Geometric Fuzzy Logic Systems

Simon C. Coupland, B.Sc.

Submitted in partial fulfilment of
the requirements for the degree of Doctor of Philosophy
at De Montfort University

June, 2006

THESIS CONTAINS CD/DVD

PAGES NOT SCANNED AT THE REQUEST OF THE UNIVERSITY

SEE ORIGINAL COPY OF THE THESIS FOR THIS MATERIAL

Acknowledgements

I would like to thank my supervisors Professor Robert John, Dr Mario Gongora and Dr Peter Innocent for their support and guidance and Professor Jerry Mendel for the helpful and informative discussions that I have had with him. I would also like to thank my friends and family for supporting me throughout the course of this research.

Abstract

There has recently been a significant increase in academic interest in the field of type-2 fuzzy sets and systems. Type-2 fuzzy systems offer the ability to model and reason with uncertain concepts. When faced with uncertainties type-2 fuzzy systems should, theoretically, give an increase in performance over type-1 fuzzy systems. However, the computational complexity of generalised type-2 fuzzy systems is significantly higher than type-1 systems. A direct consequence of this is that, prior to this thesis, generalised type-2 fuzzy logic has not yet been applied in a time critical domain, such as control. Control applications are the main application area of type-1 fuzzy systems with the literature reporting many successes in this area. Clearly the computational complexity of type-2 fuzzy logic is holding the field back.

This restriction on the development of type-2 fuzzy systems is tackled in this research. This thesis presents the novel approach of defining fuzzy sets as geometric objects - geometric fuzzy sets. The logical operations for geometric fuzzy sets are defined as geometric manipulations of these sets. This novel geometric approach is applied to type-1, type-2 interval and generalised type-2 fuzzy sets and systems. The major contribution of this research is the reduction in the computational complexity of type-2 fuzzy logic that results from the application of the geometric approach.

This reduction in computational complexity is so substantial that generalised type-2 fuzzy logic has, for the first time, been successfully applied to a control problem - mobile robot navigation. A detailed comparison between the performance of the generalised type-2 fuzzy controller and the performance of the type-1 and type-2 interval controllers is given. The results indicate that the generalised type-2 fuzzy logic controller outperforms the other robot controllers. This outcome suggests that generalised type-2 fuzzy systems can offer an improved performance over type-1 and type-2 interval systems.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis reports the results of research into type-2 fuzzy logic systems. In particular, it is concerned with the problem of the computational complexity of such systems. This problem has, to date, prevented the application of type-2 fuzzy systems to real applications. This research proposes a novel solution to this problem using geometric models. The new geometric approach to fuzzy logic, it is argued in this thesis, is a significant contribution to the field of fuzzy logic.

This Chapter sets out the main issues that surround this research. The reasons why type-2 fuzzy methods can give better models than type-1 fuzzy methods for certain concepts are discussed. Issues surrounding the use of discrete fuzzy sets in computer implementations of fuzzy systems are considered and the reasons why the computational complexity of type-2 fuzzy systems is such a substantial problem are identified. All of these points are revisited in the main body of the thesis. As the argument of this thesis unfolds, it becomes possible to enrich the points made in this Chapter with technical arguments and worked examples. The aims and objectives of this research are stated in the research hypothesis. The organisation of the remainder of the thesis is also given.

1.1 Fuzzy Logic Systems

Fuzzy logic systems have had significant successes over the past 40 years (Seising 2005). However, recently the ability of type-1 fuzzy sets to model uncertain concepts has come under close scrutiny. This has led researchers to propose the use of richer fuzzy models, type-2 fuzzy sets.

1.1.1 Type-1 Fuzzy Logic Systems

Real world decisions are taken based on vague knowledge that cannot be modelled by crisp sets (Klir and Folger 1998). The sharp boundaries of crisp sets limit the descriptive power of such a model. Consider the example of the crisp set *long books*. In this example let the set *long books*

be defined as a book containing over 799 pages. According to this crisp definition a 820 page is a long book, so is 1020 page book. This definition seems perfectly adequate. Experts agree that both of these books are long. The flaw in this method of knowledge representation is the fact that although both books are long, one book is longer than the other. This fact is not being captured by the set *long books*, as this cannot be modelled by a crisp set However, in many cases decisions are made based on such facts.

> "A sharp, unambiguous distinction exists between the members and nonmembers of the class or category represented by the crisp set. Many of the collections and categories we commonly employ, however (for instance, in natural language), such as the classes of tall people, expensive cars, highly contagious diseases, numbers much greater than 1, or sunny days, do not exhibit this characteristic."
>
> From Klir and Folger (1988) page 3

Type-1 fuzzy sets do not dichotomise assertions into binary, true or false categories. Fuzzy sets have graduated boundaries, memberships of such sets can be partial. According to the crisp set, a book containing 799 pages is not a long book, although one containing 800 pages is a long book. The addition of a single page has changed the classification of a book from not a *long book* to being a *long book*. The addition of a single page has made a significant change to the classification of the length of the book. Fuzzy classifications and assertions are not simply true or false but are true to a degree. Reworking the previous example, let the set of *long books* now be a type-1 fuzzy set. Type-1 fuzzy sets have graduated boundaries. One consequence of this is that the law of the excluded middle, $A \cup \neg A = X$, is broken. This means that a 799 page and a 800 page book can both be members and non-members of the class *long books*. This is because fuzzy sets measure membership as a matter of degree. This allows type-1 fuzzy sets to be more expressive than crisp sets, that is fuzzy sets are capable of modelling facts that crisp sets. Let the 820 page book be a *long book* to a degree of 0.8. Let the 1020 page book be a *long book* to a degree of 0.95. Both books are *long books*, however, the 1020 page book is longer than the 820 page book. This fact that could not be modelled by a crisp set, is now being captured by a fuzzy set. The concept of a *long book* is a vague concept that cannot be modelled by the crisp set *long books*. This example demonstrates how a vague concept can be modelled by the fuzzy set *long books*. Fuzzy logic systems provide the means for reasoning and making decisions based on the crisp facts available and the knowledge encoded in fuzzy sets and rules. An improvement in the decision making process results from fuzzy sets modelling the vague nature of concepts.

> "The key idea of fuzziness comes from the multivalued logic of the 1920s: *Everything is a matter of degree.* A statement of fact like "The sky is blue" or "The angle

is small" or "$e = mc^2$" does not have a binary truth value. It has a *vague* or "fuzzy" truth value between 0 and 1."

From Kosko (1995) page 4

Type-1 fuzzy logic systems extend the crisp logical operations 'and', 'or' and 'implies' in order to process fuzzy production rules. The rules in a fuzzy system only fire to a degree. This propagates the vagueness of the concepts through the inferencing process. Type-1 fuzzy logic systems, incorporating vague concepts, have been successful in a variety of contexts. However, the applications reported in the literature are predominantly in the area of control systems.

Type-1 fuzzy sets, although an improvement over the crisp model, do not capture every aspect of the decision making process (John 1999*a*). Concepts in the decision making process are not only vague, they are also uncertain. Throughout this thesis a distinction is made between the concepts of vagueness and uncertainty.

- A concept is considered vague if that concept cannot be adequately defined by crisp boundaries, and
- a concept is considered to be uncertain if the boundaries of that concept are themselves vague i.e., the knowledge about a concept is itself vague.

The fact that a book of 820 pages is 0.8 a *long book* requires a completely accurate model of the concept of a *long book*. There are a variety of reasons why such a model is not realistic. A group of experts may disagree about what constitutes a *long book*. The definition of a concept may vary under certain environmental conditions. The opinion of an expert may change over time. To model uncertainty, the vagueness of knowledge, in fuzzy sets requires the fuzzy set model to be extended.

1.1.2 Type-2 Fuzzy Logic Systems

The model of the decision making process needs to take account of the vagueness of the concepts being reasoned with and the uncertainty associated with these concepts. Type-1 fuzzy sets require that concepts are defined using crisp (not vague) measurements. The boundaries of the concepts are vague, however, the membership of a particular element is crisp. In the earlier example the 820 page book was defined to be 0.8 a *long book*. This means that a 820 page book has a crisp membership grade of 0.8 in the type-1 fuzzy set *long book*. This crisp measure of set membership does not encapsulate any notion of uncertainty. The membership functions of type-1 fuzzy sets are based on the numerical and linguistic knowledge of the system. Such information may suffer

from uncertainties. To model uncertainty the membership function of a fuzzy set needs to express the vagueness of the conceptual knowledge that the set is modelling.

> "We conclude that increased fuzziness in a description means increased ability to handle inexact information in a logically correct manner"
>
> Hisdal (1981) page 385

Type-2 fuzzy sets capture this increased fuzziness. Like type-1 fuzzy sets, the boundaries of type-2 fuzzy sets are graduated. This means that type-2 fuzzy sets can model vague concepts. The membership grade of a particular element in a type-2 fuzzy set is measured as a vague number, expressed as a type-1 fuzzy set. Thus, the boundaries of type-2 fuzzy set are not only graduated but are also vague. Type-2 fuzzy sets not only model the vagueness of a concept but the vagueness of the knowledge about the concept - the uncertainty. Returning for a final time to the example of *long books*. Using type-2 fuzzy sets the 820 page book would be *about 0.8* a *long book* and 1020 page book *about 0.95* a *long book*. The set membership itself is now vague and is therefore modelled by a type-1 fuzzy set.

1.1.3 Generalised and Interval Type-2 Fuzzy Sets

There are two distinct classes of type-2 fuzzy set that can be easily confused with one another. Generalised type-2 fuzzy sets (John 1998*a*, 1998*b*, 1999*a*, 1999*b*, Karnik and Mendel 1998*a*, 1998*b*) are type-2 fuzzy sets where the membership grade is given by a type-1 fuzzy set. This is the class of type-2 fuzzy sets that this thesis is mainly concerned with. Whenever this research refers to type-2 fuzzy sets, it is generalised type-2 fuzzy sets that are being referred to. Early literature in the field also used this convention. Type-2 interval fuzzy sets (Türkşen 1993*a*, 1993*b*, Liang and Mendel 2000*b*) are type-2 fuzzy sets where the membership grade is given by a crisp interval set. Recently, the type-2 literature has often used the term type-2 fuzzy sets when discussing type-2 interval fuzzy sets. This maybe due to the recent literature predominantly discussing type-2 interval fuzzy methods. To avoid confusion, in this thesis the term 'interval' will always be used when referring to type-2 interval fuzzy sets.

1.1.4 Summary

Type-2 fuzzy sets offer a richer, fuller, more expressive method for describing a concept. However, type-2 fuzzy sets are also far more complex than type-1 fuzzy sets. Correspondingly, any decision making process that uses type-2 fuzzy sets is more complex. It is the view of the author that

the level of computational complexity of type-2 fuzzy logic systems has to date held back the development of such systems. This is the central point that this thesis seeks to address.

1.2 Computer Implementations of Fuzzy Logic Systems

To make best use of fuzzy systems they must be implemented on computer hardware or software, allowing fast, autonomous execution. However, there can be a fundamental disparity between a theoretical fuzzy set and the computer model of that fuzzy set. Fuzzy sets, of all types, will often be defined over a continuous domain such as height, distance, voltage, etc. Each point in the domain of a fuzzy set has a corresponding membership grade in that fuzzy set. For continuous fuzzy sets there are an infinite number of points in the domain, and therefore, an infinite number of membership grades. Infinite concepts cannot be directly implemented in computer memory and processing architectures, requiring other solutions to be found. To overcome this problem, it is standard practice to discretise continuous fuzzy sets when implementing a computer model. Discretisation takes a given set of points from the domain of the fuzzy set and maps these points to the respective membership grades in the fuzzy set. This gives a finite fuzzy set model which can be easily modelled in computer software and hardware. This ease of implementation comes at a cost. Whenever a continuous function is discretised, information about that function is lost. This happens to the membership functions of continuous fuzzy sets whenever such sets are implemented in a computer. Consider the example fuzzy set *Middle Aged* depicted in Figure 1.1(a). Discretising the domain of this set at ten equidistant points gives the fuzzy set *Middle Aged* depicted in Figure 1.1(b). The information lost through this discretisation process is depicted in Figure 1.1(c).

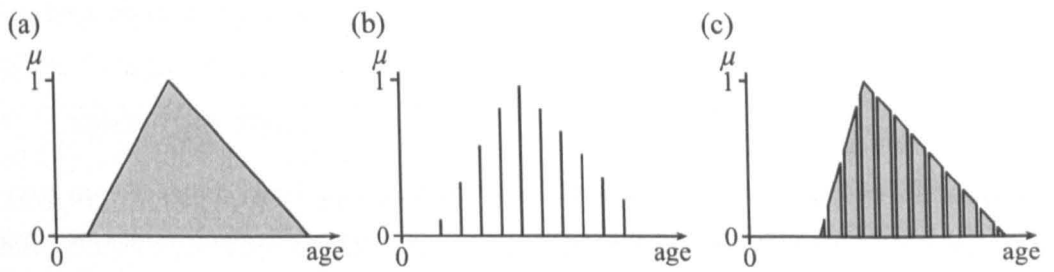


Fig. 1.1. (a) The Continuous Fuzzy Set *Middle Aged*. (b) The Discrete Fuzzy Set *Middle Aged*. (c) The Information About The Fuzzy Set *Middle Aged* Which is Lost Through Discretisation.

Under certain circumstances it is possible to retrieve some of this lost information. For example, it is possible to linearly interpolate between consecutive points in the domain of a set. In some cases this will give the correct membership grade, in some cases, particularly near apex points, it will not. As the inferencing process continues, the discrete sets undergo more processing, linear interpolation will become less and less accurate. The geometric approach proposed in this thesis

allows fuzzy sets to be modelled over continuous domains. By modelling fuzzy sets over continuous domains it may be possible to reduce and, in certain cases, eliminate this loss of information.

1.3 The Computational Problems of Type-2 Fuzzy Logic

Type-1 fuzzy systems are all based on a 2-dimensional set model, mapping the elements of the set to values between zero and one. Type-2 fuzzy sets depart from this 2-dimensional model. The type-2 fuzzy set model is a 3-dimensional set model. Type-2 fuzzy sets map domain elements to a function that maps values between zero and one to a second set of values between zero and one. This third dimension gives type-2 fuzzy sets the expressive power needed to model uncertainty. However, this extra dimension has a serious impact on the computational complexity of such systems. Typically, the amount of computation it takes for a type-2 fuzzy logic system to arrive at a decision is significantly greater than that of crisp or type-1 fuzzy logic systems. No general comparison has yet been made between the complexity of these approaches, due to the number of variables that effect system complexity. In this thesis, however, theoretical and empirical comparisons are used to illustrate the relative complexity of various fuzzy approaches in specific examples.

To date, no applications of generalised type-2 fuzzy logic in a time critical domain such as control or signal processing have been reported. One of the main applications areas of type-1 fuzzy logic is control systems. The author's view is that the fact that no applications of generalised type-2 fuzzy logic have been reported in control suggests that the computational problems of such systems are holding back practical applications of the theory.

1.3.1 Logical Operations

The extra dimension of type-2 fuzzy sets increases the amount of computation required to perform a logical operation. Logical operations on crisp sets are trivial and can be represented with truth tables. Type-1 fuzzy sets use the t-norm and t-conorm set of functions to perform logical operations. T-norms and t-conorms are functions that conform to certain properties as described by Mizumoto and Tanaka (1981), Dubois and Prade (1981) and Karnik and Mendel (1998). According to Mendel (2001*a*), the most commonly used t-norms are the minimum and algebraic product and the most commonly used t-conorm is maximum. Logical operations on type-1 fuzzy sets are performed by finding the minimum and maximum (and other t-norms and t-conorms) of two values between zero and one. The logical operations on type-2 fuzzy sets are far more complex. The membership grade of a type-2 fuzzy set is a function. To logically combine two such functions ev-

ery mapping in one function must be logically combined with every mapping in the other function. This results in large number of functional mappings, many of which are redundant. These redundant mappings must be identified and removed. These operations are significantly more complex than computing a binary truth or performing a single t-norm or t-conorm operations.

1.3.2 Output Processing

Fuzzy logic systems are often required to express a decision as a crisp result. However, the decision arrived at by the fuzzy system is expressed as a fuzzy set. To convert a fuzzy set to a crisp decision requires the process known as defuzzification. Type-1 defuzzifiers are not trivial, however, the amount of computation required does not prohibit real-time execution. As with the logical operations, the addition of a third dimension results in increased complexity in the output processing of a type-2 fuzzy system. Type-2 defuzzifiers require a far large amount of computation that type-1 defuzzifiers to give a crisp decision. Acquiring a crisp result from a type-2 fuzzy set is a two stage process. First the set must be type-reduced. Type-reduction (Karnik and Mendel 1998*c*, 2001*a*) takes a type-2 fuzzy set and produces a type-1 fuzzy set that represents the centroid of that set. This type-reduced fuzzy set can then be defuzzified, resulting in a crisp decision. Type-reduction finds every possible type-1 fuzzy set that could lie inside the vague boundaries of the type-2 fuzzy set. This is usually a very large number of sets. Each of these type-1 fuzzy sets is then defuzzified. This defuzzification of a large number of type-1 fuzzy sets is what causes the explosion in computational cost. Each of these defuzzified values is then mapped to a membership grade in the type-reduced set. Redundant mappings must then be removed before the type-reduced set can be defuzzified. Type-reduction is clearly a complex operation.

1.3.3 Summary

Type-2 fuzzy sets offer the ability to model and reason with uncertain concepts. Many real-world problems contain inherent uncertainties and may benefit from the type-2 fuzzy approach. However, type-2 fuzzy systems have a level of computational complexity that is significantly higher than type-1 and type-2 interval fuzzy systems. The aim of this thesis is to reduce the computational complexity of the 3-dimensional fuzzy logic operations by using techniques developed in computational geometry. If the computational problems of type-2 fuzzy logic can be overcome then this would be a significant result for the field of fuzzy logic. These points are set out formally in the research hypothesis.

1.4 Research Hypothesis

The research hypothesis addressed in this thesis can be stated as:

"Type-2 fuzzy logic systems offer a great deal in terms of modelling uncertain concepts and inferencing under uncertain conditions. However, the computational complexity of type-2 fuzzy logic is arresting the research and development of such systems. Geometric methods can resolve these computational problems making it possible for type-2 fuzzy logic to be applied in a time critical domain such as control."

As will be shown (in Chapter 2) there has recently been a significant growth in the number of papers on the subject of type-2 fuzzy logic being published. A majority of these publications, indeed all the publications on time critical applications, are only concerned with type-2 interval fuzzy logic. Type-2 interval fuzzy logic is computational simpler than type-2 but has a reduced capacity for modelling uncertainty. There is potential for generalised type-2 fuzzy systems to outperform type-2 interval systems under conditions where uncertainties are high. This potential can only be achieved when the computational problems of type-2 fuzzy logic have been resolved.

To address the issue of complexity more efficient representations and operators are needed. Computational geometry offers an ideal solution with well defined methods for manipulating 2 and 3 dimensional objects efficiently. The main issue addressed in this thesis is how to model fuzzy sets as geometric objects and how to manipulate such objects to define logical operations. The research reported here (in Chapters 3 and 4) provides a complete, novel geometric model of type-1, type-2 interval and generalised type-2 fuzzy logic.

As the argument in this thesis develops it becomes clear (in Chapter 5) that the computational complexity of type-2 fuzzy logic can be reduced so significantly that control applications are possible. Furthermore it is demonstrated that type-2 fuzzy logic can give an improved performance in such an application.

1.5 Structure of the Thesis

The remainder of this thesis is structured in the following way:

- Chapter 2 describes the literature on type-2 fuzzy logic. A comprehensive set of fundamental definitions, representations and operators is compiled. These definitions and operators are critically reviewed. Previous work on improving the efficiency of type-2 fuzzy systems is discussed. A brief description of type-2 interval fuzzy logic is given. The relative merits

of generalised type-2 and type-2 interval systems are discussed. An overview of the historical development of type-2 fuzzy logic is given, demonstrating the necessity and timeliness of this work.

