k+1;
       }
      }
  }

void makeeyehum(void)
 {
 int i,j, k;
 k = 1;
 for (i = 1; i<=12; i++)
   {for (j = 1; j<=16; j++)
     {Ainputs[k-1] =1.0 - (eye2[i-1][j-1])/255.0;  //dark values are informative andd get higher real value
      k = k+1;
      }
     }
 }


 void saveage(char *fname)
// save memories to file
{
 // get an image from a file
   if ((age_file = fopen (fname, "w")) == NULL)
    {
             fprintf (stderr, "Could not find the required image file.\n") ;
             exit (1) ;
    }
             fprintf (age_file, "%d", age);
              fclose (age_file) ;

}

 void saveepoch(char *fname)
 {
 // get an image from a file
   if ((age_file = fopen (fname, "w")) == NULL)
    {
             fprintf (stderr, "Could not find the required image file.\n") ;
             exit (1) ;
    }
             fprintf (age_file, "%d", learnepoch);
              fclose (age_file) ;

}



 void getage(char *fname)
// save memories to file
```

```
{

  if ((age_file = fopen (fname, "r")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    }
 fscanf (age_file,"%d", &age) ;
            fclose (age_file) ;
}

 void getepoch(char *fname)
 {

  if ((age_file = fopen (fname, "r")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    }
 fscanf (age_file,"%d", &learnepoch) ;
            fclose (age_file) ;
}

void  getcomm(char *fname)
// get commands from file
{
   int i;
   float a,b;
  if ((flail_file = fopen (fname, "r")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <175; i++)
    {   fscanf (flail_file,"%f %f\n",  &a, &b) ;
             armcommand[i][0] = a*1.0;
             armcommand[i][1] = b*1.0;
     }
            fclose (flail_file) ;

}
 void  getcomm3(char *fname)
// get commands from file
{
    int i;
    float a,b,c;
   if ((flail_file = fopen (fname, "r")) == NULL)
     {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <175; i++)
    {   fscanf (flail_file,"%f %f %f\n",  &a, &b, &c) ;
             armcommand[i][0] = a*1.0;
             armcommand[i][1] = b*1.0;
             armcommand[i][2] = c*1.0;
     }
            fclose (flail_file) ;

}


   void getWNN(char *fname)
// get  weights for clasic nn from file   double WeightNN[2][192];
{
   int i;
   float a,b,c;
  if ((flail_file = fopen (fname, "r")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    }

    for (i = 0; i <192; i++)
    {   fscanf (flail_file,"%f %f\n",  &a, &b) ; // fscanf (flail_file,"%f %f %f\n",  &a, &b, &c) ;
             Ws[i] = a*1.0 ;
             We[i] = b*1.0 ;
             Wz[i] = c*1.0 ;
             WeightNN[0][i] = a*1.0 ;
             WeightNN[1][i] = b*1.0 ;
             WeightNN[2][i] = 0 ;  // WeightNN[2][i] = c*1.0 ;
     }
            fclose (flail_file) ;


}


   void getbias(char *fname)
{
   int i;
   float a,b,c;
  if ((flail_file = fopen (fname, "r")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    }

            fscanf (flail_file,"%f %f %f\n",  &a, &b, &c) ;
```

```
            bias[0] = a*1.0 ;
            bias[1] = b*1.0 ;
            bias[2] = c*1.0 ;
            fclose (flail_file) ;

}


   void getreference(char *fname)
// get  weights for clasic nn from file   double WeightNN[2][192];
{
   int i;
   float a,b;
  if ((flail_file = fopen (fname, "r")) == NULL)
   {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
   }

   for (i = 0; i <192; i++)
   {  fscanf (flail_file,"%f %f\n",  &a, &b) ;
            reference1[i] = a*1.0 ;
            reference2[i] = b*1.0 ;

    }
            fclose (flail_file) ;


}

   void getcounter(char *fname)
// get  weights for clasic nn from file   double WeightNN[2][192];
{
   int i;
   int  a,b;
  if ((flail_file = fopen (fname, "r")) == NULL)
   {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
   }

   for (i = 0; i <192; i++)
   {  fscanf (flail_file,"%d %d\n",  &a, &b) ;
            counter1[i] = a;
            counter2[i] = b;

    }
            fclose (flail_file) ;
}


void getorder(void)
{


 printf("\n G'day MASTER, \n            I'm TINMAN, \n \n what do you want me to do? \n (Learn, Testrob,
Follow2cam, Watchhum, Humtest, \n (M)asterfollcam, (J)umper, Ysave6Masterfollow , (R)epeat, (S)leep, \n
FuzLearnFromRobot(1), FuzLearnFromHuman(2), FuzFollow(3), FuzSkill(4), ZTEST (5), Master robot (6)  ");
 behavior = getch();
   printf(" Do you want to take age from file or this is my birth (y or n) ");
 if (getch() == 'y')
                  {        getage("age.dat");
                  }
  else          age = 0;

}


void dispeye()
{
   int x,y;
  // put eyes image on the screen

    for (x = 0 ; x < 16; x++)
    {
     for (y = 0 ; y < 12 ; y++)

    {if (eye1 [y][x] > 240) cam1[y][x] = 0;   180
              else {
              // if (eye1 [y][x] > 70) cam1[y][x] = 15; // 100
             // else
              cam1[y][x] = 15;
              }
              putpixel (256 + 16* x, 16*y, cam1[y][x]); // camera [y][x]) ;

              }
   }

              for (x = 0 ; x < 16; x++)
    {
     for (y = 0 ; y < 12 ; y++)
              {if (eye2 [y][x] > 240) cam2[y][x] = 0;
              else {
               // if (eye2 [y][x] > 70) cam2[y][x]    15;
              // else
```

```
            cam2[y][x] = 15;
               }
            putpixel (16 * x, 16 *y, cam2 [y][x]) ;     // first in time is to the left...

           }
   }
}


void beep ()
{  sound(4000);
   delay(2000);     delay(200000);
    sound(5000);
   delay(2000);     delay(200000); sound(6000);
   delay(2000);     delay(200000);
   nosound();
}


void saveeye(char *fname)

{
    int i,j;
    float a;
   if ((flail_file = fopen (fname, "wt")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <175; i++)
            for (j =0; j<192;j++)
    {
            a= eyemems[j][i] ;

            fprintf (flail_file,"%7.5f\n", a) ;

     }
            fclose (flail_file) ;
}

 void umarcot(char *fname)

{
    int i, a,b;
   if ((flail_file = fopen (fname, "wt")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    }
    for (i = 0; i <145; i++)
    {
            a= umar[i] ;
            b = cot[i];
            fprintf (flail_file,"%d %d \n", a,b) ;

     }
            fclose (flail_file) ;
}



void saveimg(char *fname)

{

              if(image_file=fopen(fname,"wb"))
             {
              fwrite(big_b,32768,1,image_file); //16384  65535
              fclose(image_file);
             }
    else
            {fprintf(stderr,"Can't open file for writing \n");exit(1);}


}



   void putwnnfuz(char *fname)
{
    int i;
    float a,b;
   if ((flail_file = fopen (fname, "w")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <192; i++)
    {
            a = Ws[i];
            b = We[i];
            fprintf (flail_file,"%7.5f %7.5f\n",  a, b) ;
     }
            fclose (flail_file) ;

}
```

```
   void putcounter(char *fname)
{
   int i;
   int a,b;
  if ((flail_file = fopen (fname, "w")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <192; i++)
    {
            a = counter1[i];
             b = counter2[i];
            fprintf (flail_file,"%d %d\n",  a, b) ;
    }
             fclose (flail_file) ;

}


   void putreference(char *fname)
{
   int i;
   float a,b;
  if ((flail_file = fopen (fname, "w")) == NULL)
    {
            fprintf (stderr, "Could not find the required file.\n") ;
            exit (1) ;
    } for (i = 0; i <192; i++)
    {
            a = reference1[i];
             b = reference2[i];
            fprintf (flail_file,"%7.5f %7.5f\n",  a, b) ;
     }
             fclose (flail_file) ;

}


 void messages (double activation[3],int sh, int el,int zaxis, double shpos, double elpos, double zpos, double
shposold, double elposold, double zposold)
 {
                gotoxy(1,18);
                 printf(" activation[0] =  %4.3f, activation[1] =  %4.3f, activation[2] =
%4.3f",activation[0],activation[1], activation[2]),
                printf(" shpos =  %4.3f, elpos =  %4.3f, zpos = %4.3f",shpos,elpos,zpos);
                 printf(" shposold =  %4.3f, elposold =  %4.3f, zposold = %4.3f",shposold,elposold,zposold);
                 printf(" sh =  %d, el =  %d, zaxis = %d",sh,el,zaxis);