- Chapter 3 presents the novel geometric approach to type-1 fuzzy logic systems. The limited existing work that relates to the geometric approach is discussed and critically reviewed. The geometric methods that are required to define geometric fuzzy logic are given and are explored with examples. Geometric type-1 fuzzy sets and systems are defined using 2-dimensional geometric methods. Some issues with the use of discrete fuzzy sets are discussed.

- Chapter 4 presents novel geometric approach to type-2 interval and type-2 fuzzy logic systems. These methods build upon those presented in Chapter 3. A geometric alternative to type-reduction is given for both types of geometric type-2 fuzzy set. The notion of hybridised type-2 fuzzy systems, part discrete and part geometric, is discussed. New methods for converting between discrete and geometric type-2 fuzzy sets and vice-versa are given. This Chapter presents substantial novel theoretical work in the field of type-2 fuzzy logic that forms the core of this thesis.

- Chapter 5 gives the materials, methods and results from a set of experiments designed to compare the performance of discrete and geometric type-1, type-2 interval and type-2 fuzzy logic controllers for a mobile robot. The task to be performed by the robot and the design of the controllers is discussed. A preliminary study is reported which gives the computational performance, in terms of execution speed, of the various discrete and geometric type-2 fuzzy system components. These results are used to design the configuration of two hybrid type-2 fuzzy logic controllers. The design of the experiments is given, with particular attention being paid to the methods of statistical analysis that will be applied to the results. The results from the experiments are given and it is concluded that a hybrid type-2 fuzzy logic controller gave a strong performance, at least equal to the other technologies.

- Chapter 6 provides the conclusion to the thesis, stating what has been achieved and the importance of these achievements. Opportunities for further research on the theory and application of geometric fuzzy logic are discussed.

Chapter 2

Type-2 Fuzzy Logic Systems

The previous Chapter defined the scope and the goals of this thesis. This Chapter presents the field type-2 fuzzy sets and systems. The work presented in this Chapter is needed for the novel geometric approach to fuzzy systems, presented in the following Chapter, to be understood.

Type-2 fuzzy sets and logic provide a mathematically rigorous methodology for reasoning with uncertain terms. Type-2 fuzzy logic is an extension of type-1 fuzzy logic, itself an extension of crisp logic. Klir and Folger (1988) noted that crisp or classical sets dichotomize elements into members and non-members, leaving no ambiguity in the knowledge captured by the set. In crisp logic all logical propositions must be either true or false, black or white, zero or one. Everyday human reasoning, using natural language, does not appear to employ such clear cut categories (Zadeh 1996). Consider how humans think and reason about age. It is often the case that a given persons age is known as a crisp fact. Decisions may be taken based on that crisp knowledge. However the decision making process may not be crisp in nature and may take into account additional vague or uncertain knowledge. In some situations the reasoning and outcomes must be crisp, for example when a person is driving a car they must be legally old enough to drive that car. In this example a crisp set *legally old enough to drive* is sufficient. In other situations it may be advantageous to allow some ambiguity into the reasoning process. For example, when an insurance company calculates the motor insurance premium for a driver, the age of the driver will be very important in calculating the risk associated with that driver. If the driver is young the risk of that driver making a claim will be higher. Once again this piece of inferencing uses the crisp knowledge of driver age, however the set *young* may be better defined as a fuzzy rather than a crisp set. To illustrate this point let the crisp set *young* be the set depicted in Figure 2.1(a). Figure 2.1(a) depicts the function, called the membership function, that describes the crisp set *young*. Let the fuzzy set *young*, also characterised by a membership function, be the set depicted in Figure 2.1(b). Consider how two people, one aged 25 years old and one aged 30 years old, could be reasoned about in a decision making system. A person who is 30 years old is young and their car insurance premium should reflect this. A person who is 25 years old is also young and their car

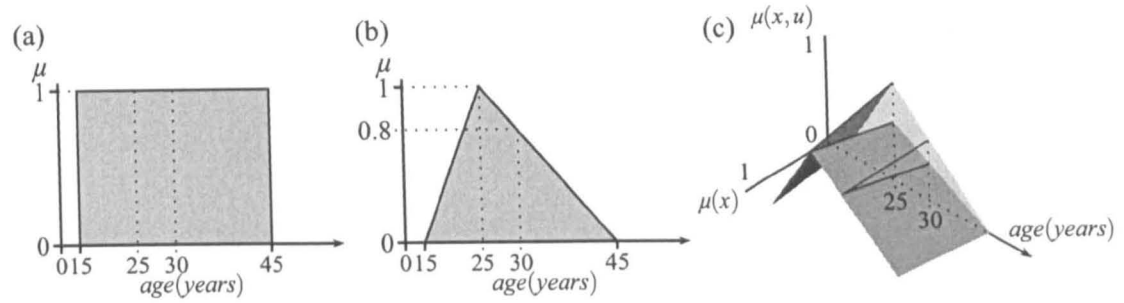


Fig. 2.1. (a) The Crisp Set *Young*. (b) The Type-1 Fuzzy Set *Young*. (c) The Type-2 Fuzzy Set *Young*.

insurance premium should reflect this. However, the 25 year old is younger than the 30 year old and this should be reflected in both their premiums. This is the vague notion that Klir and Folger discussed, illustrated by a practical example. The crisp set *young* is not able to capture the fact that two people may both be young, but one person is younger than the other one. By allowing some ambiguity and/or some vagueness into the set model, such assertions are not only possible but sensible. Type-1 fuzzy logic allows these vague notions to be captured, modelled and reasoned with mathematically.

Type-1 fuzzy logic provides a robust paradigm for computing with vague concepts and partial truths and partial set memberships. However, type-1 fuzzy methods cannot incorporate uncertainty into the set model. To explore this point further return to the example of motor car insurance risk and age. A persons age is an absolute fact, a crisp number that has no uncertainty associated with it. This does not mean that there is no uncertainty in this example. If this person is 27 years old they may be young. A group of experts on motor insurance risk could no doubt agree on this fact. It may be more difficult however to get a group of experts to agree to what *degree* this person is young. If the experts do not agree on a single crisp number for the degree of youngness, there is some vagueness associated with the experts knowledge, then *young* can be regarded as an uncertain term. The modelling of this inter-expert variation is further explored by Mendel (Karnik, Mendel & Liang 1999, Mendel 1999). Mendel produced a survey with sixteen labels including 'a bit', 'some' and 'a moderate amount'. Respondents (in this case US engineering undergraduate students) were asked to assign a range to each of the labels. With each label a boundary of uncertainty was identified from the standard deviations of the responses. This boundary represents the linguistic uncertainty associated with the label. The range of three labels 'a bit', 'some' and 'a moderate amount' are depicted in Figure 2.2. The continuous lines depict the meaning of the label and dashed lines depict the linguistic uncertainty associated with that label. Mendel draws an analogy between this measure of linguistic uncertainty and the standard deviation of a measured random variable.

"When we work in the province of probability, we find it useful and important

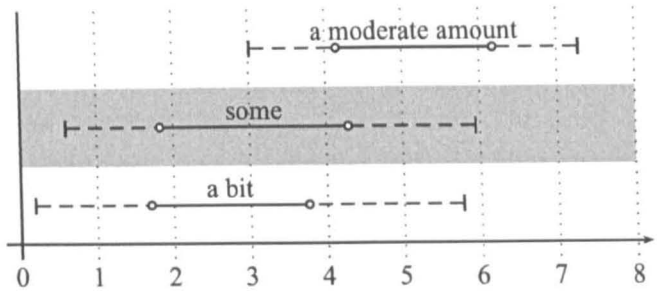


Fig. 2.2. The Linguistic Labels 'a bit', 'some' and 'a moderate amount'. Adapted from Mendel (2001*a*).

to distinguish between the mean and the standard deviation, so why should less be expected of us when we work with linguistic uncertainties?"

From Mendel 1999

Place this in the context of the motor vehicle insurance example; why should the opinion of our experts be disregarded by only taking a combined measure of their opinions? Mendel believes it is better to use a fuzzy set model that reflects the range of opinions expressed by the experts. The fuzzy model needs to capture linguistic uncertainty and Mendel states that to do so requires type-2 fuzzy sets and logic. The author disagrees with this point. Type-2 fuzzy sets can model some aspects of linguistic uncertainty, but by no means offer a complete computational model of a word.

Type-2 fuzzy sets were proposed by Zadeh (1975*a*,1975*b*,1975*c*) as a method for defining the membership grades of fuzzy sets such as *young* with linguistic terms such as *low, medium* and *high*. Zadeh suggests that measuring fuzzy membership grades in this way may allow experts to better understand the terms used. However, Mendel's work on survey based fuzzy sets suggests that experts can easily transfer their knowledge to arbitrary scales when given a specific context and set of terms. Whether a particular membership grade is a label with a context specific term such a *high* or a scale based term such as *about 0.9* the type-2 fuzzy set model still captures the associated linguistic uncertainty. Each membership grade, regardless of the label, gives a possibilistic distribution of the membership grade of a particular point in a domain of a set. This fuzzy measure that captures the vagueness of the knowledge modelled by the fuzzy set, the uncertainty surrounding the knowledge. This is the mechanism by which type-2 fuzzy sets capture the uncertain nature of a concept.

Type-2 fuzzy systems overcome the problem of inter-expert vagueness, where a group of experts may not agree on a specific point by modelling set membership with fuzzy numbers. Another form of uncertainty, intra-expert variation, when a single expert may disagree with their own previous conclusion cannot be obviously modelled by type-2 fuzzy sets. To model this variation in decision

making over time Garibaldi *et al* (Ozen & Garibaldi 2004, Garibaldi, Musikasuwan & Ozen 2005) proposed non-stationary fuzzy systems. Such systems are effectively type-1 fuzzy systems where the membership functions are subject to variation over time. The exact nature of this variation such as random distributions, normal distributions or variation within an interval and the level of this variation are dependent on the specific system. The result is a fuzzy system that behaves in a correct manner with the fine detail of the decision making process varying over time. It may be that type-2 fuzzy systems model the range of decisions that a non-stationary system is capable of producing, however a type-2 fuzzy system will always be deterministic and therefore cannot model the non-deterministic nature of intra-expert variation.

Both inter-expert and intra-expert uncertainty can be viewed as semantic uncertainty, uncertainty about the meanings of the fuzzy sets. This is not the only source of uncertainty that real world systems have to deal with. Mendel (2001*b*) identified four sources of uncertainty:

- Uncertainty about the meanings of words that are used in the rules;
- Uncertainty about the consequent that is used in a rule;
- Uncertainty about the measurements that activate the FLS, and
- Uncertainty about the data that is used to tune the parameters of a FLS.

The first two of these points are concerned with semantic uncertainty. The final two are concerned with noise from sensors, whether during system execution or when data was collected for training the system. Noise from various sources creates uncertainty about the inputs going into such systems. Take the example of a washing machine (Takagi 1994, Kosko 1997) where a fuzzy system is used to decide which wash programme should be used. One of the factors considered when making such a decision is the soiling level of the clothes. Measuring the degree of soiling of an entire load of washing to a high level of precision is a challenging task. One method is to measure the turbidity of the water used during the initial rinse. There are many sources of uncertainty involved in taking such a measurement, such as particulates responding differently to the sensor or larger particulates obscuring smaller ones. In essence these are engineering problems. However, when such problems cannot be overcome or the cost of overcoming them is disproportionate to the advantage gained, a system that can cope with such noise based uncertainties will be very useful.

> "First, computing with words* is a necessity when the available information is not precise enough to justify the use of numbers. And second, computing with words is advantageous when there is a tolerance for imprecision, uncertainty and partial truth that can be exploited to achieve tractability, robustness, low cost solution and better rapport with reality."
>
> From Zadeh (1997) page 103

Both type-1 and type-2 fuzzy systems help overcome sensory noise as they both allow small changes in measurements to only have a small change on the logical value of an input. This leads to smoother control surfaces than crisp systems based on a comparable number of sets. From a theoretical point of view type-2 fuzzy systems should be better than type-1 systems at coping with noisy operating conditions. This statement is based on the fact that type-2 fuzzy sets use type-1 fuzzy sets to model membership values. A control surface represents an approximation of a non-linear function (Kosko 1997). A type-2 fuzzy set will, in effect increase the resolution of this function approximation, giving a high resolution, a smoother, control surface. This assertion is tested empirically in Chapter 5 of this thesis.

This Section has demonstrated that real world problems often have to be solved in the face of large uncertainties. Type-2 fuzzy logic models and takes decisions with these uncertain terms. Type-1 fuzzy logic only models the vague nature of a concept and not the uncertainty associated with it. The following Section of this Chapter, Section 2.1, discusses type-2 fuzzy sets. Section 2.2 describes the logical methods used to make decisions based on type-2 fuzzy sets. Section 4.1 gives some novel work that increases the usefulness of type-2 fuzzy logic. Section 3.4 discusses the additional problems of discretising type-2 fuzzy sets that are based on a continuous domain. Section 2.3 describes the widely used subset of type-2 fuzzy logic, type-2 interval fuzzy logic. Section 2.4 describes how the field of type-2 fuzzy logic has developed over the past 30 years. Finally, Section 2.5 discusses the outcomes of this Chapter and how the work presented here relates to the rest of this thesis.

2.1 Type-2 Fuzzy Sets

The previous Section discussed the motivation and rationale behind the development of type-2 fuzzy methods. This Section provides definitions and representations of type-2 fuzzy sets. These will be built upon later in this Chapter when type-2 fuzzy logic is discussed.

*Zadeh claims that fuzzy logic is part of a wider notion of computing with words, see Zadeh 1996.

2.1.1 Type-n Fuzzy Sets

This thesis is primarily concerned with the development of fast and efficient methods for defining type-2 fuzzy logic operations by taking a geometric approach. The discussion of type-2 fuzzy systems would not be complete without a brief digression into type-n fuzzy systems. The introduction to this Chapter discussed the use of crisp sets in logic systems. In some applications where there is no uncertainty and no vagueness associated with the information contained in a logic system it is right and proper that a crisp system be used. Type-1 fuzzy systems are an extension of crisp systems, which are sometimes called type-0 fuzzy systems. When there is some vagueness about the concepts being modelled in a logic system type-1 fuzzy methods should then be employed. When there is some vagueness surrounding the knowledge of the concepts being modelled, the concepts are uncertain, then type-2 fuzzy logic provides a methodology for modelling this uncertainty. These three fuzzy technologies, type-0, type-1 and type-2, all have well defined methods and operations. There is no theoretical reason why fuzzy technologies should stop at type-2. To this end type-n fuzzy sets (Zadeh 1975*a*) are defined below:

Definition 2.1 *A fuzzy subset of type-0 over a universe X is an element of X. A fuzzy subset of type-n over universe X is characterised by a membership function whose domain X is mapped to a fuzzy subset of type $n-1$ over the real interval* $[0,1]$.

Adapted from Zadeh (1975*a*)

This recursive definition gives the possibility of n dimensional fuzzy sets. To date the highest order of fuzzy set to be explored in any detail is the type-2 fuzzy set. As dimensions are added to the fuzzy set nature of the fuzzy set model is changed.

- Type-0 fuzzy sets model concepts defined with crisp numerical boundaries;
- type-1 fuzzy sets model concepts defined with vague boundaries;
- type-2 fuzzy sets model concepts defined with uncertain boundaries, and
- type-3 fuzzy sets take this one stage further. Each element belongs to a degree given by a type-2 fuzzy number. There is uncertainty surrounding the measurement of the membership grade of each element. This can be viewed a piece of knowledge where the measurement of the vagueness of that knowledge is itself vague.

This description suggests that type-3 fuzzy systems may well be useful in some applications where the subject knowledge is highly uncertain. However, no operations currently exist for type-3 systems and the performance problems encountered with type-2 systems would be significantly

more challenging. For this reason this research focuses on exploiting the useful type-2 methods but does not dismiss the potential of type-3 or even higher type methods in certain applications.

This Section has discussed the context of type-*n* fuzzy sets that type-2 fuzzy sets and logic are defined in. This brief insight shows type-2 methods as a developmental step rather than an endpoint in fuzzy system development. The following Section discusses the widely accepted definition of type-2 fuzzy sets put forward by Mendel and John (2002).

2.1.2 The Mendel and John Definition

Zadeh proposed the idea of a type-2 fuzzy set in a trilogy of papers (1975*a*, 1975*b*, 1975*c*). Basic operations were defined at this point but have since been expanded by Mizumoto and Tanaka (1976 and 1982), Dubois and Prade (1980 and 1982), Karnik *et al* (1999, 2001a, 2001b) and Mendel and John (2002). The definition of a type-2 fuzzy set given by Mendel and John is now widely used across the literature.

Definition 2.2 *A type-2 fuzzy set, denoted $\tilde{A}$, is characterised by a type-2 membership function $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$, i.e.,*

$$\tilde{A} = \{((x,u),\mu_{\tilde{A}}(x,u)) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{2.1}$$

in which $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$, X is the domain of the fuzzy set and J_x is the domain of the secondary membership function at x. $\tilde{A}$ can also be expressed as

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u)/(x,u) \quad J_x \subseteq [0,1] \tag{2.2}$$

where $\int\int$ denotes union over all admissible x and u. For discrete universes of discourse $\int$ is replaced by $\sum$.

From Mendel and John (2002)

A type-1 fuzzy set is a collection of ordered pairs of domain elements mapped to their membership grades. The Mendel and John definition extends the type-1 definition, modelling a type-2 fuzzy set as a collection of nested ordered pairs. The primary ordered pair is the same as the type-1 ordered pair, domain elements mapped to their respective primary membership grades. The secondary ordered pair maps this primary ordered pair to a secondary membership grade which measures the possibility that the ordered pair belongs to the type-2 fuzzy set. According to the Mendel and John definition, the membership function of type-2 fuzzy set maps such primary ordered pairs to their

respective membership grades. A Venn diagram depicting the relationship between the terms in the fuzzy set is given in Figure 2.3.

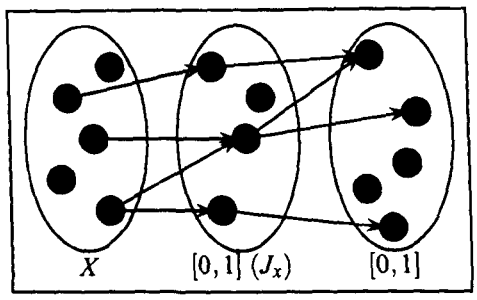


Fig. 2.3. A Venn Diagram of the Relationship between the Terms in a Type-2 Fuzzy Set According to the Mendel and John (2002) Definition.

This definition does not view the membership grade of type-2 fuzzy set as a fuzzy number between zero and one. For that, the vertical slice representation, also given by Mendel and John, is used.

2.1.3 The Vertical Slice Definition

The vertical slice definition views the membership grade at each point in a type-2 fuzzy set as a type-1 fuzzy number bounded by the interval $[0,1]$.

Definition 2.3 *At each value of x, such that $x \in X$, in the type-2 fuzzy set $\widetilde{A}$, i.e., $\mu_{\widetilde{A}}(x)$ maps to a secondary membership function $f(x)$, which map values in $[0,1]$ to values in $[0,1]$. Let the domain of the secondary membership function denoted by J_x then;*

$$\widetilde{A} = \int_{x \in X} \left[\int_{u \in J_x} f_x(u)/u \right] \Big/ x \tag{2.3}$$

Where $J_x \subseteq [0,1]$, $x \in X$, $u \in [0,1]$ and $f_x(u) \in [0,1]$.

Adapted from Mendel And John (2002)

A Venn diagram depicting the vertical slice representation of a type-2 fuzzy set $\widetilde{A}$ is given in Figure 2.4. The vertical slice definition of a type-2 fuzzy set requires some additional terminology to be defined. A secondary membership function of a type-2 fuzzy set $\widetilde{A}$, the membership grade at a given point x, is a type-1 fuzzy number bounded by the interval $[0,1]$ given by $\mu_{\widetilde{A}}(x)$. The domain of a secondary membership function is called the primary membership, J_x being the primary membership of $\mu_{\widetilde{A}}(x)$, where $J_x \subseteq [0,1]$. The amplitude at a specific point u in a secondary

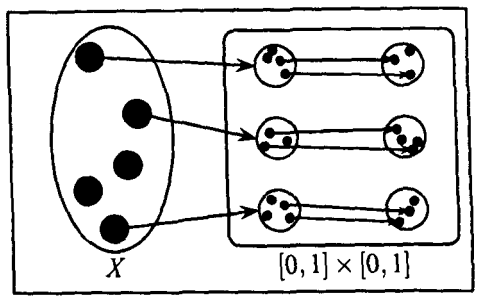


Fig. 2.4. A Venn Diagram of the Relationship between the Terms in a Type-2 Fuzzy Set According to the Vertical Slice Definition.

membership grade is called the secondary grade which is given by $\mu_{\tilde{A}}(x,u)$. This terminology will be used throughout the rest of this thesis.

The vertical slice definition results in the membership function of a type-2 fuzzy set having two interpretations, one giving secondary membership functions and one giving secondary grades. The membership grade of a type-2 fuzzy set $\tilde{A}$ at a point x is given below.

$$\mu_{\tilde{A}}(x) = \int_{u \in J_x \subseteq [0,1]} f_x(u)/u \tag{2.4}$$

where $x \in X$ and f_x is the secondary membership function at x. Throughout this thesis this definition will be called upon as it is very useful when defining logical operations for type-2 fuzzy sets. The membership grade, the secondary grade, of a type-2 fuzzy set $\tilde{A}$ at a point u at the domain point x is given below.

$$\mu_{\tilde{A}}(x,u) = f_x(u) \quad ; \quad f_x = \mu_{\tilde{A}}(x) \tag{2.5}$$

In Coupland and John (2005*b*), the author showed that these two interpretations can be combined into a single definition by allowing the membership function of a type-2 fuzzy set to be a curried function. A curried function is a function where the result of that function is itself a function. In the case of a membership function of a type-2 fuzzy set, currying produces a membership function that takes the domain element as an argument and returns the secondary membership function associated with that point. This is no different from definition 2.2 except that this function may be partially applied.

2.1.4 The Representation Theorem

The two definitions of a type-2 fuzzy set provided here offer alternative models of the equivalent membership function of a type-2 fuzzy set. The representation theorem (Mendel & John 2002) uses a completely different underlying model of what constitutes a type-2 fuzzy set. The repre-

sentation theorem states that a type-2 fuzzy set can be decomposed into a collection of type-2 embedded sets. The benefit of using the representation theorem is that operations can be derived directly from type-1 fuzzy methods without the use of the extension principle (see Section 2.2.1). The downside is that for any useful type-2 fuzzy set the number of embedded sets needed to represent that set is very large.

Definition 2.4 *For discrete universes of discourse X and U, an embedded type-2 set $\widetilde{A}_e$ has N elements, where $\widetilde{A}_e$ contains exactly one element from $J_{x_1}, J_{x_2}, \ldots, J_{x_N}$, namely $u_1, u_2, \ldots, u_N$, each with its associated secondary grade, namely $f_{x_1}(u_1), f_{x_2}(u_2), \ldots, f_{x_N}(u_N)$, i.e.,*

$$\widetilde{A}_e = \sum_{i=1}^{N} [f_{x_i}(u_i)/u_i]/x_i \quad u_i \in J_{x_i} \subseteq U = [0,1] \tag{2.6}$$

Where $\widetilde{A}_e^j$ is the j^{th} embedded set in $\widetilde{A}$ and M_i is the number of points in the domain of the i^{th} secondary membership function of $\widetilde{A}$. Set $\widetilde{A}_e$ is embedded in $\widetilde{A}$, and, there are a total† of $\prod_{i=i}^{N} M_i$.

Adapted from Mendel and John (2002)

A discrete type-2 fuzzy set $\widetilde{A}$ can be represented as the union of its type-2 embedded sets, i.e,

$$\widetilde{A} = \sum_{j=i}^{n} \widetilde{A}_e^j \tag{2.7}$$

where

$$n = \prod_{i=1}^{N} M_i \tag{2.8}$$

The representation theorem allows operations for type-2 fuzzy sets to be defined without the need of the extension principle. The major disadvantage of doing this (as noted by Mendel and John) is that such operations are terribly inefficient and contain an enormous amount of redundancy. One area where the representation theorem has proven to be practically useful is in the definition of arithmetic operators for type-2 fuzzy numbers (Coupland & John 2003). Consider the two type-2

† For continuous type-2 fuzzy sets, there are an uncountable number of embedded type-2 fuzzy sets, and this concept is not very useful.

fuzzy numbers $\widetilde{about}\ 3$ and $\widetilde{about}\ 12$ enumerated below.

$\widetilde{About}\ 3 =$

$$\begin{aligned}
&(1/0+0.6/0.1+0.3/0.2) && /1+\\
&(0.1/0.3+0.6/0.4+1/0.5+0.7/0.6+0.2/0.7) && /2+\\
&(1/1) && /3+\\
&(0.1/0.3+0.6/0.4+1/0.5+0.7/0.6+0.2/0.7) && /4+\\
&(1/0+0.6/0.1+0.3/0.2) && /5
\end{aligned}$$

$\widetilde{About}\ 12 =$

$$\begin{aligned}
&(1/0+0.8/0.1+0.4/0.2+0.2/0.3+0.1/0.4) && /10+\\
&(0.2/0.5+1/0.6+0.4/0.7) && /11+\\
&(1/1) && /12+\\
&(0.2/0.5+1/0.6+0.4/0.7) && /13+\\
&(1/0+0.8/0.1+0.4/0.2+0.2/0.3+0.1/0.4) && /14
\end{aligned}$$

In order to add these two type-2 fuzzy numbers together every embedded set in $\widetilde{about}\ 3$ has to be added to every embedded set in $\widetilde{about}\ 12$ as stated below:

$$\widetilde{About}\ 3+\widetilde{About}\ 12=\sum_{j=1}^{225}\sum_{i=1}^{225}\widetilde{About}\ 3_e^j+\widetilde{About}\ 12_e^i \tag{2.9}$$

The addition of the first embedded sets of $\tilde{About}\ 3$ and $\tilde{About}\ 12$ is given below:

$$\begin{aligned}
\widetilde{About}\ 3_e^1 &= (1/0)/1+(0.1/0.3)/2+(1/1)/3+(0.1/0.3)/4+(1/0)/5\\
\widetilde{About}\ 12_e^1 &= (1/0)/10+(0.2/0.5)/11+(1/1)/12+(0.2/0.5)/13+(1/0)/14\\
\widetilde{About}\ 15_e^1 &= (1/0)/11+(0.2/0)/12+(0.1/0.3)/13+(0.2/0.5)/14+(1/1)/15\\
&\quad +(0.2/0.5)/16+(0.1/0.3)/17+(0.2/0)/18+(1/0)/19
\end{aligned}$$

Once all 255 embedded sets have been added in this way, the union of all these sets is taken. This

results in a final type-2 fuzzy number which is given below:

$$\widetilde{about}\ 15 =$$

$$\begin{aligned}
&(1.0/0.0+0.6/0.1+0.3/0.2) && /11+\\
&(1.0/0.0+0.8/0.1+0.4/0.2+0.2/0.3+0.1/0.4) && /12+\\
&(0.2/0.3+0.6/0.4+1.0/0.5+0.7/0.6+0.2/0.7) && /13+\\
&(1.0/0.5+1.0/0.6+0.4/0.7) && /14+\\
&(1.0/1.0) && /15+\\
&(1.0/0.5+1.0/0.6+0.4/0.7) && /16+\\
&(0.2/0.3+0.6/0.4+1.0/0.5+0.7/0.6+0.2/0.7) && /17+\\
&(1.0/0.0+0.8/0.1+0.4/0.2+0.2/0.3+0.1/0.4) && /18+\\
&(1.0/0.0+0.6/0.1+0.3/0.2) && /19
\end{aligned}$$

This example demonstrates that the representation theorem is useful for the definition of fuzzy arithmetic but also highlights the large redundancies that are inherent in this approach.

2.1.5 Discussion

This Section has explored three definitions of type-2 fuzzy sets. Both the Mendel and John definition and the vertical slice definition give type-2 fuzzy sets as 3-dimensional entities. The Mendel and John definition is a simple extension of type-1 fuzzy sets. Each point in the domain of the set has a membership grade between zero and one, this membership grade then has an associated degree of possibility, also measured as a number between zero and one. So, the Mendel and John definition allows a single point in the domain of a type-2 fuzzy set to have a number of associated membership grades. The vertical slice definition maps each point in the domain of a type-2 fuzzy set to a single secondary membership function, a type-1 fuzzy number bounded by the interval $[0,1]$. These two definitions give two notational methods for describing exactly the same thing. Both are equally valid and the decision to use one or the other should be based on notational convenience. In this thesis the vertical slice definition is used as it provides the simplest notation when defining type-2 fuzzy logic operations.

The representation theorem is quite separate from the other definitions. The representation theorem models each type-2 fuzzy set as a collection of simpler embedded type-2 fuzzy sets. This deconstruction of a type-2 set into a number of constituent parts allows operations to be defined using type-1 fuzzy methods. Such definitions are easy to follow and have the advantage of making type-2 fuzzy logic widely accessible to type-1 fuzzy practitioners. The major drawback with the representation theorem is the large number of embedded type-2 sets required to model a type-2 fuzzy set. The following example illustrates this point. Consider the two type-2 fuzzy sets $\widetilde{A}$ and

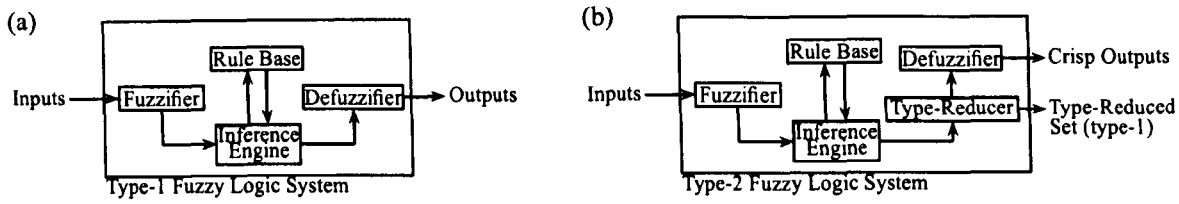


Fig. 2.5. (a) A Type-1 Fuzzy Logic System. (b) A Type-2 Fuzzy Logic System.

$\tilde{B}$. $\tilde{A}$ has 8 discrete points in the primary domain, with the domain of each secondary membership function having 5 discrete points. $\tilde{B}$ has 7 discrete points in the primary domain, with the domain of each secondary membership function having 6 discrete points. The number of embedded type-2 fuzzy sets required to model $\tilde{A}$ and $\tilde{B}$ are 390625 and 279936, respectively. To find $\tilde{A}$ *and* $\tilde{B}$ requires 1.0935×10^{11} type-1 fuzzy 'and' operations to be performed. This example demonstrates that the representation theorem is rarely useful in a practical context due to the large number of embedded sets required to model a type-2 fuzzy set, nevertheless, they are useful theoretically.

This Section has given a mathematical description of a type-2 fuzzy set. The following Section describes type-2 fuzzy logic systems. Such systems allow logical inferences to be made using type-2 fuzzy sets.

2.2 Type-2 Fuzzy Logic Systems

Having arrived at a set of definitions for type-2 fuzzy sets the logical operations needed to make decisions based on such sets will now be explored. Throughout this thesis only Mamdani (1974, 1977) fuzzy systems are discussed. The geometric methods discussed in the next Chapter are only defined for Mamdani type systems. Rules in a Mamdani system are direct extensions of crisp production rules with fuzzy sets forming the antecedent and consequent. The other widely reported type of fuzzy system is the TSK model (Takagi & Sugeno 1985, Sugeno & Kang 1988). TSK systems utilise rules that have functions in the consequent, not fuzzy sets. Currently, only Mamdani rules have a geometric interpretation and that is why only this type of fuzzy system is discussed here. For a discussion on type-2 TSK based systems see John (2000) and Mendel (2001*b*).

The architecture of a type-2 fuzzy system is a direct extension of a type-1 fuzzy system. A type-1 fuzzy system consists of four components: fuzzifier, inference engine, rule base and defuzzifier, see Figure 2.5(a). A type-2 fuzzy logic system has an additional component, the type reducer, see Figure 2.5(b). The rules in a type-2 system (Karnik & Mendel 1998*a*) take the same IF-THEN form as type-1 rules. Type-2 fuzzy rules have type-2 fuzzy sets in the antecedent and consequent. A type-2 fuzzy rule R^i maps n inputs of the input space $X_1 \times X_2 \times \ldots \times X_n$ to the output space Y,

i.e.,

$$R^i = \text{IF } x_1 \text{ is } \widetilde{F_1^i} \text{ and } x_2 \text{ is } \widetilde{F_2^i} \text{ and } \ldots x_n \text{ is } \widetilde{F_n^i} \text{ THEN } y \text{ is } \widetilde{G}^i$$

From Karnik and Mendel (1998*a*)

where $x_1 \ldots x_n$ are crisp inputs to the system, y is the crisp output from the system, $\widetilde{F_1^i}$ to $\widetilde{F_1^n}$ the antecedent type-2 fuzzy sets of the rule and $\widetilde{G}^i$ is the type-2 fuzzy consequent set of the rule. Any of the 'and' operators in the rule R^i could be replaced with an 'or' operator. The 'and' operation is performed by the meet ($\sqcap$) operation and the 'or' operation by the join ($\sqcup$). The implication operation is generally taken to be performed by the meet operation although other operations could be used. For example, Karnik and Mendel (1998*b*) suggest that a scaling operation could used as an alternative. Whether implication uses the meet or a scaling operation, the result of a type-2 fuzzy inferencing process is still a type-2 fuzzy set. Most applications require a crisp output to be derived from this resultant type-2 fuzzy set. This is because the decision being taken based on the result of fuzzy system are usually crisp in nature, turn right, turn left, class A, class B, etc. To arrive at a crisp output the type-2 fuzzy set must be type-reduced before it can be defuzzified. Before discussing the inferencing operations, the method used to define these operations, the extension principle, is given.

2.2.1 The Extension Principle

The extension principle was introduced by Zadeh (1975*a*,1975*b*,1975*c*) in the same set of papers that type-2 fuzzy sets were first presented. Initially all the logical operations for type-2 fuzzy sets were defined using the extension principle. The representation theorem also makes it possible to define these operations from the type-1 fuzzy equivalents.

Definition 2.5 *The extension principle for fuzzy sets is in essence a basic identity which allows the domain of the definition of a mapping or a relation to be extended from points in U to fuzzy subsets of U. More specifically, suppose that f is a mapping from U to V, and A is a fuzzy subset of U expressed as*

$$A = \mu_1 u_1 + \ldots + \mu_n u_n. \tag{2.10}$$

Then the extension principle asserts that

$$f(A) = f(\mu_1 u_1 + \ldots + \mu_n u_n) \equiv (\mu_1 f(u_1) + \ldots + \mu_n f(u_n)) \tag{2.11}$$

Thus, the image of A under f can be deduced from the knowledge of the images of $u_1, \ldots, u_2$ *under f.*

The extension principle allows any function defined for a crisp set to be extended for a type-1 fuzzy set. Furthermore, the extension principle allows any function defined for a type-1 set to be extended for a type-2 fuzzy set. The join and meet operations were derived by Zadeh with the use of the extension principle. This Section has discussed the methods used to define the operations used in type-2 fuzzy inferencing. Each of the inferencing stages of a type-2 fuzzy logic system will now be discussed in turn.

2.2.2 Fuzzification

Fuzzification is the process of finding the membership grade of a given input in a given fuzzy set. For discrete systems this is a trivial process which takes a crisp input value and returns a secondary membership function. For the continuous case either linear interpolation or function application is used to calculate the value of a membership grade. The membership grade of the inputs must be taken for every type-2 fuzzy set in the antecedent of every rule in the rule base. Once these membership grades, which are secondary membership functions, have been found they can be logically connected to form the rule antecedent values.

2.2.3 Combination of Antecedents

The secondary membership functions given by the fuzzifier have to be logically combined to form rule antecedents. The logical connectives, the 'or' and 'and' of type-2 fuzzy sets were given by Zadeh (1975*b*). Mizumoto and Tanaka (1976) renamed these operations the 'join' and 'meet' and were the first to look at the properties of these operations. Mizumoto and Tanaka(1981), along with Dubois and Prade (1980), also discuss the use of different t-norm and t-conorm operators. Karnik and Mendel(2001*b*) defined more computational efficient methods for calculating the join and meet of secondary membership functions that are normal and convex. A fuzzy membership function is said to be normal is it contains at least one point with a value of 1. A fuzzy membership function is said to be convex if it only contains one apex point, that is one peak.

The join ($\sqcup$) operation finds the conjunction of two secondary membership functions $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$. Let $\mu_{\tilde{A}}(x) = \sum_{i=1}^{M} \alpha_i / v_i$ and let $\mu_{\tilde{B}}(x) = \sum_{j=1}^{N} \beta_j / w_j$. The conjunction of $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$ is given by

$$\mu_{\tilde{A} \sqcup \tilde{B}}(x) = \sum_{i=1}^{M} \sum_{j=1}^{N} (\alpha_i \star \beta_j) / (v_i \vee w_j) \tag{2.12}$$

Adapted from Mizumoto and Tanaka (1981)

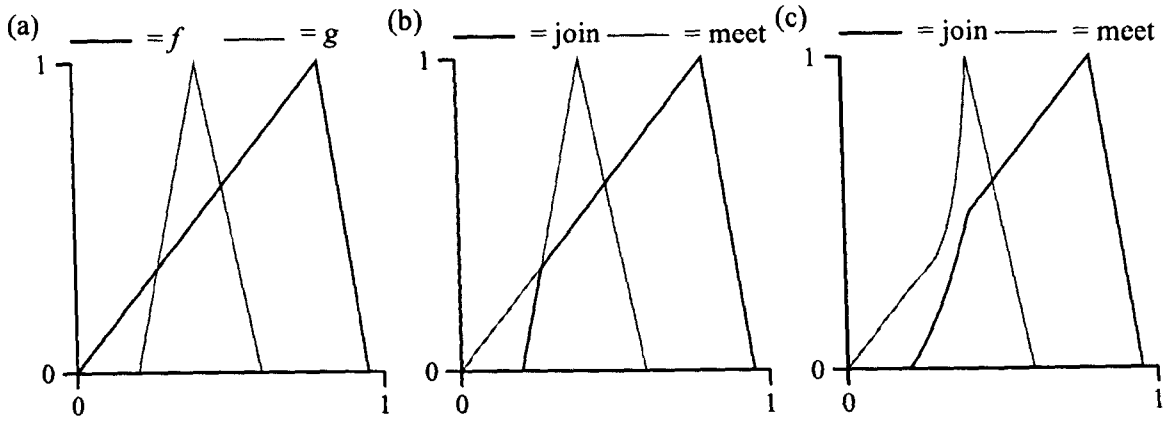


Fig. 2.6. (a) The Secondary Membership Functions f and g. (b) The Join and Meet of f and g Under the Minimum T-norm. (c) The Join and Meet of f and g under the Product T-norm.

where $\vee$ is the t-conorm, generally taken to be maximum and $\star$ is a t-norm such as minimum or product.

The meet ($\sqcap$) operation finds the disjunction of two secondary membership functions $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$. The disjunction of $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$ is given by

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x) = \sum_{i=1}^{M}\sum_{j=1}^{N}(\alpha_i \star \beta_j)/(v_i \star w_j) \tag{2.13}$$

Adapted from Mizumoto and Tanaka (1981)

where again $\star$ is a t-norm.