  }
```

```
// THIS FILE CONTAINS DEFINITIONS OF FUNCTIONS USED TO IMPLEMENT
// FUZZY SYSTEMS BASED ON S-T-COMPOSITION

// Written by Adrian Stoica, 1995
// Electrical and Electronic Engineering, Victoria University of Technology, Melbourne,
// adrian@cabsav.vut.edu.au


// Contents:
 /*  MINI
    MAXI
    TNORM
    SNORM
    FCSSST
    FC
    WEIGHTHEB
    FRENODESS
    LOGSIGNN
    LOGSIG23x32
    MBMS
    OUTMBMS
  */


#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "mc400_b.h"
#include    "ammp.h"
#include       "fuzzyhea.h"
#include       "mischea.h"

// ****Function MINI returns the min of two variables, MIN(X,Y)

 double mini(double value1, double value2)
 {
   if (value1 < value2) return value1;
   else return value2;
  }


// ****Function MAXI returns the maximum of two variables, MAX(X,Y)

 double maxi(double value1, double value2)
 {

  if (value1 > value2) return value1;
   else return value2;
 }


// ****Function TNORM returns the fundamental TNORM of two variables for a given
//              logic s, TNORM(s,X,Y)

  double tnorm (double s, double x, double y)
{
   double t;
if (s == 0)
 t = mini(x,y);
else if (s != 1)
 t = (1/log(s))*log(1+(pow(s,x) - 1)*(pow(s,y) - 1) (s-1));
else
 t = x*y;

 return t;

}


// ****Function SNORM returns the fundamental SNORM of two variables for a given
//              logic s, SNORM(s,X,Y)


  double snorm (double s, double x, double y)
{
   double tco;
if (s == 0)
 tco = maxi(x,y);
else if (s != 1)
 tco = 1 - (1/log(s))*log(1+(pow(s,1-x) - 1)*(pow(s,1-y) - 1) (s-1));
else
 tco = x + y - x*y;

 return tco;

}



// ****Function FC performs S-T composition between two matrices for a given
//              logic ss for snorm, st for tnorm, Y = FC(ss,st,X,W)


void fc (double ss, double st, double X[], double W [][4], double YYY[], int m, int n)  // function
{
```

```
  int i,j,k;
  int p = 1;
for (k = 1; k <= p; k++)     // for all examples
 { for (j = 1; j <= n; j++)  // for all output nodes
   {
    YYY[j-1] = 0;
    for (i = 1; i <= m; i++)
    YYY[j-1] = snorm(ss, YYY[j-1], tnorm(st, X[i-1], W[i-1][j-1]));
   }
 }
return;
}


void weightheb (double ss, double st, double X[][1], double W [][4], double YYY[][4], int p, int m, int n)
//function
{
  int i,j,k;
for (k = 1; k <= p; k++)     // for all examples
 { for (j = 1; j <= n; j++)  // for all output nodes
   {
    YYY[k-1][j-1] = 0;
    for (i = 1; i <= m; i++)
    YYY[k-1][j-1] = snorm(ss, YYY[k-1][j-1], tnorm(st, X[k-1][i-1], W[i-1][j-1]));
   }
 }
return;
}

// ****Function FRENODESS(ss,st, X, bb, inc, ww,m,n) does for every
//                  node, example by example gradient descent learning


void frenodess(double ss, double st, double X[], double bb[], double inc, double W[][4], int m, int n)

{
int i,j,k,g;
double T;
double E;
double DYDT;
double DTDW;
double DYDW;
double prod, tg;
double Y[4] = {0};

fc(ss,st,X,W,Y, m, n);

 for (j = 1; j <= n; j++)  // for all output nodes
   {
   prod = 1;                    calc product
   for (g = 1; g<=m; g++)
          {
           tg = tnorm(st, X[g-1], W[g-1][j-1]);
           prod = prod*(pow(ss, 1-tg) - 1);
          }
   E = Y[j-1] - bb[j-1];        // calc error for j node

   for (i = 1; i <= m; i++)
          {
            T = tnorm(st, X[i-1], W[i-1][j-1]);
            DYDT = pow(ss,(1-T))*(prod (pow(ss,(1-T)) -1))/(pow(ss-1, m-1) + prod);
            DTDW = (pow(st,W[i-1][j-1]))*(pow(st,X[i-1]) - 1)/((st-1) + (pow(st,X[i-1]) - 1)*(pow(st,W[i-1][j-
1]) -1));
            DYDW = DYDT*DTDW;
            W[i-1][j-1] = W[i-1][j-1] - inc*E*DYDW;
            if (W[i-1][j-1] < 0) W[i-1][j-1] = 0.001;
            if (W[i-1][j-1] > 1) W[i-1][j-1] = 0.999;
          }
   }
return;
}

// ****Function LOGSIGNN(W,At,b,a) calculates the output of a neural layer
// weights W inputs A, bias b and activations/outputs a.


 void logsigNN(double W[][192],double At[],double b[],double a[])
{
int i,j;
int nrin = 192;
double sum[3];

for (j = 0; j< 3; j++)
 {
  sum[j] = 0;

  for (i = 0; i< nrin; i++)
    sum[j] = sum[j] + W[j][i]*At[i];
  sum[j] = sum[j] + b[j];
  a[j] = 1.0/ (1.0 + exp(-sum[j]));
   }
}



 void logsig24x32(double W[][768],double At[],double b[],double a[])

{
```

```
int i,j;
int nrin = 768;
double sum[2];

for (j = 0; j< 2; j++)
 {
 sum[j] = 0;

 for (i = 0; i< nrin; i++)
  sum[j] = sum[j] + W[j][i]*At[i];


 sum[j] = sum[j]+b[j];

 a[j] = 1.0/ (1.0 +exp(-sum[j]));

 }

}

void fcssst (double ss, double st, double X[][192], double W [][2], double YYY[][2], int m, int n, int p)  //function
{
  int i,j,k;

for (k = 1; k <= p; k++)     // for all examples
 { for (j = 1; j <= n; j++)  // for all output nodes
   {
    YYY[k-1][j-1] = 0;
    for (i = 1; i <= m; i++)
    YYY[k-1][j-1] = snorm(ss, YYY[k-1][j-1], tnorm(st, X[k-1][i-1], W[i-1][j-1]));
   }
  }
return;
}

    void mbms(double W[],int counter[],double reference[],double x[], double y, double frac)
{
//Calculates weights for max-t neurons (maximize if bigger minimize if smaller)
int nrin = 192;
int i;
double a,b;

  for (i = 0; i< nrin; i++)
   {       a = x[i]; b = reference[i];
   if ((a -b) >= frac)
    {
     reference[i] = x[i];
     counter[i] = counter[i] + 1;
     W[i] = y;
    }
    if (fabs(a-b) < frac)   W[i] = mini(W[i],y);
   if ((a-b) < -frac)       counter[i] = counter[i] + 1;
   }
}



    void outmbms(int care, double st, double W[], double e2[], double rez[])
{

//Calculates output of a max-t neuron
double vali;
double inter[192];
int i;
vali = 1.0-1.0;
 for (i = 0; i < 192; i++)
  {
   inter[i] =  tnorm(st, e2[i], W[i]);
   if (inter[i] > vali)
     {
      vali = inter[i];
      rez[care] = W[i];
     }
  }