The join and meet operations perform differently under product and minimum t-norms. Consider the two secondary membership functions f and g depicted in Figure 2.6(a). The join and meet of f and g under the minimum t-norm is depicted in Figure 2.6(b). The join and meet of f and g under the product t-norm is depicted in Figure 2.6(c). Observe in Figure 2.6(c) that the resultant secondary membership functions are curved. If the product t-norm is used when performing the join and meet on piecewise-linear function, the resultant functions may not be piecewise linear. This characteristic is part of the reason why the geometric model of join and meet given in the following Chapter do not make use of the product t-norm. A geometric interpretation of the product t-norm is more complex than that of the minimum and as such has yet to be developed.

Throughout this thesis the computational overheads that come with type-2 fuzzy logic are discussed. The origins of some of these issues can now be identified. The basic logical operations are far more complex than the type-1 equivalent. Assume that t-norms and t-conorms require an equal amount of processing resource t, then cost of finding the 'and' or 'or' of two discrete type-1 fuzzy sets at a point x is t. The cost of finding the 'and' or 'or' of two discrete type-2 fuzzy sets at

a point x is $2MNt$, where M and N are the number of discrete points in the domain of the respective secondary memberships. Karnik and Mendel (2001*b*) give methods to reduce this overhead significantly. These methods rely on the secondary membership functions being both normal and convex. If the condition of normality and convexity are not met then incorrect results are produced. These optimisations are achieved by defining the join and meet under both minimum and product t-norm for a single point in the domain of the resultant secondary membership functions. The optimised join and meet operations (Karnik & Mendel 2001*b*) are now given. Let there be n convex, normal, type-1 real fuzzy sets $F_1,\ldots,F_n$ characterized by membership functions $f_1,\ldots,f_n$, respectively. Let $v_1, v_2,\ldots,v_n$ be real numbers such that $v_1 \leq v_2 \leq \ldots v_n$ and $f_1(v_1) = f_2(v_2) = \ldots = f_n(v_n) = 1$. Then restricting the t-conorm to maximum ($\vee$) and the t-norm to minimum ($\wedge$) gives,

$$\mu_{\sqcup_{i=1}^{n} F_i}(\theta) = \begin{cases} \wedge_{i=1}^{n} f_i(\theta), & \theta < v_1, \\ \wedge_{i=k+1}^{n} f_i(\theta), & v_k \leq \theta < v_{k+1}, 1 \leq k \leq n-1, \\ \vee_{i=1}^{n} f_i(\theta), & \theta \geq v_n \end{cases} \tag{2.14}$$

and

$$\mu_{\sqcap_{i=1}^{n} F_i}(\theta) = \begin{cases} \vee_{i=1}^{n} f_i(\theta), & \theta < v_1, \\ \wedge_{i=k+1}^{n} f_i(\theta), & v_k \leq \theta < v_{k+1}, 1 \leq k \leq n-1, \\ \wedge_{i=1}^{n} f_i(\theta), & \theta \geq v_n \end{cases} \tag{2.15}$$

From Karnik and Mendel (2001*b*)

These definitions significantly reduce the computational cost of the join and meet operations. The reduction in computational cost is achieved by reducing the number of times the domain of the sets have to be traversed to one. Prior to this definition the number of times that the domain of each secondary membership function had to be traversed was equal to the number of points in other secondary membership function it is being combined with. Novel extensions to the Karnik and Mendel optimisations which reduce the limitations of these operations are explored in Section 4.1. This work underpins the geometric definition of join and meet given in the next Chapter.

The join for a single point under the product t-norm can be expressed as

$$\mu_{\sqcup_{i=1}^{n} F_i}(\theta) = \begin{cases} \prod_{i=1}^{n} f_i(\theta), & \theta < v_1, \\ \prod_{i=k+1}^{n} f_i(\theta), & v_k \leq \theta < v_{k+1}, 1 \leq k \leq n-1 \\ \vee_{i=1}^{n} f_i(\theta), & \theta \geq v_k \end{cases} \tag{2.16}$$

From Karnik and Mendel (2001*b*)

The meet under the product t-norm can also be expressed for a single point. However, this requires a supremum operation, needing iterations over several terms and does not result in the same reduction in computation obtained for the other operations. Let f_1 and f_2 be two normal and convex

secondary membership functions. Let θ, v and w be points in the domain of $f_1 \sqcap f_2$. To find the membership of grade of θ in $f_1 \sqcap f_2$:

- find all the pairs $\{v, w\}$ such that $vw = \theta$;
- take the product of grades at v and w. Where there is more than one pair that gives the value θ take the pair whose membership grade gives the higher product value, and
- the possible pairs $\{v, w\}$ that can give the value θ are $\{v, \theta/v\}$ or $\{\theta/w, w\}$ for $\theta \neq 0$ and $\{v, 0\}$ or $\{0, w\}$ for $\theta = 0$.

This gives the following:

$$\mu_{f_1 \sqcap f_2}(\theta) = \begin{cases} \sup_{v \in [0,1], v \neq 0} f_1(v) f_2(\frac{\theta}{v}) & \theta \neq 0 \\ f_1(0) \vee f_2(0) & \theta = 0 \end{cases} \tag{2.17}$$

From Karnik and Mendel (2001*b*)

where *sup* is the supremum operation. The join and meet operations given in this Section are the basic logical building blocks that are used extensively in the other stages of the type-2 fuzzy inferencing process. The join and meet give methods for logically combining the antecedent values in type-2 fuzzy rules. These antecedent values can then be used when calculating the value of a rule consequent, i.e. when performing the type-2 fuzzy implication operation.

2.2.4 Implication

Prior to the implication stage of the inferencing process the antecedent value of each rule has been calculated. The implication of the antecedent to the rule consequent of each rule must now be found. This is done by finding the meet of the antecedent with every point in the consequent type-2 fuzzy set. Let the antecedent value be $\mu_{\tilde{A}}(x_1) \sqcap \mu_{\tilde{B}}(x_2)$ and the consequent be $\tilde{C}$ over the domain Y. The value of $\mu_{\tilde{A}}(x_1) \sqcap \mu_{\tilde{B}}(x_2) \Rightarrow \tilde{C}$ is given below.

$$\mu_{\tilde{A}}(x_1) \sqcap \mu_{\tilde{B}}(x_2) \Rightarrow \tilde{C} = \int_{y \in Y} \left(\mu_{\tilde{A}}(x_1) \sqcap \mu_{\tilde{B}}(x_2) \right) \sqcap \mu_{\tilde{C}}(y) \tag{2.18}$$

This Section has given the type-2 fuzzy implication operation, based on the meet operation. Once the value of each rule consequent has been found these consequent sets can be combined to give a final inferred type-2 fuzzy set.

2.2.5 Combination of Consequent Sets

Each rule will produce an inferred consequent type-2 fuzzy set. To combine these sets the 'or' operation, the join operation, is applied at every point in the domain of the sets. Let consequent sets be $\widetilde{C_1}, \widetilde{C_2}, \ldots, \widetilde{C_n}$ and the final combined set be $\widetilde{F}$, all over domain X. The value of $\widetilde{F}$ is given below.

$$\widetilde{F} = \int_{x \in X} \bigsqcup_{i=1}^{n} \widetilde{C}_i(x) \tag{2.19}$$

This operation results in a single type-2 fuzzy set which represents the decision of the fuzzy logic system. A crisp output must now be derived which is representative of this final fuzzy set.

2.2.6 Output Processing

Most fuzzy logic systems deployed in the real world will be required to use crisp numbers for inputs and to express the output decisions as crisp numbers. This requires methods for taking a fuzzy set and finding a crisp value that is representative of that set. In type-2 fuzzy logic systems this is currently performed by the type-reducer and defuzzifier during the output processing stage. Type-reduction (Karnik & Mendel 1998*b*, Karnik & Mendel 2001*a*) is a method that allows for the computation of a type-1 fuzzy set that gives an aggregated representation of the possible centroids of a type-2 fuzzy set. Here the generalised centroid (GC) is given as this will be used for comparative purposes in following chapters.

The generalised centroid (Karnik & Mendel 2001*a*) finds every possible embedded type-2 fuzzy set that could represent the type-2 fuzzy set. A pairing is then made between the centroid of each possible embedded set and the uncertainty associated with that embedded set resulting in a type-1 fuzzy set. This type-reduced (from type-2 to type-1) fuzzy set gives a distribution of the possible centroids of the type-2 fuzzy set and the uncertainty associated with each of them. This type-1 fuzzy set can then be defuzzified to give a crisp answer.

Definition 2.6 *The generalized centroid (GC) gives a possiblistic distribution of the centroids of a type-2 fuzzy set. Let $\widetilde{A}$ be a discrete type-2 fuzzy set with L discrete points in its domain. Let n be the number of embedded type-2 sets required to represent $\widetilde{A}$ using the representation theorem as given by equation 2.8. The generalised centroid of $\widetilde{A}$ may be given as*

$$GC_{\widetilde{A}} = \sum_{i=1}^{n} [\bigstar_{j=1}^{L} \mu_{\widetilde{A}_e^i}(x_j, u_j)] \Big/ \frac{\sum_{j=1}^{L} x_j u_j}{\sum_{j=1}^{L} x_j} \tag{2.20}$$

where $\mu_{\widetilde{A}_e^i}(x_j, u_j)$, x_j and u_j follow from the definition of a type-2 embedded set given in equation

2.8 and $\bigstar_{j=1}^{L} \mu_{\tilde{A}_e^i}(x_j, u_j)$ *is the t-norm of all values of* $\mu_{\tilde{A}_e^i}(x_j, u_j)$ *from* 1 *to L.*

Adapted from Karnik and Mendel (2001*a*)

Clearly the computational cost of the type-reduction operation is very large. Karnik and Mendel (2001*a*) noted the potential for concurrent computation of all the embedded sets simultaneously. However, parallel processing capacity is neither inexpensive nor widely available. Karnik and Mendel also give a set of methods that approximate type-reduction for certain classes of set, although these have not been widely taken up. Greenfield *et al* (2005) proposed the sampling defuzzification method as a way of reducing the computational complexity of type-reduction. The sampling defuzzifier selects a predetermined number of embedded sets at random and uses this subset of the total number of embedded sets to calculate the centroid. The sampling method consistently gives results that are close to the actual centroid whilst massively reducing the amount of computation needed to find the centroid of a type-2 fuzzy set. The sampling method is to date the only approach which attempts to reduce the computational complexity of the type-reduction of a generalised type-2 fuzzy set. The novel geometric approach presented in the following Chapter also reduces the computational complexity of type-reduction. The significant decrease in the computational cost of type-reduction that results from the novel geometric approach is one of the main achievements of this thesis. Unlike the sampling method, the geometric approach is not an approximation of the centroid but a geometric interpretation of the centroid. During the development of the sampling method, the author of this thesis has provided input to Greenfield's research leading to joint authorship of Greenfield *et al* (2005), with a journal article on this currently in preparation.

The following Section presents some novel work by the author which extends the Karnik and Mendel optimised join and meet operations. These novel extensions widen the applicability of these operations to include both normal and, for the first time, non-normal secondary membership functions.

Type-2 fuzzy logic provides a method of encapsulating uncertainty into computerised, mathematical systems. The computational overheads inherent in type-2 methods make fast decision making very difficult. Type-2 interval fuzzy logic systems provide a useful compromise between the speed of type-1 fuzzy systems and the modelling power of type-2 fuzzy systems. The following Section discusses an efficient and widely adopted subset of type-2 fuzzy logic, type-2 interval fuzzy logic.

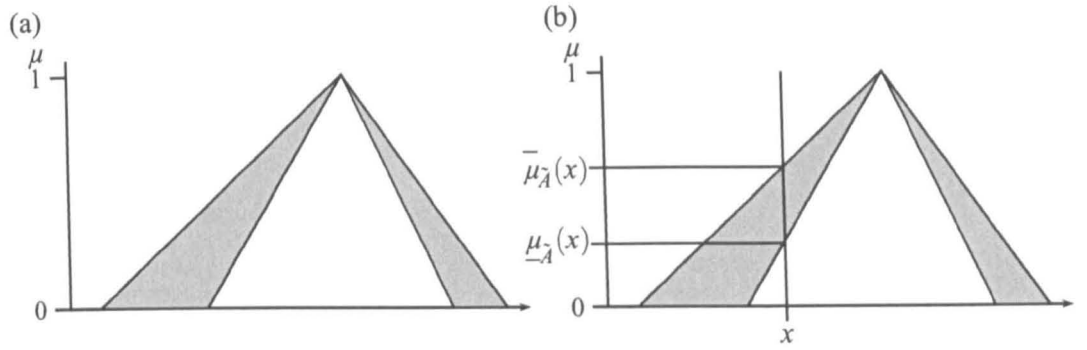


Fig. 2.7. (a) The Type-2 Interval Fuzzy Set $\tilde{A}$. (b) The Membership Grade of a Point x in $\tilde{A}$.

2.3 Type-2 Interval Fuzzy Logic

Type-2 interval fuzzy systems were first presented by Zadeh (1975b) under the term interval-valued fuzzy sets. Zadeh's original definition relied on the concept of an α-cut. The definition given here is an adaptation of the definition of a type-2 fuzzy set given in definition 2.2.

Definition 2.7 *A type-2 interval fuzzy set, denoted $\tilde{A}$, is characterised by a type-2 interval membership function $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$, i.e.,*

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \mid x \in X, u \in J_x \subseteq [0,1]\} \tag{2.21}$$

in which $\mu_{\tilde{A}}(x,u) \in \{0,1\}$, X is the domain of the fuzzy set and J_x is the domain of the secondary membership function at x.

Such a set $\tilde{A}$ is depicted in Figure 2.7(a). Since all points in each of the secondary membership functions are at unity, each secondary can be represented as an interval set, hence the name type-2 interval set. An interval set A contains two elements a lower bound denoted $\underline{A}$ and an upper bound denoted $\overline{A}$, so that $A = [\underline{A}, \overline{A}]$. All points between these bounds are implicit elements of the set.

By restricting the secondary membership functions to only being interval sets a great deal of the computational redundancy in type-2 methods can be eliminated. Essentially, the third dimension of a type-2 interval fuzzy can be ignored for computational purposes as all points in that dimension have an equal value. Liang and Mendel(2000b) proposed the notion of representing a type-2 interval set as two type-1 membership functions, specifically the upper and lower bounds of the footprint of uncertainty of that set.

Definition 2.8 *The Footprint of Uncertainty (FOU) (Liang & Mendel 2000b) of a type-2 fuzzy set consists of a bounded region that is the union of all the primary membership grades in that set.*

Let $\widetilde{A}$ be a type-2 fuzzy set over X. The FOU of $\widetilde{A}$ is given below:

$$FOU(\widetilde{A}) = \sum_{x \in X} (x, J_x) \tag{2.22}$$

where J_x is the domain of the secondary membership function at x.

From Mendel and John (2002)

Mendel and John noted that the FOU is a useful concept since it gives an easily understood two-dimensional description of the spacial distribution and the level of the uncertainty captured by a general type-2 fuzzy set. For a type-2 interval set this notion can be extended since the *FOU* of such a set gives a complete description of that set. Liang and Mendel exploit this by representing the region covered by the *FOU* with two type-2 membership functions, an upper and a lower bound i.e.,

$$\widetilde{A} = \int_{x \in X} \left(\int_{u \in [\underline{\mu}_{\widetilde{A}(x)}, \overline{\mu}_{\widetilde{A}(x)}]} 1/u \right) \tag{2.23}$$

where $\underline{\mu}_{\widetilde{A}(x)}$ and $\overline{\mu}_{\widetilde{A}(x)}$ denote the value of x in the lower and upper membership functions respectively. Figure 2.7(b) depicts the membership grade of such a set $\widetilde{A}$ and a point x.

The logical operations for inferencing can also be given in terms of upper and lower membership functions. The meet of two type-2 interval fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ at a point x is given below.

$$\mu_{\widetilde{A} \sqcap \widetilde{B}}(x) = \int_{u \in [\underline{\mu}_{\widetilde{A}(x)} \star \underline{\mu}_{\widetilde{B}(x)}, \overline{\mu}_{\widetilde{A}(x)} \star \overline{\mu}_{\widetilde{B}(x)}]} 1/u \tag{2.24}$$

From Liang and Mendel (2000*b*)

Where $\star$ is the product or minimum t-norm. The join can be given in a similar way, as below.

$$\mu_{\widetilde{A} \sqcup \widetilde{B}}(x) = \int_{u \in [\underline{\mu}_{\widetilde{A}(x)} \vee \underline{\mu}_{\widetilde{B}(x)}, \overline{\mu}_{\widetilde{A}(x)} \vee \overline{\mu}_{\widetilde{B}(x)}]} 1/u \tag{2.25}$$

From Liang and Mendel (2000*b*)

where $\vee$ is a t-conorm, generally taken to be the maximum. These operations are depicted in Figure 2.8(b) and (c).

This Section has shown how the efficient representations of type-2 interval fuzzy sets can eliminate massive redundancies in the logical operations. Karnik and Mendel's work, and the novel work presented in Section 4.1, show how this can also be done for general type-2 logic, albeit with the imposition of some restrictions. Unlike generalised type-2 fuzzy sets, an efficient solution has been found to the problem of type-reducing a type-2 interval fuzzy set. The iterative method

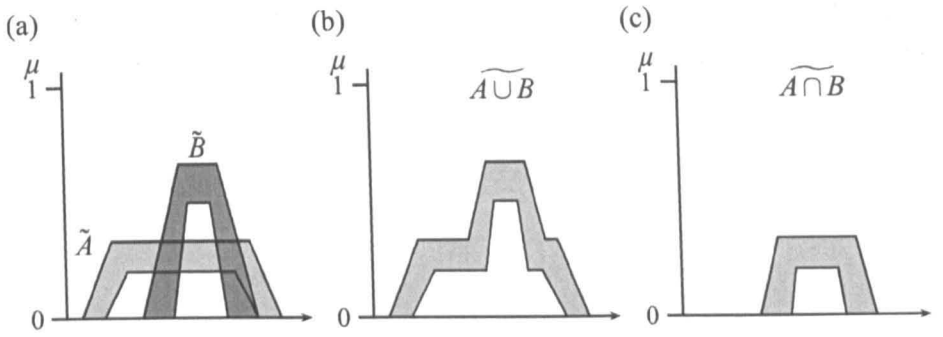


Fig. 2.8. (a) The Type-2 Interval Fuzzy Sets $\tilde{A}$ and $\tilde{B}$. (b) The Type-2 Interval Fuzzy Set $\widetilde{A \cap B}$. (c) The Type-2 Interval Fuzzy Set $\widetilde{A \cup B}$.

provides a highly efficient way of finding the centroid of a type-2 interval set and the uncertainty propagated through system that is associated with that set. The iterative method is discussed in the next Section.

2.3.1 Type-Reduction: The Iterative Method

The Karnik-Mendel (2001*a*) iterative method provides a computational efficient way of type-reducing type-2 interval fuzzy sets. Type reducing a type-2 interval fuzzy set $\widetilde{C}$ produces an interval set C. This set is the distribution of the possible centroids of the set $\widetilde{C}$. The midpoint of C gives the final defuzzified value of the set. The endpoints of the set $[C_l, C_r]$ give the range of possible points where the centroid of C could lie taking into account the uncertainty in the system (Karnik & Mendel 1998*c*). The iterative method provides an optimised method for finding these endpoints.