}
```

A listing of main Matlab and C programs used

MATLAB functions for fuzzy neural processing

Written by Adrian Stoica, Victoria University of Technology, Australia, a.stoica@ieee.org

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  T-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function t = tnorm(s,x,y)
% TNORM(s,x,y) returns the fundamental t-norm of x and y, where s is a parameter
% x and y can be matrices

if s == 0
  t = min(x,y);
elseif s == 1
  t = x.*y;
else
  t = (1/log(s))*log(1+(s.^x - 1).*(s.^y - 1)/(s-1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   S-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function tco = snorm(s,x,y)
% SNORM(s,x,y) returns the fundamental t-conorm of x and y, where s is a parameter

if x <= 0
  x = 0.000001; %avoid some errors by MATLAB/Unix which for x = 0 y = 0 s = 0.1 returns a negative (very
small) result!
end
if y <= 0
  y = 0.000001;
end

if s == 0
   tco = max(x,y);
elseif s == 1
   tco = x+y - x.*y;
else
   tco = 1 - (1/log(s))*log(1+(s.^(1-x) - 1).*(s.^(1-y) - 1)/(s-1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   VECTORIAL  S-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function tcov = snormvec(s,X)
% SNORMVEC(s,X) returns the fundamental t-conorm of elements in vector X,  where s is a parameter

[pp,p] = size(X);
tcov = 0;
for i = 1:p
          tcov = snorm(s, tcov, X(i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   S-T COMPOSITION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function c = fcssst(ss,st,a,b)
% FCSSST(ss,st,a,b) returns S-T composition of fuzzy relations a and b, t-norm being the fundamental t-norm
(Frank), ss and st are parameters for S and T.

sa = size(a);  %size of matrices rows and columns
sb = size(b);

m = sa(1);    % nr of lines of first matrix
n = sb(2);    % nr of columns of second matrix
c = zeros(m,n);  %prealocation of space for increasing the speed
for i=1:m
  for j=1:n
          c(i,j)=snormvec(ss,(tnorm(st,a(i,:), b(:,j)')));
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   FUNDAMENTAL FUZZY NEURON
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function c = fn(ip)
% FN(ss,st,a,b) returns S-T composition of fuzzy relations a and b, t-norm being the fundamental t-norm
(Frank).This is the output of a fundamental fuzzy neuron.

%ip is a vector of four components: ss, st, Inputs, Weights.
%for example IP = [0.3  0.5 0.1 0.4 0.5 0.5 0.6 0.7]' has 3 inputs and 3 coresponding weights

[np2,nppp] =size(ip);
np = np2-2;
npj = np/2+2;
ss=ip(1);
st=ip(2);
a=ip(3:npj)';
b=ip(npj+1:np2);

sa = size(a);  %size of matrices rows and columns
sb = size(b);

m = sa(1);    % nr of lines of first matrix
```

```
n = sb(2);    % nr of columns of second matrix
c = zeros(m,n);  %prealocation of space for increasing the speed
for i=1:m
  for j=1:n
          c(i,j)=snormvec(ss,(tnorm(st,a(i,:), b(:,j)')));
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   GRADIENT DESCENT LEARNING FOR FUNDAMENTAL FUZZY NEURAL NETWORKS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ww = frenodess(ss,st, X, bb, times, inc, ww)
%FRENODESS(ss,st, X, bb, times, inc, ww) finds approximate solution for S-T FRE, performing example by
example learning, for each output node

%start initialisation
[p,m] = size(X);
[pp,n] = size(bb);

if nargin < 6
    inc = 0.9;
          ww = fcssst(ss,st,X',bb);
end

if nargin == 6
          ww = fcssst(ss,st,X',bb);
end
%end initialisation

e = zeros(n,times);

for j = 1:n       %for all nodes
  w = ww(:,j);
  ba = bb(:,j);
es = 0;

for par = 1:times
  for k = 1:p
    x = X(k,:);
    b = ba(k);
    y = snormvec(ss, tnorm(st,x, w'));

    eroare = y - b;
    ersqr = eroare^2;

      prod = 1;
      for g = 1:m
        t(g) = tnorm(st, x(g), w(g));
      prod = prod*(ss^(1-t(g)) - 1);
     end

    for i = 1:m
     t(i) = tnorm(st, x(i), w(i));


     DYDT(i) = (ss^(1-t(i)))*(prod (ss^(1-t(i)) -1)) ((ss-1)^(m-1) + prod);
     DTDW(i) = (st^w(i))*(st^x(i) - 1)/( (st-1) + (st^x(i) - 1)*(st^w(i) -1));
     DYDW = DYDT.*DTDW;
     w(i) = w(i) - inc*eroare*DYDW(i);
     if w(i) < 0
      w(i) = 0.001;
     end
     if w(i) > 1
      w(i) = 0.999;
     end
    end
   es = es + ersqr; %sum of eroareors for all ks
   end
e(j,par) = es; es = 0;
ww(:,j) = w;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