In this thesis only the centroid type-reducer is discussed. This is because this is the only defuzzifier with a direct geometric interpretation and as such, will be the focus of later discussion on type-reduction. The generalised centroid of a type-2 interval fuzzy set $\widetilde{A}$ over the domain X is given below.

$$GC_{\widetilde{A}} = \int_{\theta_1 \in J_{x_1}} \cdots \int_{\theta_1 \in J_{x_N}} 1 \Big/ \frac{\sum_{i=1}^{N} x_i \theta_i}{\sum_{i=1}^{N} \theta_i} = [C_l, C_r] \tag{2.26}$$

From Mendel (2001*a*)

where J_{x_N} is the secondary membership grade at N in the secondary membership function J_x and $x \in X$. The type-reduced set C only needs two endpoints to define it, C_l and C_r. Each of these points come from the centroid values of a set that is embedded in $\widetilde{A}$. The iterative method exploits the properties of the centroid operation to find these two sets with a relatively low amount of computational effort.

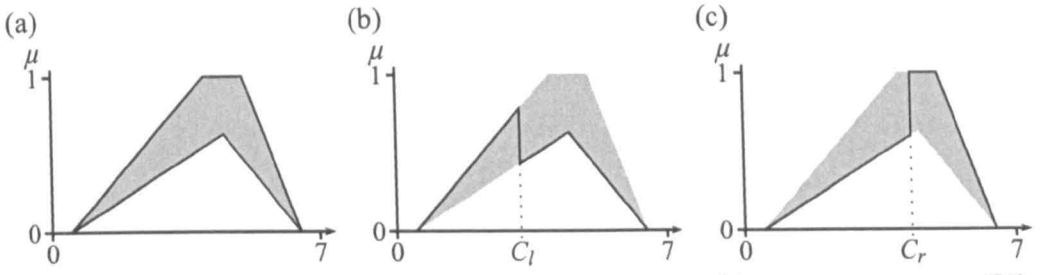


Fig. 2.9. (a) The Type-2 Interval Fuzzy Set $\widetilde{A}$. (b) The Embedded Set $\widetilde{A_e^{C_l}}$. (c) The Embedded Set $\widetilde{A_e^{C_r}}$.

Consider the type-2 interval fuzzy set $\widetilde{A}$ depicted in Figure 2.8(a). For each point in the domain of $\widetilde{A}$ there are a range of values that each embedded set can take. The type-reduced interval will be given by the sets with the highest and the lowest centroids. The centroid is in essence a weighted area. As such, the set with the lowest centroid will have as much area as possible towards the left and as little as possible on the right. Karnik and Mendel found that such a set takes all its points from the upper bound up to a crossover point and then takes all its points from the lower bound. The task of finding the set with the lowest centroid then becomes the problem of finding this crossover point. The procedure for doing this, the iterative method, is given below.

1. Set the embedded set $\widetilde{A}_e^i$ to take all points from the lower bound.
2. Find the domain point p that is closest to the centroid e of $\widetilde{A}_e^i$.
3. Assign a value of $\overline{\mu_{\widetilde{A}}}(x)$ for all values of x in $\widetilde{A}_e^i$ that are less than or equal to p. Assign a value of $\underline{\mu_{\widetilde{A}}}(x)$ for all values of x in $\widetilde{A}_e^i$ that are greater than p.
4. Find the domain point p' that is closest to the centroid e of $\widetilde{A}_e^i$.
5. Repeat steps 2-4 until $p = p'$.

The final value of e gives the lower point in C, C_l. If there are L discrete points in the domain of $\widetilde{A}_e^i$ then this procedure can take at most L iterations. Figure 2.8 (b) depicts the embedded set $\widetilde{A}_e^{C_l}$, the centroid of which gives the value C_l.

Finding the maximum value C_r uses a similar procedure. Step 1 takes all points from the upper bound. Step 3 sets all points up to p from the lower bound and all points after and including p from the upper bound. Figure 2.8 (c) depicts the embedded set $\widetilde{A}_e^{C_r}$, the centroid of which gives the value C_r. The two procedures may be run in parallel. Each will converge in at most L steps, although in most cases convergence will be achieved before this number of iterations, usually only taking two to three iterations. This represents a massive improvement in computational speed over the generalised centroid.

The iterative method (Karnik & Mendel 2001*a*) provides a fast and efficient means of finding the centroid of a type-2 interval fuzzy set. Such efficiencies have not been obtained for type-reducing type-2 fuzzy sets. Some authors (Wu & Mendel 2001, Wu & Tan 2005) have focused on further increases in computational speed using mathematical approximations or evolving type-reducers using genetic algorithms. Melgarejo *et al* (2004*a*, 2004*b*) have been working towards a full hardware implementation of a type-2 interval fuzzy system. These theoretical advances have for the first time allowed type-2 interval fuzzy logic to be used in control applications. This represents a significant step forward for type-2 technologies, giving applicability to an area where type-1 methods have proved very successful, control systems. This is one of the main areas fuelling the current burst in type-2 fuzzy research. However, a majority of the recent publications, particularly applications led research, has been limited to type-2 interval systems only. To date no control applications have been reported that use full type-2 methods. The motivation behind this thesis is to provide computational efficient method for type-2 fuzzy inferencing so that applications led research can take advantage of the richer model of uncertainty that is offered by generalised type-2 fuzzy logic.

This Section has given a technical description of the Karnik-Mendel iterative method and discussed the impact this method has had on the development and adoption of type-2 fuzzy logic. The following Section broadens this discussion to all aspects of type-2 interval fuzzy logic.

2.3.2 Discussion

Type-2 interval fuzzy logic systems are a halfway house, a compromise between the modelling capabilities of type-2 fuzzy logic and the applicability of type-1 systems. The operations are efficient, with few redundancies. The iterative method offers the ability to calculate the defuzzified value of a set with a comparatively small computational effort. For these reasons interval technologies are currently filling the void left by the computational problems associated with generalised type-2 fuzzy logic. The increased modelling power of interval over type-1 systems has been practically demonstrated in several applications, for examples see (Mendel 2001*a*, Liang & Mendel 2001, Hagras 2004, Figueroa *et al* 2005). In this thesis type-2 interval systems are viewed as a compromise. Type-2 fuzzy sets model the vagueness of a particular set membership. Type-2 interval fuzzy sets require that this vagueness is represented with a uniform distribution. Effectively, type-2 interval fuzzy sets model set membership grades as crisp sets rather than crisp values. It remains to be seen to what degree, if at all, a type-2 system can outperform an interval system. Certainly the type-2 set model provides a much richer model of uncertainty. Common sense suggests that the better the model of uncertainty, the better a system will perform in an uncertain environment. The later chapters of this work attempt to answer this question by comparing the

relative performance of a generalised type-2 system against a type-1 and an interval system.

This Section has discussed type-2 interval fuzzy logic. This form of fuzzy logic provides a greater model of uncertainty than type-1 methods without the associated computational cost of type-2 methods. Interval systems do not however have the same capability for modelling uncertainty. The next Section discusses the historical development of type-2 fuzzy logic.

2.4 The Historical Development of Type-2 Fuzzy Logic

Type-2 fuzzy logic is a growing research topic. Figure 2.10 illustrates the growth in academic activity in the field since its inception. In this Section the main themes reported in literature are discussed in a historical context. The aim of this Section is to illustrate the position of this thesis in the field of type-2 fuzzy logic. Figure 2.11 depicts a time line showing the historical development of type-2 methods. Each block on the time line relates to a paragraph presented in this Section.

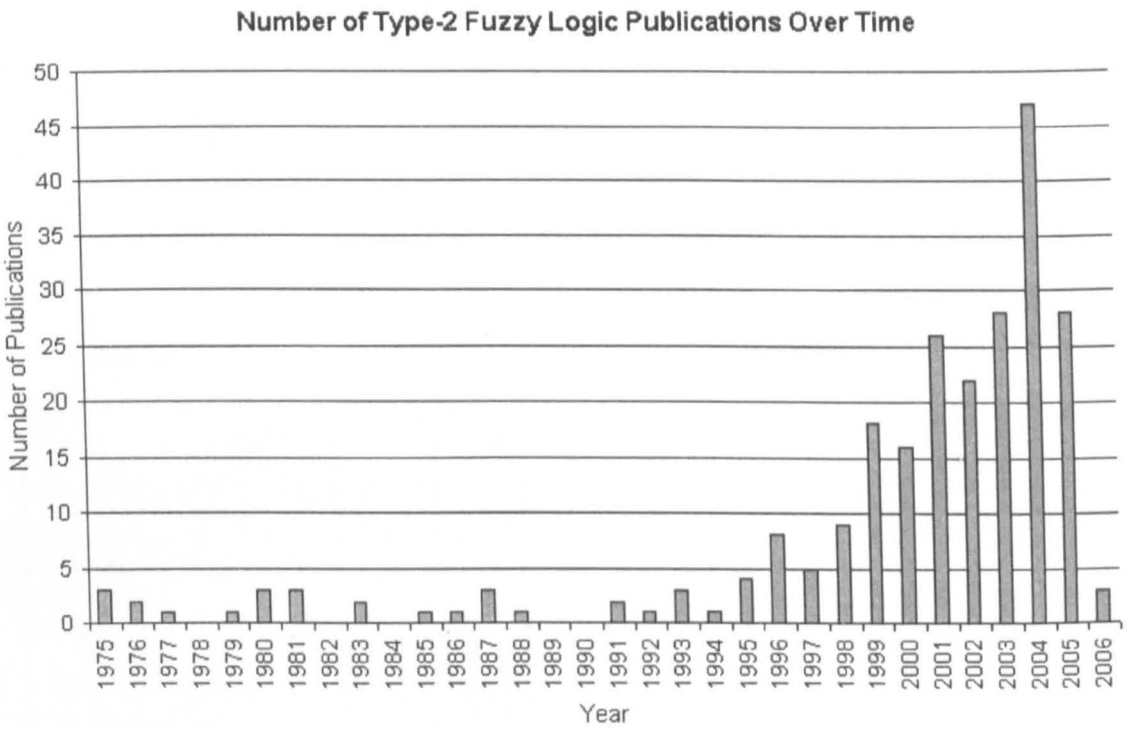


Fig. 2.10. The Number of Type-2 Related Publications Over Time. Source: www.type2fuzzylogic.org/publications/ accessed 01/03/2006.

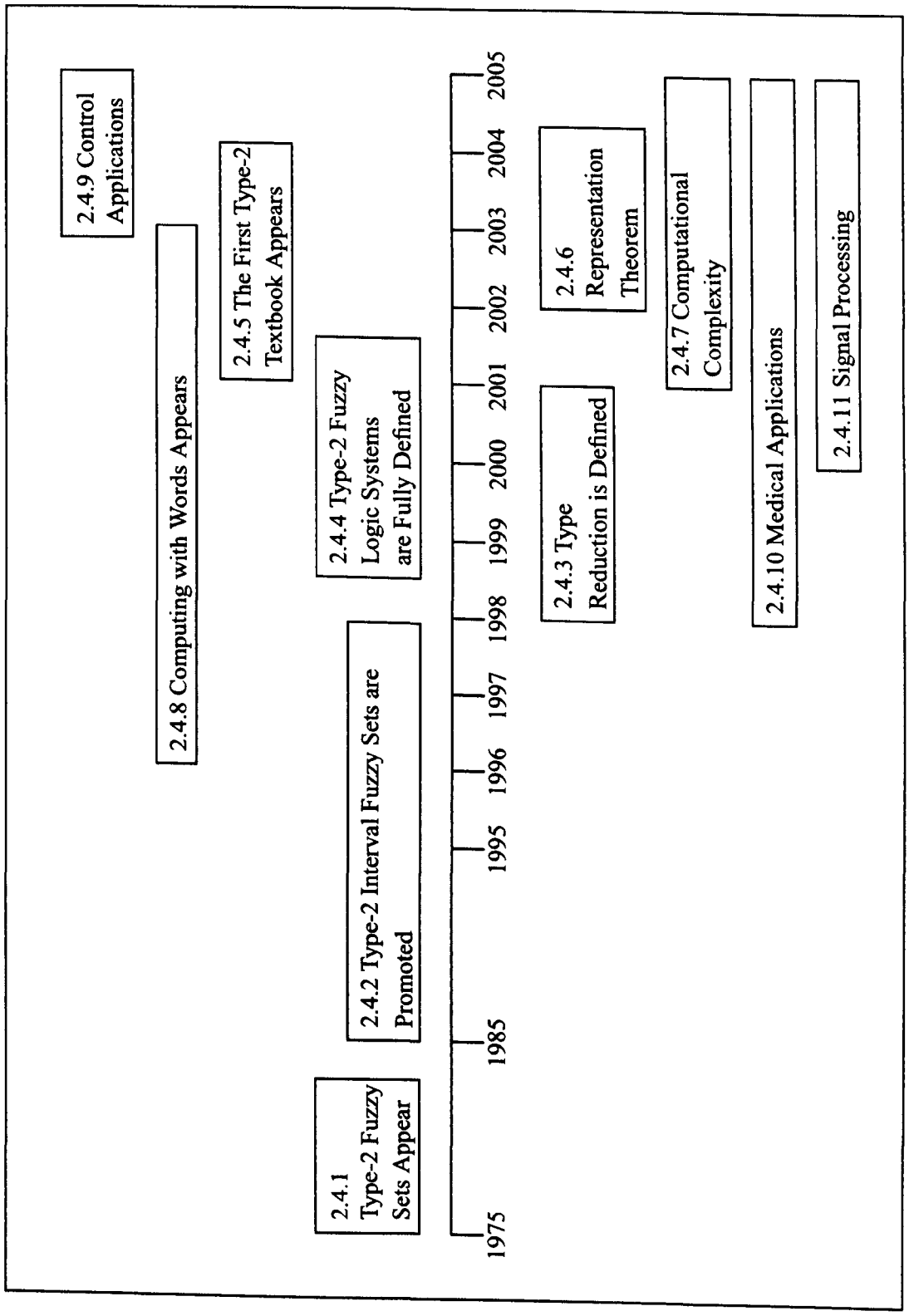


Fig. 2.11. A Time Line Depicting the Historical Development of Type-2 Fuzzy Logic.

2.4.1 Type-2 Fuzzy Sets Appear

Type-2 fuzzy sets were first defined and discussed in a trilogy of papers by Zadeh (1975*a*, 1975*b*, 1975*c*). These papers concentrated on the notion of a fuzzy set where the memberships grades of a fuzzy set are measured with linguistic terms such as *low*, *medium* and *high*. Logical connectives for such sets were also given, although the terms join and meet were not used. Zadeh only explored the use of the minimum and maximum operators t-norm and t-conorm when investigating the logical operations. Mizumoto and Tanaka (1976,1981) and Dubois and Prade (1980) both studied the logical connectives of what became known as secondary membership functions. Mizumoto and Tanaka were the first to use the terms join and meet for these logical connectives. Both Dubois and Prade and Mizumoto and Tanaka studied the join and meet under a variety of t-norm and t-conorm operators.

2.4.2 Type-2 Interval Fuzzy Sets are Promoted

Turksen (1993*a*, 1993*b*, 1995), Schwartz (1985) and Klir and Folger (1988) promoted the use of type-2 fuzzy sets, at that time called interval valued or IV fuzzy sets. Schwartz believes that type-2 interval fuzzy sets should be employed when the linguistic uncertainty of a term cannot be sufficiently modelled by the type-1 methods. Klir and Folger advocate the use of IV fuzzy sets when the membership functions of type-1 fuzzy sets could not be agreed upon. These arguments were explored in greater detail by Mendel (1999). Turksen put forward a collection of logical connectives for type-2 interval fuzzy sets noting that the expressive power of type-2 fuzzy reasoning lies in the ability to retain the uncertainty throughout the inferencing process.

2.4.3 Type-reduction is Defined

Karnik and Mendel (1998*a*, 1998*c*, 2001*a*) defined type-reduction by applying the extension principle to a variety of type-1 defuzzifiers. The notion of an output processing stage of a type-2 fuzzy system was developed in these papers.

2.4.4 Type-2 Fuzzy Logic Systems are Fully Defined

Karnik and Mendel (1998*a*, 2001*a*) gave a complete description of the fuzzy inferencing process. This allowed work on the application of type-2 fuzzy logic to proceed. Around this time John published a series of review papers (1998*a*, 1998*b*, 1999*a*, 1999*b*) on type-2 fuzzy systems. Early applications of the technology also began to appear (John 1996, John, Innocent & Barnes 1998, Karnik & Mendel 1999). As Figure 2.10 demonstrates, the recent growth in type-2 fuzzy publications began around this time.

2.4.5 The First Textbook on the Subject of Type-2 Fuzzy Logic Appears

Following the consolidation of the definitions and existing literature by John and Karnik and Mendel, the field was opened up to a wider potential audience with the publication of the first type-2 textbook. Uncertain Rule-Based Fuzzy Logic System: Introduction and New Directions was written by Mendel and published in 2001 (Mendel 2001*b*). This textbook references a great deal of the work on type-2 fuzzy logic that had been published to date, bringing together many of Mendel's earlier publications.

2.4.6 The Representation Theorem is Defined

Mendel and John (2002) gave the representation theorem of type-2 fuzzy sets. By representing a type-2 fuzzy set as a collection of simpler type-2 embedded sets it is possible to define operations of type-2 fuzzy sets without the use of the extension principle. The motivation behind this work was that by eliminating the need to learn about the extension principle, the field would be more accessible to type-1 fuzzy practitioners. However, the representation theorem has its own learning curve, and is not significantly simpler to understand than the extension principle. One of the outcomes of the representation theorem has been the definition of arithmetic operators for type-2 fuzzy numbers (Coupland & John 2003).

2.4.7 Issues of Computational Complexity Begin to be Explored

The complexity of join and meet operations and type-reduction of a type-2 fuzzy set limit the applicability of type-2 methods. Although type-2 interval sets are simpler, type-reduction is still a problem, due to inherent complexity and redundancies. The iterative method (Karnik & Mendel 2001*a*) and the Wu-Mendel (2001,2002) approximation were developed to make the type-reduction of type-2 interval fuzzy sets more efficient. This has led to the majority of the publications in the field of type-2 only discussing type-2 interval methods. Indeed, many authors refer to type-2 interval fuzzy set as type-2 fuzzy sets and add the qualifying term 'generalised' when discussing actual type-2 fuzzy sets. The computational problems of join and meet were effectively resolved by Karnik and Mendel (2001*a*), with the novel work given in Section 4.1 extending that work to include non-normal secondaries. This work is also discussed by the author, along with some aspects of the geometric approach in (Coupland & John 2004*a*, Coupland & John 2004*b*, Coupland & John 2005*b*, Coupland & John 2006*b*, Coupland & John 2005*a*). Greenfield *et al* (2005) give an efficient method for approximating the the type-reduced set of a type-2 fuzzy set using a stochastic approach. The work presented in this thesis clearly falls into this area of type-2 research, although the approach being taken here is wholly different to any reported in the literature.

2.4.8 Computing with Words Appears

Zadeh (1996, 1999) made the claim that fuzzy logic, approximately at least, equates to computing with words (CWW). In CWW numbers are replaced with words not only when reasoning, but also when solving calculations. Zadeh's examples use fuzzy granules to model words. A fuzzy granule is actually the FOU of a type-2 interval fuzzy set. Both Mendel (2001*a*, 2003) and Turksen (2002) point out that CWW requires type-2 fuzzy sets, both opting to use the simpler type-2 interval representations. Mendel (1999) re-emphasised this point by demonstrating that human models of words as obtained through a survey require at least interval representations. Again, the author's opinion is that type-2 fuzzy logic does not constitute computing with words. Type-2 fuzzy sets can model concepts that are uncertain. Linguistic terms are uncertain but also may contain other characteristics, such as contextual dependence, that fuzzy logic bears no relation to.

2.4.9 Control Applications

With the iterative method and the Wu-Mendel approximation allowing fast execution of type-2 fuzzy systems, control applications began to emerge. Melin and Castillo (2003, 2004) used type-2 interval systems in the context of plant control. Hagras (2004) demonstrated that a type-2 interval fuzzy logic controller could outperform a type-1 fuzzy controller under large uncertainties. Wu and Tan (2004) applied type-2 interval systems to the control of a complex multi-variable liquid level process. Figueroa *et al* (2005) used a type-2 interval control for non-autonomous robots in the context of a robot football game. The author has performed a comprehensive study of both general and type-2 interval fuzzy controllers for an autonomous mobile robot. These studies are presented in Chapter 5 of this thesis and in Coupland (2006*a*) and in a journal article that is currently being prepared. Doctor *et al* (2004,2005) used a type-2 interval system to model and adapt to the behaviour of people in an intelligent dormitory room. Lynch *et al* (2005) are continuing to build a type-2 interval control system for large marine diesel engines. Melgarejo *et al* (2004) have developed a limited hardware implementation of a type-2 interval controller.

2.4.10 Medical Applications

Medical applications are one of the few areas where a generalised type-2 fuzzy logic has been used in preference to type-2 interval fuzzy logic. This is largely because such systems do not require fast execution times but do contain large uncertainties. John *et al* (1998, 2000) used a type-2 fuzzy system for the pre-processing of tibia radiographic images. Garibaldi *et al* (1997, 2003) have done extensive work on assessing the health of a new born baby using knowledge of acid-base balance in the blood from the umbilical cord. Innocent and John (2004) proposed the

use of fuzzy cognitive maps to aid the differential diagnosis of confusable diseases and suggest that type-2 cognitive maps may yield improved results. Di Lascio *et al* (2005) also used type-2 fuzzy sets to model differential diagnosis of diseases, modelling the compatibility of the symptom to a disease as a linguistic term. John *et al* (2001*a*,2001*b*) used type-2 fuzzy sets to model the perception of clinical opinions of nursing staff as linguistic terms.

2.4.11 Signal Processing

Signal processing, like control, has to date only used type-2 interval methods. Liang and Mendel (2000*a*) implemented a fuzzy adaptive filter for the equalization of non-linear time-varying channels. Mitchell (2005) defined a similarity measure for use with type-2 fuzzy sets which was used in a radiographic image classifier. Karnik and Mendel (1999) used a type-2 interval system to predict the next value in a chaotic time series. Musikasuwan *et al* (2004) investigated how the learning capabilities of type-1 and type-2 interval systems differ according to the number of learning parameters used. Both systems were designed to to predict a Mackey-Glass time series.

2.4.12 Summary

This Section has given the major developments that have taken place in the field of type-2 fuzzy logic and places them in a historical context. This helps to highlight the motivation of this thesis and show how the work presented here contributes to the field of type-2 fuzzy logic in a timely fashion. Type-2 literature has become predominately concerned with type-2 interval methods. The likely reason for this is the elimination of the computational problems for type-2 interval methods. This thesis seeks to overcome the computational problems of type-2 fuzzy logic so that the advantages associated with these methods can be fully exploited by practitioners.

2.5 Discussion

This Chapter has demonstrated how type-2 fuzzy sets model uncertainty as vagueness about a piece of knowledge. The degree to which any element belongs to a type-2 fuzzy set can be vague, as described by a fuzzy number. This is the means by which type-2 fuzzy sets capture and model uncertainty. Type-2 fuzzy logic retains this model of uncertainty throughout the inferencing process. The logical operators, join and meet, preserve this uncertainty when finding the conjunction and disjunction of two type-2 fuzzy sets and when finding the result of an implication operation. Type-reduction, extended from defuzzification, also uses the uncertainty captured by a type-2 fuzzy set. Type-reduction aggregates the uncertainty, finding the centroid of every possible type-1

fuzzy set embedded within the type-2 set and calculating the possibility of that centroid. This process results in a type-reduced set, a type-1 fuzzy number that gives a vague measure of the centroid.

Type-2 interval methods also model uncertainty as a vagueness of knowledge, but with a much lower computational cost. Type-2 interval methods are limited in that it is only possible to model the vagueness of a piece of knowledge with a uniform distribution. This approach may reduce the modelling capability of type-2 interval fuzzy sets, but it also allows for computationally efficient representations and logical operations. The conjunction and disjunction of two type-2 interval fuzzy set is far simpler to calculate than the generalised type-2 equivalent. The type-reduction process is also less complex than that of generalised type-2 fuzzy sets. The iterative method gives an efficient way of calculating the type-reduced interval, essentially by performing an optimised search for the endpoints of the type-reduced set. The iterative method, although not guaranteed to, often converges in only two or three iterations. With these methods a type-2 interval fuzzy system can have a far lower computational cost than a type-2 fuzzy system and not a great deal larger than a type-1 fuzzy system. Type-1 fuzzy sets however, cannot model uncertainty. The degree to which an element belongs to a type-1 fuzzy set is crisp, the inferencing process from fuzzification to defuzzification is crisp. Type-1 fuzzy sets capture and model vagueness but cannot model uncertainty. This should mean that both type-2 interval and type-2 fuzzy system outperform the equivalent type-1 fuzzy system under uncertain conditions.

The ability of type-2 fuzzy logic to model uncertainty is being exploited in a growing number of applications. To date, all applications that have required fast execution, such as control or signal processing have only utilised type-2 interval methods. This is a significant limitation. Although the type-2 logical operators have been made more efficient, type-reduction still remains a significant barrier to the adoption of type-2 methods. Novel methods, techniques or models that significantly increase in the execution speed of type-2 fuzzy systems are required. When type-2 interval systems have been compared to equivalent type-1 systems, the type-2 interval systems often give a better performance, particularly when uncertainties are present. Since generalised type-2 fuzzy sets offer an improved model of uncertainty it is reasonable to expect that a type-2 fuzzy logic system will show the ability to outperform both type-1 and type-2 interval fuzzy system under large uncertainties. This point will be explored in Chapter 5 of this thesis.

The following Chapter describes the novel geometric approach to fuzzy logic. The geometric approach to fuzzy logic forms the theoretical core of this thesis. Applying geometric models and methods to fuzzy logic results in significant performance differences with discrete systems. In the case of type-2 fuzzy logic, significant improvements can be made for execution speed.

Chapter 3

Type-1 Geometric Fuzzy Logic Systems

This Chapter presents the novel geometric approach to fuzzy logic systems. The geometric approach models fuzzy sets as geometric objects. Algorithms, selected from the field of computational geometry, are used to define the logical operations of these geometric fuzzy sets.

The vast majority of applications of fuzzy logic use a rule base based approach, referred to as a fuzzy logic system (FLS). A FLS uses fuzzy logic in combination with a set of fuzzy rules to arrive at a conclusion based on one or more premise. A FLS usually consists of an input processor, a rule base, an inference engine and an output processor. Inputs are fed into the input processor or fuzzifier, the inference engine using the fuzzy rule base reasons with the fuzzified inputs, a final fuzzy set is arrived at before an output is calculated by the output processor and fed out from the system. Figure 3.1 depicts a type-*n* FLS where the defuzzifier could use further output processing, including type-reduction and may output measures of uncertainty in conjunction with crisp numbers. Fuzzy systems are usually computerised, residing in software or on specialised

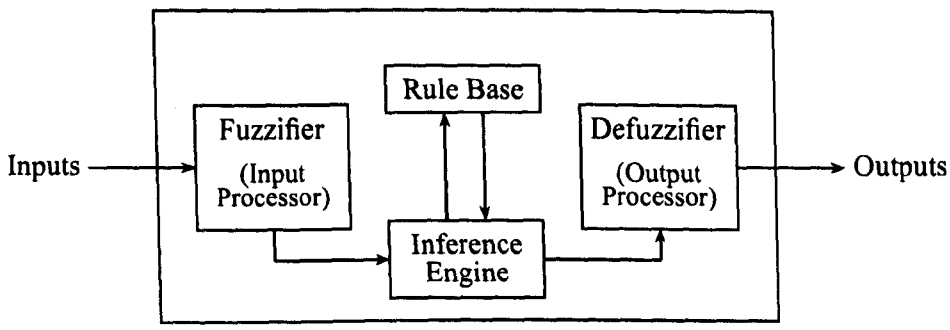


Fig. 3.1. A Type-*n* Fuzzy Logic System.

hardware. This work concentrates on Mamdani (1977) fuzzy logic as this type of fuzzy logic is a direct extension of crisp production rules and is widely used in the literature. Mamdani systems are simple to model geometrically as the main components of such systems are fuzzy sets, *t*-norms and *t*-conorms. This Chapter defines geometric interpretations of these three system components,

making it possible to define the novel geometric fuzzy logic operations. The rules in a Mamdani fuzzy system use fuzzy sets in their antecedent and their consequent. The consequent of each rule in the rule base must be either be aggregated or combined with a disjunction operation. This Chapter discusses the techniques and methods required to model Mamdani fuzzy logic systems of type-1, type-2 interval and type-2 geometrically. These geometric systems offer an alternative approach to fuzzy logic which some system developers may find advantageous. This Chapter describes the reasons why geometric fuzzy systems are limited to only utilising the minimum t-norm and maximum t-conorm, which will be a problem for some applications. It is demonstrated that the geometric approach offers an increase in accuracy of fuzzy set models over discrete systems, giving more efficient representations. Geometric interval defuzzification is shown to have a predictable level of computation, as opposed to the iterative type reduction method which only offers a predictable upper limit of computation. Other advantages and disadvantages of the geometric approach will be discussed in this Chapter.

This Chapter is subdivided into five main sections. Section 3.1 describes the small amount of previous work by other authors that relates directly to this Chapter. Section 3.2 gives geometric techniques, equations and algorithms that will be useful throughout the remainder of this Chapter. Much of the work in this Section is well understood including long established techniques from computational geometry although some of the 3-dimensional techniques are at the cutting edge of this field. Section 3.3 gives a complete geometric model of a type-1 FLS. This model offers an alternative technique for implementing type-1 fuzzy systems. It is shown that type-1 geometric fuzzy systems can be more accurate than discrete systems, but do suffer from increased complexity. This Section also lays the foundations for the next two sections. Section 4.2 presents a complete geometric model of an interval type-2 FLS. The work presented in this Section is a direct extension of the geometric type-1 FLS. This Section also presents a geometric alternative to type-reduction for geometric interval fuzzy sets. Section 4.3 presents a complete geometric model of a type-2 FLS. This model utilises the 3-dimensional work presented in Section 3.2 to extend the geometric models presented in sections 3.3 and 4.2 to give a complete geometric model of type-2 fuzzy logic. This Section extends the geometric defuzzifier for a type-2 fuzzy set. This has a significant impact on the computational cost of type-2 FLS. Section 4.4 discusses the potential of type-2 FLS that are hybrids of discrete and geometric systems in maximising the performance of type-2 FLS. Techniques for transforming between the two models are given. The final part of this Chapter, Section 4.5 discusses and summarises the work presented in this Chapter and describes the impact of this work on the field of type-2 fuzzy logic.

3.1 Piecewise Linear Functions

This Section describes other work that has taken a geometric approach to modelling fuzzy sets or fuzzy inferencing. The use of piecewise linear membership functions along with approaches to reasoning with fuzzy sets based on such functions and the impact that such models have on the accuracy of fuzzy systems are all discussed. The later sections of this Chapter describe novel work that gives a complete geometric description of type-1, interval and type-2 fuzzy logic. The work presented in the Section underpins the novel work later in this Chapter.

Piecewise linear functions are widely used when modelling membership functions in fuzzy systems. This type of function is defined below.

Definition 3.1 *A Piecewise linear function (PLF) is a series of ordered vertices that are connected by line segments to form a function over a continuous domain. This function is linear in all but a finite set of points. For a PLF to model a fuzzy set A over the domain X the y component of all the vertices must in the interval* $[0,1]$ *i.e.,*

$$\mu_A : X \rightarrow [0,1] \tag{3.1}$$

The membership grade μ_A *for any particular value of x is given by*

$$\mu_A(x) = \begin{cases} 0 & ; x \leq x_1 \text{ or } x_n \leq x \\ y_i & ; x = x_i \\ y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) & ; x_i < x < x_{i+1} \end{cases} \tag{3.2}$$

where x_0 *and* x_n *are, respectively, the x-component of the first and last vertices of A. For convenience a PLF will also be denoted by a set of vertices, i.e.,*

$$A = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n) \mid x_i \in X, y_i \in [0,1], x_i < x_{i+1}, \forall i\} \tag{3.3}$$

Where x_i *is the x component or domain value of the* i^{th} *vertex and* y_i *is the y component or range value of the* i^{th} *vertex.*

The type-1 fuzzy set A is depicted in Figure 3.2. The membership function of A is a piecewise linear function consisting of the vertices (x_0,y_0), (x_1,y_1), (x_2,y_2) and (x_3,y_3). Like discrete membership functions PLF cannot model a curved function with complete accuracy. Most systems only use PLF during the fuzzification inferencing stage after which the functions are discretised to allow inferencing to be performed. The goal of the geometric fuzzy systems presented in this Chapter is to remove the need for discretisation. The consequences of discretisation are now discussed.

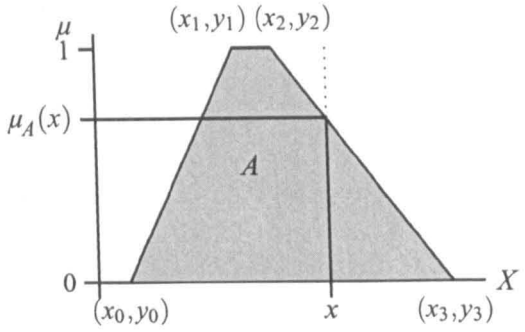


Fig. 3.2. The Fuzzy Set A Represented by a Piecewise Linear Membership Function.

3.1.1 The Effect of Discretisation

Discretisation is the process which takes a fuzzy set over a continuous domain and places it on a discrete domain. Whenever a (non-geometric) fuzzy system is deployed on computer hardware or software, the fuzzy sets used in that system must be discretised. A discrete fuzzy set comprises of a set of points from the domain of a fuzzy set, each with a respective membership grade. These pairings are usually ordered by domain point value and are generally equally spaced across the domain, although this is not a formal requirement. The accuracy of such sets in modelling a given function is dependent on the number of points in the set and the level of approximation given by these points. The system developer decides how the fuzzy sets in a particular system are discretised as to fit the hardware and performance specifications of the system. Many fuzzy sets are defined over a continuous domain. Whenever such a fuzzy set is discretised information about the membership function of that fuzzy set is lost. Examples given in this Chapter demonstrate that, for piecewise linear membership functions, the geometric approach eliminates this loss of information.

Two typical discretisation options are now explored, demonstrating how the accuracy of fuzzy systems can be effected by discretisation. The intersection of the sets in the following examples is performed using the minimum t-norm and the centroid values are calculated with the centre of area defuzzifier. One approach would be to discretise the domain of a fuzzy set into a given constant number of points. Figure 3.3(a) depicts the intersection of two sets discretised in such a way using 20 discrete points. Another method of discretisation is to ensure the discrete points in the set separated by a predefined distance from a predetermined origin. The intersection of two such sets is depicted in Figure 3.3(b) where the seperation distance between each discrete point is 0.5 and the origin is 0. Whichever method is chosen discretisation will have an impact on the accuracy of the set model. Figure 3.3(c) depicts the intersection of the two original membership functions that have been discretised in Figures 3.3 (a) and (b). The resultant sets depicted in Figures 3.3 (a) and (b) contain errors caused purely by discretising the sets. To quantify this error in these

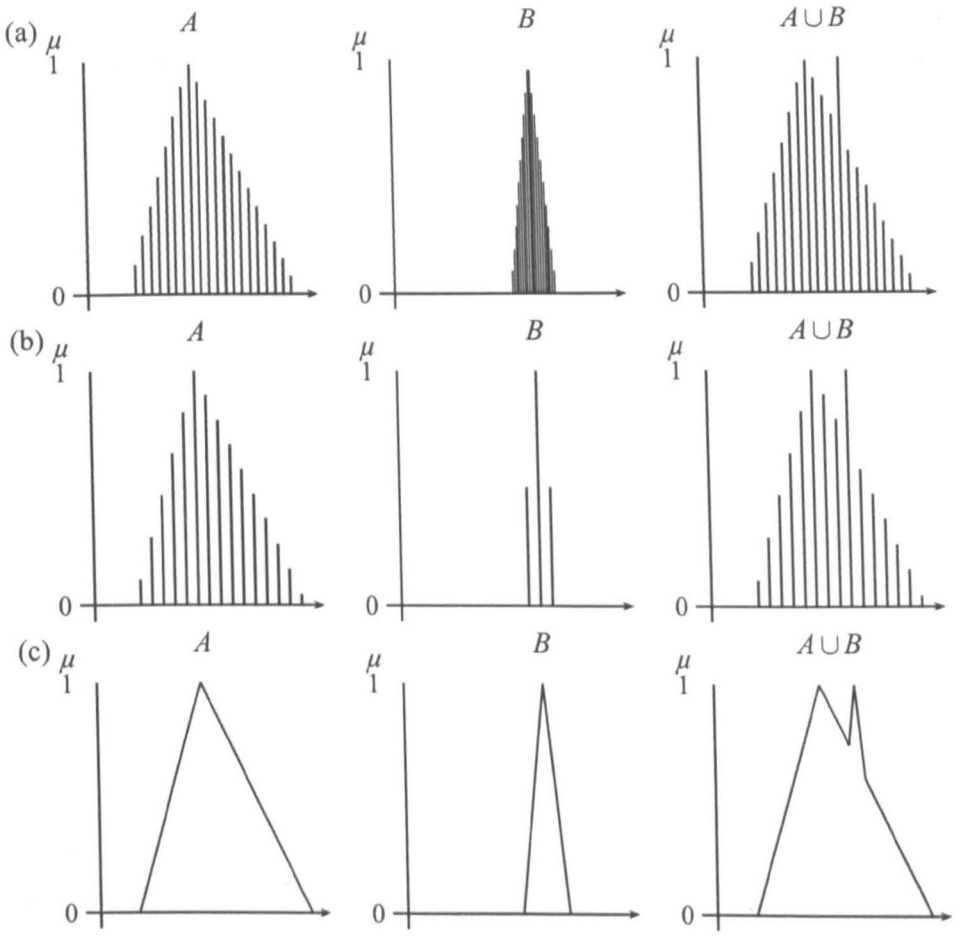


Fig. 3.3. The Effect of Discretisation of a Fuzzy Set on Accuracy of the Set Model.

examples the centroid of the resultant PLF fuzzy set depicted in Figures 3.3 (b) is compared to the centroid of the other two resultant sets. The centroids are given in Table 3.1. The percentage error is calculated as a percentage of the support of the fuzzy set $A/capB$ depicted in Figure 3.3 (c).

The differences in the errors for the discretisation methods do not reflect the accuracy of either method. These results only hold for this particular example. The errors shown serve to illustrate the point that the discretisation process causes errors in the fuzzy inferencing process which can be avoided with the use of piecewise linear membership functions. Furthermore these errors could be compounded through subsequent inferencing. These examples only show one simple rule. Over

Set	Centroid	Error	% Error
Resultant Set from Figure 3.3(a)	0.5113	0.004562	0.88%
Resultant Set from Figure 3.3(b)	0.491756	0.024106	4.7%
Resultant Set from Figure 3.3(c)	0.515862	0	0%

Table 3.1. A Comparison of The Centroid Values of Three Fuzzy Sets using Different Discretisation Methods.

an entire rule base these errors, the loss of information about the fuzzy sets, would compound and therefore increase. One of the motivating factors for proposing geometric fuzzy systems is that such errors are eliminated from the inferencing process. PLF do however have some modelling limitations that are common to discrete fuzzy set models. Neither of these models can capture a curved function with total accuracy. However both discrete and PLF membership function can approximate any curve with the same level of accuracy. The following Section describes previous work producing fuzzy systems that use PLF during the inferencing process.