VECTORIAL T-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function tnv = tnormvec(s,X)
% TNORMVEC(s,X) returns the fundamental t-norm of elements in vector X,  where s is a parameter

pp = size(X);
p = pp(2);
tnv = 1;
for i = 1:p
          tnv = tnorm(s, tnv, X(i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   ARRAY T-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function tnv = tnormarray(s,x,y)
% TNORMARRAY(s,x,y) returns an array pointwise tnorm of vector x and y

[p,pp] = size(x);
[q,qq] = size(y);
```

```
for i = 1:qq
        tnv(i,:) = tnorm(s, y(i), x);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   CUBE T-NORM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function tncv = tnormcube(s,x,y,z)
% TNORMARRAY(s,x,y) returns a vector scan of a 3D array pointwise tnorm of vector x, y, z

[p,pp] = size(x);
[q,qq] = size(y);
[g,gg] = size(z);

for i = 1:qq
        tnv(i,:) = tnorm(s, y(i), x);
end
sos = ar2vec(tnv);
for i = 1:gg
    tnc(i,:) = tnorm(s, z(i), sos);
end

tncv = ar2vec(tnc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   ALPHA-COMPOSITION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sol = solfre(A,B)
%This function implements alpha-composition to find the greatest solution of MAX-MIN FRE
% MAX(MIN(A,sol)) = B.

Aprim = A';
[rezin,nrex] = size(Aprim);
[nrex,nodes] = size(B);

sol = zeros(rezin,nodes);
col = zeros(1,nrex);

 for k = 1:rezin

   for node = 1:nodes

     x = Aprim(k,:);
     y = B(:,node);

     for i = 1:nrex

       if x(i) > y(i)
         col(i) = y(i);
       else
         col(i) = 1;
       end
     end  % for i nrex

   sol(k,node) = min(col);

   end  % for node

 end  % for k

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  ALPHA
M -COMPOSITION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sol = solfremod(A,B)
%This function uses minimal alpha-composition to find a solution for MAX-MIN FRE

Aprim = A';
[rezin,nrex] = size(Aprim);
[nrex,nodes] = size(B);

 sol = zeros(rezin,nodes);
 col = zeros(1,nrex);

 for k = 1:rezin
   for node = 1:nodes

     x = Aprim(k,:);
     y = B(:,node);


     for i = 1:nrex

       if x(i) > y(i)
         col(i) = y(i);
       else
         col(i) =1;
       end
     end  % for i nrex

 if min(col) == 1
  sol(k,node) = max(y);
 else
  sol(k,node) = min(col);
 end
   end  % for node
```

```
end   % for k

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   ALPHA 0 -COMPOSITION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sol = solfre0(A,B)
%This function uses alpha 0 -composition to find a solution for MAX-MIN FRE

Aprim = A';
[rezin,nrex] = size(Aprim);
[nrex,nodes] = size(B);

 sol = zeros(rezin,nodes);

col = zeros(1,nrex);

 for k = 1:rezin
   for node = 1:nodes

     x = Aprim(k,:);
     y = B(:,node);


    for i = 1:nrex

      if x(i) > y(i)
         col(i) = y(i);
       else
         col(i) =1;
       end
    end   % for i nrex

 if min(col) == 1
  sol(k,node) = 0; %max(y);
 else
  sol(k,node) = min(col);
 end
   end   % for node

 end   % for k


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   MBMS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [weight,filter] = mbms(A,B,frac)
%This function returns an MBMS approximate solution for MAX-MIN FRE and a filter

[nrex,rezin] = size(A);
[nrex,nodes] = size(B);

 counter = zeros(rezin,nodes);
 reference = counter;
 weight = reference;
 filter = ones(rezin,nodes);

%frac =  0.045;

for node = 1:nodes                          %initialise ...
 for i = 1:rezin
   w(i) = 0;
   s(i) = 0;
   o(i) = 0;
   fi(i) = 1;
 end %for i initialise

                                   %for all other examples ...
 for k = 1:nrex
   k;
   x = A(k,:);
   y = B(k,node);

  for i = 1:rezin
     if x(i)-w(i) > frac
      w(i) = x(i);
      s(i) = s(i) +1;
      o(i) = y;
     elseif abs(x(i)-w(i)) < frac
          o(i) = min(o(i),y);
     else
       s(i) = s(i) +1;
     end

   end   % for i rezin


 end   % for k=2

 for i = 1:rezin
  if s(i) <=1
   w(i) = 0;
   o(i) = 0;
   fi(i) = 0;
  end  %if s(i
 end %for i
```

```
for i = 1:rezin
  if s(i) == max(s) %changes in absolutely all examples!
    w(i) = 0;
    o(i) = 0;
    fi(i) =0;
   end  %if s(i
 end %for i

 counter(:,node)= s';
 reference(:,node) = w';
 weight(:,node) = o';
 filter(:,node) = fi';

end   % for node
```