3.1.2 Fuzzy Reasoning with Continuous Membership Functions

Previous work on reasoning with PLF has been based on implementations in functional programming languages. Van den Broek (1997, 1999) reported the implementation of various fuzzy implication operators in the functional programming language Miranda. In that work PLF were used to give the membership functions of the fuzzy sets. Each linear portion of a membership function can then be manipulated as a linear function. The code used to implement the implication operators exploits functional programming techniques and so does not relate directly to the algorithmic solutions presented later on in this Chapter although analogies may be drawn. Van den Broek represents each PLF as an ordered list of 2-dimensional points. When performing a Mamdani style implication one PLF A models the normalised A, a second PLF B models the antecedent value across the [0,1] interval. All the points where A and B intersect are found prior to resolving the main function. The method for finding these intersection points exploits the functional technique of lazy evaluation (Thompson 1999) and so has no algorithmic equivalent. The main function is resolved by recursively pattern matching portions of A with the intersection points with B to find the minimum of A and B. This technique can be viewed as a functional implementation of the type-1 implication operator presented later in this Chapter.

The major drawback with Van den Broek's work is the reliance on functional programming. The techniques used to implement the fuzzy systems are based around a computational method for which there is no current hardware implementation. The deployment of the fuzzy methods suggested by Van den Broek would require significant hardware resources, most probably a full personal computer. This may be the reason why Van den Broek only defines a limited subset of fuzzy operations and has not proposed a complete fuzzy system. Van den Broek's approach to modelling fuzzy operations as the manipulation of linear functions is important. The novel geometric approach presented in this Chapter is based on a similar notion of defining fuzzy operations as the manipulation of geometric figures. Such manipulations are defined algorithmically and can therefore be implemented on standard hardware. The (2-dimensional) geometric approach still manipulates piecewise linear functions, but uses algorithmic geometric operations to perform

them. In this sense the geometric approach extends Van den Broek's work by manipulating PLF as geometric figures rather than linear functions.

This Section has shown that the membership functions of fuzzy sets can be approximated by a piecewise linear functions. A method for fuzzifying this type of membership function has been given. It has been shown that this limited use of these models can improve the accuracy of a fuzzy logic system over a discrete system. The following Section describes the fundamental geometric methods required to extend the use of PLF to the entire inferencing process, thereby improving accuracy. These geometric techniques underpin the definitions of geometric fuzzy logic given later in this Chapter.

3.2 Geometry Primer

This Section gives the necessary mathematical grounding needed for geometric fuzzy logic systems to be defined. This Section begins with 2-dimensional operations from geometry and computer graphics, moving on to more complicated 3-dimensional methods. The techniques presented here are fundamental to geometric fuzzy logic systems. The need for these techniques was identified from initially working with fuzzy sets with piecewise linear functions. Algorithms to geometrically manipulate piecewise linear functions identified in the computational geometry literature were then exploited to define geometric fuzzy logic.

3.2.1 Geometric Interpretations of T-norms and T-conorms

Fuzzy logic makes extensive use of the t-norm and t-conorm operators. This Section describes how two of the operators minimum and maximum have direct geometric interpretation. The product and bounded sum operators are shown to not yet have a direct geometric interpretation.

Consider the two linear functions a and b depicted in Figure 3.4(a). The functions a and b can be easily represented in a geometric model as line segments. Geometric fuzzy operations on these line segments will then apply t-norms and t-conorms to these line segments. Figure 3.4(b) depicts the application of the minimum and product t-norms to the functions a and b. Figure 3.4(c) depicts the application of the maximum and bounded sum t-conorms to the functions a and b. The result from the product t-norm is a curve and can therefore only be approximated by a series line segments. The bounded sum of a and b could be given by a number of line segments, in this case two. However the properties of these line segments would be reasonably complicated to derive. The minimum and maximum of a and b can always be given by one or two line segments. The properties of these line segments are given by the vertices of the original line segment and if they

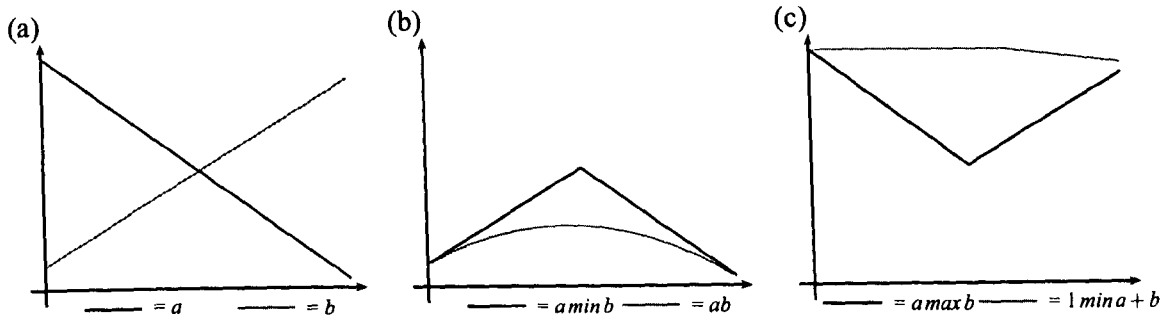


Fig. 3.4. (a) Two Linear Functions a and b. (b) T-norms of a and b. (c) T-conorms of a and b.

intersect, the point where they intersect. To summarise:

- the product t-norm, as yet, has no geometric representation;
- it may be possible to represent bounded sum geometrically, and
- minimum and maximum both have simple geometric interpretations.

The most widely used operators are minimum, maximum and product. Although the product has no current geometric interpretation, both minimum and maximum can be employed by geometric fuzzy logic systems. Throughout the remainder of this thesis the geometric approach will be limited to using the minimum t-norm and the maximum t-conorm. The following Section describes the parametric description of a line which is used to find the points where two line segments intersect.

3.2.2 The Parametric Description of A 2-Dimensional Line

The parametric equation of a straight line is useful for finding points where lines intersect. The work given here underpins the clipping techniques presented later on in this Chapter.

Any point (x_α, y_α) along a line L with respect to parameter t is given by the two equations below.

$$x_\alpha = x_1 + t(x_2 - x_1) \tag{3.4}$$

$$y_\alpha = y_1 + t(y_2 - y_1) \tag{3.5}$$

Where the start point of L is (x_1, y_1) and the end point is (x_2, y_2). If $0 \leq t \leq 1$ then the point (x_α, y_α) not only lies on the line L but lies on the segment $(x_1, y_1) - (x_2, y_2)$. An example line L is depicted in Figure 3.5(a).

Consider the two lines and segments L_a, from (x_1, y_1) to (x_2, y_2) and L_b, from (x_3, y_3) to (x_4, y_4) depicted in Figure 3.5(b). It can be verified whether these two intersect or not by testing whether

there exists a point (x_α, y_α) which satisfies the parametric equation of both of the lines. Let t_a and t_b be parameters on the respective lines L_a and L_b that give the point (x_α, y_α) such that

$$x_\alpha = x_1 + t_a(x_2 - x_1) = x_3 + t_b(x_4 - x_3) \tag{3.6}$$

$$y_\alpha = y_1 + t_a(y_2 - y_1) = y_3 + t_b(y_4 - y_3) \tag{3.7}$$

These simultaneous equations can be solved with respect to t_a and t_b as given below.

$$t_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \tag{3.8}$$

$$t_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \tag{3.9}$$

From Bourke (1989)

If $0 \leq t_a \leq 1$ and $0 \leq t_b \leq 1$ then (x_α, y_α) is given by equations 3.6 and 3.7. Bourke (1989) noted that when the denominator of the two equations is zero then the lines must be parallel and when both the numerator and the denominator are zero the lines are coincident. Therefore equations 3.8 and 3.9 can be used to test whether two given line segments intersect. Once a value for t has been calculated equations 3.6 and 3.7 can then be used to give the position of this intersecting point. Calculating the point where two line segments intersect allows one line segment to be clipped against another. These geometric clipping operations give geometric interpretations of t-norm and t-conorm operations discussed in the previous sections. Clipping is the process of the removing portions geometric objects that overlapped or lie outside a given boundary. In graphical processes it is essential to be able to clip one polygon against another, for example if, in a scene, a tree is in front a house the polygons associated with the house must be clipped to the polygons associated with the tree. This process is sometimes called hidden surface removal. In terms of fuzzy logic, if the tree and house were fuzzy sets then the removed hidden surface represents the fuzzy 'and' operation. Figures 3.6 (a), (b) and (c) depict this analogy. The following Section describes efficient techniques for finding the intersection points amongst a collection of line segments.

3.2.3 The Bentley-Ottmann Plane Sweep Algorithm

Geometric clipping operations are used to define geometric fuzzy operations later in this Chapter. When performing such an operation it is often necessary to test for intersection points amongst a large collection of line segments. The simplest solution, the brute force method, simply tests whether each line segment in one polygon intersects with any of the line segments in the collection. If there are a total of N line segments with k intersections then the execution cost will be $O(N^2)$. In the interests of performance a more efficient method was devised, the Bentley-Ottmann (1979)

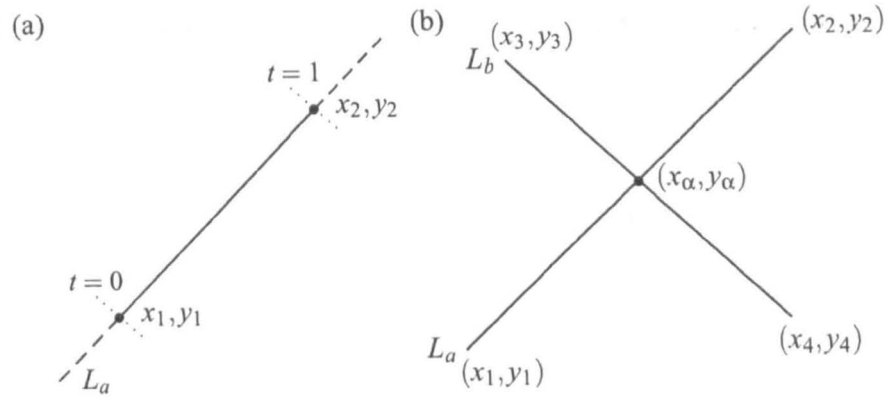


Fig. 3.5. The Intersections Points of Two Parametric Lines. (a) The Parametric Description of a Line (b) Finding the Intersection Point of Two Lines.

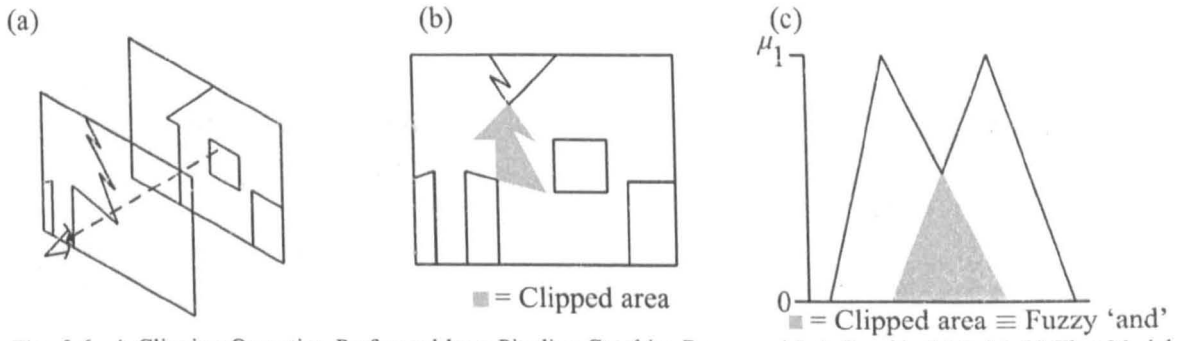


Fig. 3.6. A Clipping Operation Performed by a Pipeline Graphics Process. (a) A Graphical Model (b) That Model Clipped and Rasterised (c) The Fuzzy Equivalent

plane sweep algorithm has an execution cost of $O(N \log N + klogN)$. In essence the plane sweep algorithm moves a virtual vertical line across the x-dimension only testing for line intersections between lines whose start point has been past but whose end point has not. All points to the left of the line have been solved and the points immediately to the right of the line are about to be solved. The name *plane sweep* comes from the notion of a plane sweeping across the segments checking for intersection points immediately in front of it.

Consider the collection of line segments S that is depicted in Figure 3.7. To perform the plane sweep algorithm on S it is necessary to keep track of certain information in some data structures. All the points from all the segments will be held in x-dimensional order in a queue called the event queue. This queue ensures that the vertices are processed from left to right, giving the sweeping notion to the algorithm. When an intersection point is found it is also added, in the correct place, to this queue. The next action to be taken is dependent on the contents of the head of this queue; intersection points are processed differently from other vertices. Another queue R will hold the points that are currently being processed in order of the y-dimension. This queue, R, only holds vertices that are relevant to the intersection test, reducing the number of necessary tests. The

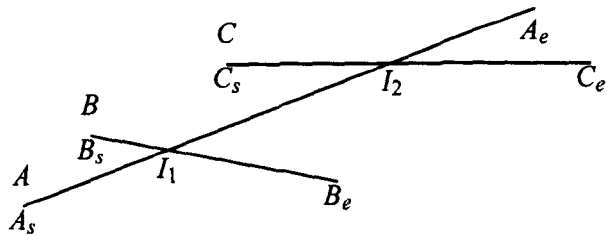


Fig. 3.7. The Collection of Line Segments S.

output intersection points will be stored in another list I. The pseudo code algorithm is given in Algorithm 3.1.

Algorithm 3.1 The Bentley-Ottmann Plane Sweep Algorithm.

- **Inputs:** list of segments S;
- Populate Q with start and end points from all segments in S;
- ensure these are ordered by x values;
- for each point p in Q:
  - if p is a start point of a segment s then:
    - if s intersects with any of the segments in R then insert the intersection points into Q and I and sort Q;
  - else if p is an end point of a s segment s then:
    - remove s from R;
  - else if p is an intersection point of segments s and t then:
    - swap the position of s and t in R;
    - check the upper segment for an intersection with the segment above it in R;
    - check the lower segment for an intersection with the segment below it in R;
    - add any intersection points to Q and I.
- **Outputs:** list of segments Q;

An example will now be given to illustrate the plane-sweep algorithm. Consider the three line segments A, B and C depicted in Figure 3.7. A visual inspection of Figure 3.7 shows there are two intersection points I_1 and I_2 to be identified by the algorithm. The algorithm will now be worked through using these three segments to demonstrate how I_1 and I_2 are identified. The notation A_s and A_e is used to denote the respective start and end points of segment A.

Initialisation Step:

$Q = \{A_s, B_s, C_s, B_e, A_e, C_e\}$. $R = \{\}$. $I = \{\}$.

Iteration 1:
$Q = \{\mathbf{A_s}, B_s, C_s, B_e, A_e, C_e\}$. $R = \{A\}$. $I = \{\}$. No intersection test possible.
Iteration 2:
$Q = \{A_s, \mathbf{B_s}, I_1, C_s, B_e, A_e, C_e\}$. $R = \{B, A\}$. $I = \{I_1\}$. Test if B and A intersect. I_1 identified.
Iteration 3:
$Q = \{A_s, B_s, \mathbf{I_1}, C_s, B_e, A_e, C_e\}$. $R = \{A, B\}$. $I = \{I_1\}$. No further intersection tests necessary.
Iteration 4:
$Q = \{A_s, B_s, I_1, \mathbf{C_s} I_2, B_e, A_e, C_e\}$. $R = \{C, A, B\}$. $I = \{I_1, I_2\}$. Test if A and C intersect. I_2 is identified. Test whether B and C intersect. No intersection found.
Iteration 5:
$Q = \{A_s, B_s, I_1, C_s, \mathbf{I_2}, B_e, A_e, C_e\}$. $R = \{A, C, B\}$. $I = \{I_1, I_2\}$. No further intersection tests necessary.
Iteration 6:
$Q = \{A_s, B_s, I_1, C_s, I_2, \mathbf{B_e}, A_e, C_e\}$. $R = \{A, C\}$. $I = \{I_1, I_2\}$. No further intersection tests necessary.
Iteration 7:
$Q = \{A_s, B_s, I_1, C_s, I_2, B_e, \mathbf{A_e}, C_e\}$. $R = \{C\}$. $I = \{I_1, I_2\}$. No intersection test possible.
Iteration 8:
$Q = \{A_s, B_s, I_1, C_s, I_2, B_e, A_e, \mathbf{C_e}\}$. $R = \{\}$. $I = \{I_1, I_2\}$. No intersection test possible.

This Section has given an efficient technique for identifying intersection points in a collection of 2-dimensional line segments. The next Section utilises this technique to provide manipulation operations for geometric primitives. These operations are later used to define fuzzy operations for geometric fuzzy sets.

3.2.4 Weiler-Atherton Clipping

In the field of computer graphics it often necessary to remove the intersecting areas from two or more of polygons. Consider the example of a computational model of a 3-dimensional scene having to be drawn or visualised in only 2-dimensions. Such an example is given in Figure 3.8, where the scene consists of a rectangle and a triangle. For the scene to be correctly drawn in 2-dimensions the area of the triangle that is obscured by the rectangle must be removed or clipped. There are a number of clipping algorithms (Cyrus & Beck 1978, Foley & Van Dam 1982, Sutherland & Hodgman 1974) each with strengths and weaknesses in differing applications areas. Later in this Chapter the geometric models of fuzzy sets based on straight line segments are explored. When defining the geometric conjunction and disjunction of two geometric fuzzy sets a clipping algorithm that works efficiently with vertices and line segment models will be required. The algo-

rithm widely used in such clipping applications is the Weiler-Atherton (1977) clipping algorithm given in Algorithm 3.2.

Algorithm 3.2 The Weiler-Atherton Clipping Algorithm.

- **Inputs:** lists of segments *clip*, *subject*;
- Ensure all vertices including intersection points in both the *subject* and *clip* polygons are ordered clockwise;
- begin at the first point in the *subject* polygon;
- until one of the vertices all ready processed is reached:
  - follow the *subject* polygon in clockwise order until an entering intersection point is come upon. This is the first vertex to be added to the clipped area A;
  - continue following the *subject* polygon clipping vertices as they are encountered until an exiting intersection is encountered;
  - follow the *clip* polygon in clockwise order from the exiting intersection until another intersection found then switch back to the following the *subject* polygon;
- if all points in the *subject* polygon have not been traversed then repeat this process beginning at the last point that was processed.
- **Outputs:** lists of segments *clip*, *subject* and A;

An example of Weiler-Atherton clipping will now be worked through. Consider the two polygons depicted in Figure 3.9. In this example the triangle A, B, C is being clipped against the rectangle a, b, c, d. This is the same situation as the example from Figure 3.8. The rectangle is in front the triangle in the viewing plane. The portion of the triangle that is being obscured by the rectangle must therefore be removed for the scene to render correctly. The triangle is referred to as the subject polygon and the rectangle is referred to as the clip polygon. The subject polygon is always clipped against the clip polygon. Before clipping can be performed all points where the two polygons intersect must be found. The Bentley-Ottmann plane-sweep algorithm can be used for this. In our example the intersection points are denoted 1 and 2. Intersection points can either be entering or exiting intersection points. An entering intersection point is one that is encountered immediately before an area covered by both polygons is reached. An exiting intersection point is one that is encountered as such an area is exited. In our example the vertex 1 is an entering point and the vertex 2 is an exiting point. The algorithm is now applied to this example.

The example depicted in Figure 3.9(a) will now be worked through.

- Create a vertex list for each polygon ensuring clockwise ordering:

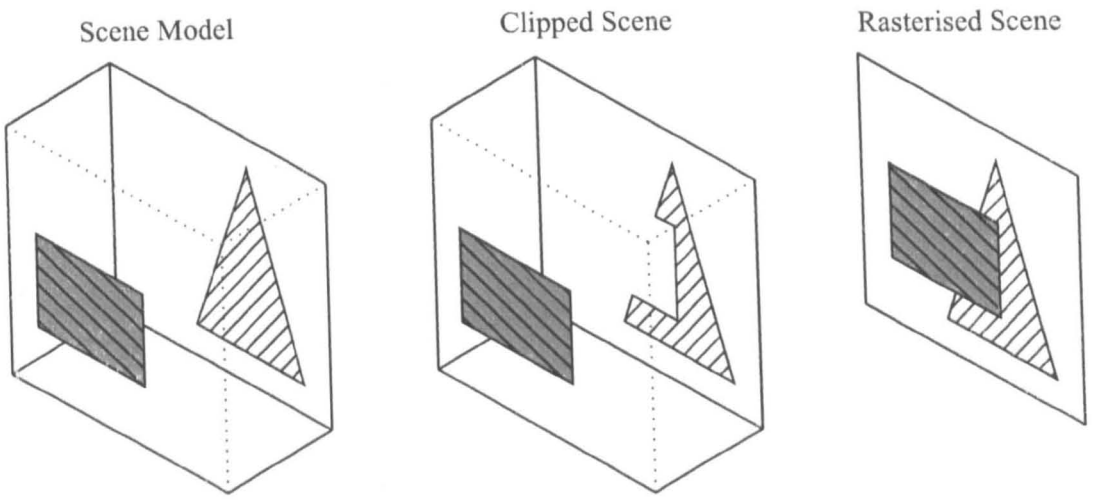


Fig. 3.8. An Example of a Polygon Clipping.

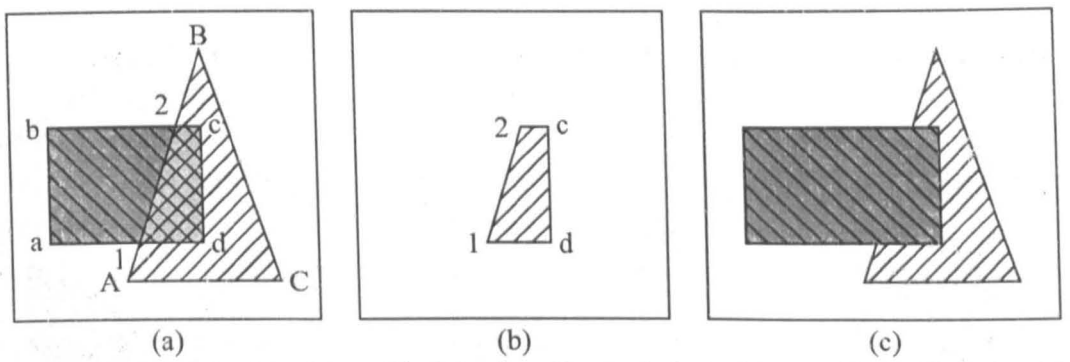


Fig. 3.9. (a) A 2-Dimensional Depiction of the Triangle and Rectangle Scene in Figure 3.8. (b) The Clipped Area of (a) After Weiler-Atherton Clipping. (c) The Rasterised Scene after Weiler-Atherton Clipping.

- subject list = $A, 1, 2, B, C$;
- clip list = $a, b, 2, c, d, 1$;

- begin iterating through the subject list starting with vertex A;
- begin processing:
  - found entering intersection point 1. 1 added to output list. Continue traversing subject list;
  - found exiting intersection point 2. 2 added to output list. Switch to following clip list;
  - found point c. c added to output list. Continue traversing clip list;
  - found point d. d added to output list. Continue traversing clip list;
  - found previously visited point 1;

- all vertices in the subject polygon have not yet been traversed so continue algorithm from point 2;
    - found point B. Continue traversing subject list;
    - found point C. Continue traversing subject list;
    - found previously visited point A;
- all vertices in the subject polygon have been traversed so the algorithm terminates;
- output list = $1, 2, c, d$;
- rectangle = a, b, c, d, and;
- former triangle = $A, 1, d, c, 2, B, C$.

Figure 3.9(b) depicts the area clipped, the output list in the example above and Figure 3.9(c) depicts the result from the clipping operation, the rectangle and former triangle lists.

The Weiler-Atherton clipping algorithm works directly with geometric primitives, namely vertices and line segments. Later this Chapter describes how this method can be used to give to the logical operators for type-1 geometric fuzzy sets based on such geometric primitives. The following Section describes the calculation of the centroid of a polygon made up of vertices and segments.

3.2.5 The Centroid of a Polygon.

A crucial step in fuzzy inferencing is defuzzification, where a fuzzy set which has been output from a fuzzy inferencing process is transformed back into a crisp number. One method for performing this transformation from a fuzzy set to a crisp number is to calculate the centroid of the fuzzy set. The centroid is the x-component of the centre of the area encompassed by the fuzzy set. This Section presents a geometric approach to calculating the centre of this area when the membership of a fuzzy set is a PLF.

As demonstrated in Section 3.1 the membership function of a fuzzy set can be given as a piecewise linear function. Connecting the ends of the PLF with a line segment $(x_n, y_n)(x_0, y_0)$ results in a polygon that is equal to the area encompassed by the fuzzy set. The centroid of a polygon (Bashein & Detmer 1994, Bourke 1988) can be calculated by deconstructing the polygon into a collection of triangles. Each of these triangles has one vertex at $(0,0)$ with the other two vertices taking values in order from one of the line segments that forms the polygon. This means a polygon with n vertices can be broken down into n triangles. The centroid of the polygon is the weighted average of the area and centre of these triangles. Consider the polygon P_1 depicted in Figure 3.10(a). This

polygon has 4 vertices and can therefore be broken down into 4 triangles t_1 to t_4 as depicted in Figures 3.10(b) to 3.10(e) where the dotted lines depict P_1. Note that the triangles t_1, t_2 and t_3 all encompass an area that lies outside the polygon P_1. The sum of these areas from t_1 to t_3 is equal to the entire area of the triangle t_4. Since a signed value for the area is taken for each triangle these overlapping areas will cancel out. This is because the sign of area of t_4 will be the opposite to all the other triangles.

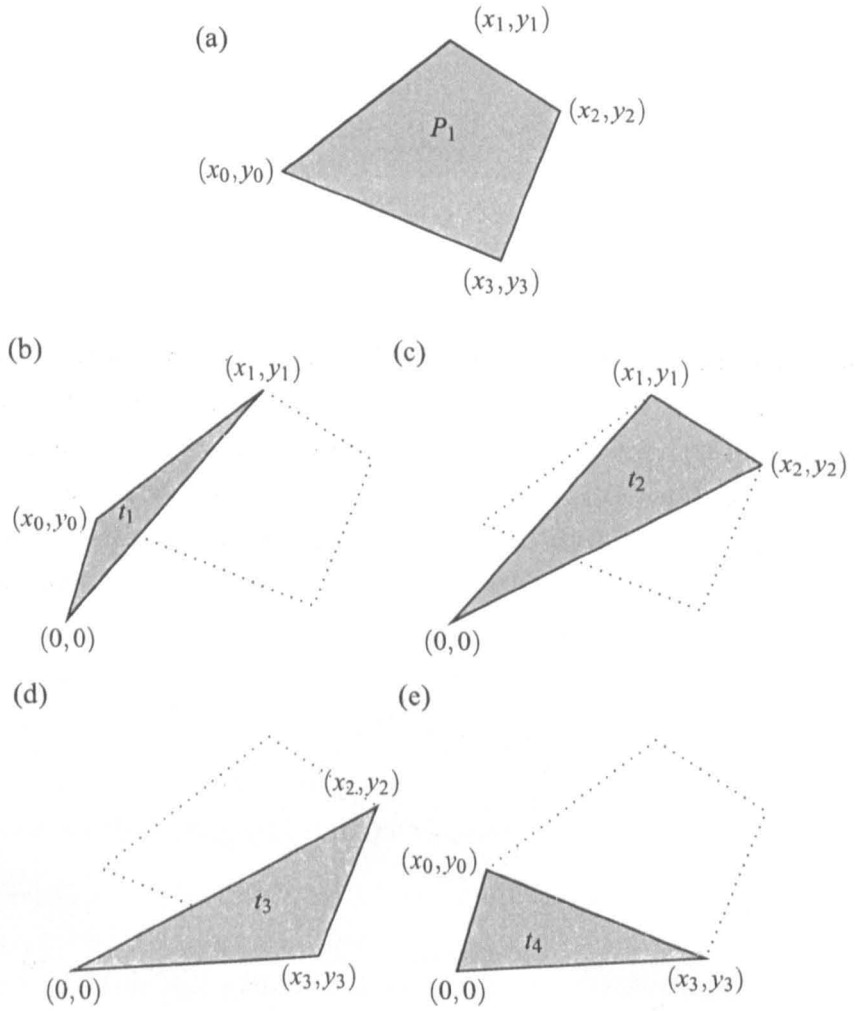


Fig. 3.10. Calculating the Centroid of Polygon P_1 using Constituent Triangles t_1 to t_4.

The signed area of a triangle is given by the half of the cross product of two of the edge vectors. The centre of a triangle is the sum of the vertices divided by three. The area of t_1 is therefore

$$Area\ t_1 = \frac{(x_1 - 0)(y_2 - 0) - (x_2 - 0)(y_1 - 0)}{2} = \frac{x_1 y_2 - x_2 y_1}{2} \tag{3.10}$$

Since all the triangles from the polygon P_1 contain the vertex $(0,0)$ the area A of any triangle t_i is

given by

$$A(t_i) = \frac{x_i y_{i+1} - x_{i+1} y_i}{2} \tag{3.11}$$

An assumption is made that the polygon starts and ends at the same vertex (x_0, y_0). The x-component of the centre or centroid C of the triangle t_i is given by

$$C(t_i) = \frac{x_i + x_{i+1}}{3} \tag{3.12}$$

The x-component of the centroid C of a polygon P is given by

$$C = \frac{\sum_{i=0}^{n-1} A(t_i) C(t_i)}{\sum_{i=0}^{n-1} A(t_i)} \tag{3.13}$$

Where n is the number of vertices that make up P, $A(t_i)$ and $C(t_1)$ are given by equations 3.11 and 3.12 respectively. Substituting equations 3.11 and 3.12 into equation 3.13 gives

$$C = \frac{\sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)}{3(\sum_{i=0}^{n-1} x_i y_{i+1} - x_{i+1} y_i)} \tag{3.14}$$

This method for finding the x-component of the centroid of a polygon will inform the discussion on defuzzification of a geometric fuzzy set later in this Chapter. Having discussed ways of modelling and manipulating 2-dimensional geometric objects, the following Section discusses 3-dimensional models. Such models will be utilised later on in this Chapter when type-2 geometric fuzzy sets and systems are explored.

3.2.6 Surface Modelling Using a Triangulated Mesh

In the proceeding Chapter type-2 fuzzy sets were discussed. It was demonstrated that type-2 fuzzy sets require a 3-dimensional model. To geometrically describe such objects a method for modelling a 3-dimensional surface must be found. A simple solution would be to extend our parametric line models to include a third dimension. A surface could then be modelled as a collection of connected polygons. Surface clipping algorithms, analogous to the line clipping algorithms, could then be defined. However, as will be seen in the next Section, for surface intersections to be calculated easily, each of the polygons would have to lie on a plane. This means either placing a constraint on the polygons that they must all be planar or restricting the number of vertices in the polygons to be three, i.e. only use triangles as a triangle must by definition always lie on a plane. This work only uses triangles to model 3-Dimensional surfaces. This makes the surfaces clipping algorithms employed later in this Chapter a great deal simpler and also less

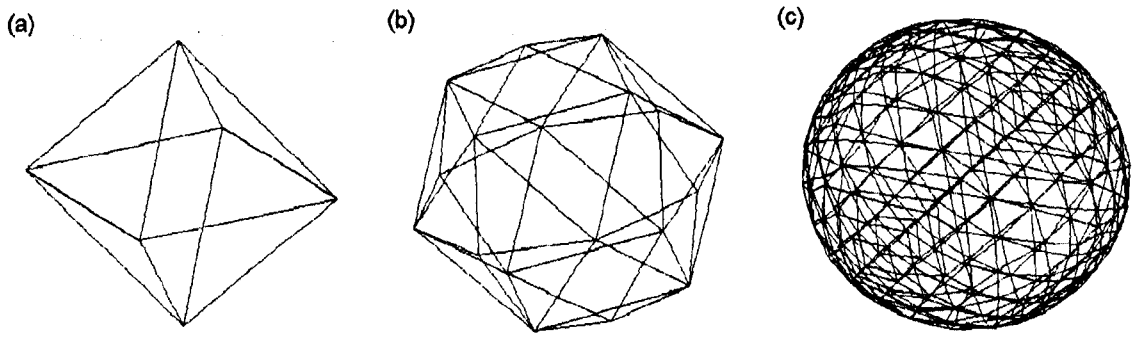


Fig. 3.11. Three Approximations of a Spherical Surface Using 8, 32 and 512 Triangular Facets.

computational expensive.

Any non planar surface can then be approximated using a mesh of connected 3-dimensional triangles. The level of approximation i.e., the error in the surface model is dependent on the number of triangles used. This is analogous to using a collection of 2-dimensional line segments to approximate any 2-dimensional curve. Consider the examples given in Figures 3.11 (a), (b) and (c). Each figure depicts an approximation of a spherical surface as modelled by a triangulated mesh. The Figures 3.11 (a), (b) and (c) use 8, 32 and 512 triangles respectively to model the surface of a sphere. The number of triangles used to model a particular surface in a system is a choice made by the system developer. A balance must be struck between the model accuracy and the computational resources needed to process the model in a timely fashion.

Triangular facet meshes provide a method for modelling, with some degree of approximation, any arbitrary 3-dimensional surface. This method will be exploited later on in this Chapter when geometric models of type-2 fuzzy sets are discussed. The advantage of only using facets consisting of three vertices i.e. triangles is that each facet must be planar. In the following Section a method for finding the points where two 3-dimensional triangles intersect is given. This method exploits planar nature of triangles to find these intersection points.

3.2.7 Guigue and Devillers Triangle-Triangle Overlap Test

Triangle meshes provide a method for modelling any 3-dimensional surface. If the surface of a type-2 fuzzy set is to be modelled using a triangular mesh then ways of defining logical operations on such meshes will need to be found. This is likely to involve a method of calculating any points where two triangles intersect so that one triangle can be clipped against the other. The ability to clip triangles is fundamental to finding the highest (maximum) and lowest (minimum) possible surface of two meshes. This is explored in more detail in Section 4.3.2.3 where the conjunction and disjunction of two type-2 fuzzy sets is discussed. The Guigue and Devillers triangle-triangle

overlap test (Guigue & Devillers 2003), an extension of Möllers intersection test (Möller 1997), provides a method for testing for and calculating any points where two triangles intersect. This method is summarised in Algorithm 3.3.

Algorithm 3.3 The Guigue and Devillers Triangle-Triangle Overlap Test.

- **Inputs:** triangles t_1 and t_2 defined by their vertices P_1, Q_1, R_1, P_2, Q_2 and R_2;
- Let the two triangles being tested be t_1 and t_2 lying on the planes π_1 and π_2 respectively;
- check whether t_1 lies entirely in a half space of π_2. If so triangles cannot intersect so exit algorithm;
- check whether t_2 lies entirely in a half space of π_1. If so triangles cannot intersect so exit algorithm;
- check that $\pi_1 \neq \pi_2$. If $\pi_1 = \pi_2$ then the triangles both lie on the same plane and can be processed using 2-dimensional methods;
- rotate both triangles into canonical form;
- test for overlapping intersection points. If no points from either triangle overlap then the triangles do not intersect so exit the algorithm, and
- calculate the intersection points I_1 and I_2 of the triangles.
- **Outputs:** points I_1 and I_2.

The method is now explored in greater detail. Consider the two triangles t_1 with vertices P_1, Q_1 and R_1, and t_2 with vertices P_2, Q_2 and R_2 on the respective planes π_1 and π_2 depicted in Figure 3.12. The algorithm begins by testing whether t_1 intersects with π_2 and whether t_2 intersects with π_1. This is done by comparing the distance from each vertex to the plane of the opposing triangle. First take the distances from P_1, Q_1 and R_1 to the plane π_2, denoted dp_1, dq_1 and dr_1. When the signs of dp_1, dq_1 and dr_1 are compared three distinct situations can be identified. If all three have the same sign, then t_1 lies in one of the half spaces of π_2. If all three are zero then the t_1 and t_2 lie on the same plane and can therefore by handled using 2-dimensional methods. If one of the distances has a different sign then t_1 intersects the plane π_2 and the algorithm continues. The same is done for the distances dp_2, dq_2 and dr_2. If one of the distances has a different sign then the algorithm continues as each triangle must intersect the plane of the opposing triangle. The next stage involves rotating the vertices of each triangle in such a way that P_1 lies on the opposite side of π_2 than Q_1 and R_1 and P_2 lies on the opposite side of π_1 than q_2 and R_2. Also P_2, Q_2 and R_2 are ordered counter clockwise with respect to P_1 and P_1, Q_1 and R_1 are ordered counter clockwise with respect to P_2. This is the vertex ordering depicted in Figure 3.12. The line l depicts the line

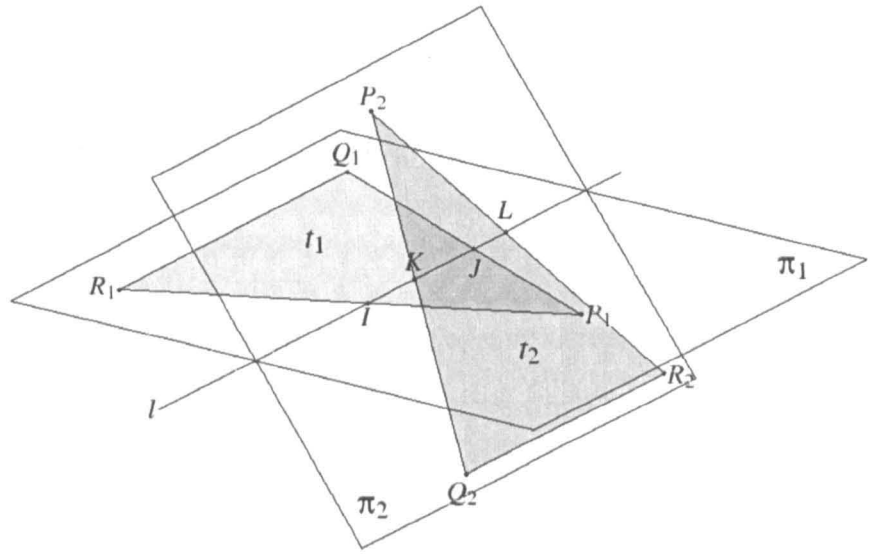


Fig. 3.12. Intersecting Triangles and The Planes in Which They Lie. Adapted from Guigue and Devillers (2003).

where the planes π_1 and π_2 intersect. There must now be a point along each of vectors $p_1\vec{q}_1$, $p_1\vec{r}_1$, $p_2\vec{q}_2$ and $p_2\vec{r}_2$ that intersects the line l. These points are denoted I, J, K and L respectively. A series of boolean operations then finds the order of I, J, K and L along l. If this ordering confirms that the triangles do intersect then intersection points are calculated. In the case of t_1 and t_2 the points follow the ordering I, J, K and L giving the intersection points K and J. The calculations to give the points J and K are given below.

$$K = P_2 - (\vec{p}_2 - \vec{q}_2)\frac{(\vec{p}_2 - \vec{p}_1)\cdot\vec{n}_1}{(\vec{p}_2 - \vec{q}_2)\cdot\vec{n}_1} \tag{3.15}$$

$$J = P_1 - (\vec{p}_1 - \vec{p}_2)\frac{(\vec{p}_1 - \vec{p}_2)\cdot\vec{n}_2}{(\vec{p}_1 - \vec{q}_1)\cdot\vec{n}_2} \tag{3.16}$$

Where $\vec{n}_1$ and $\vec{n}_2$ are the normals to the respective planes π_1 and π_2, $\cdot$ is the dot product of two vectors and $\vec{p}_i$ is the position vector of P_i. This algorithm provides a method for testing for and calculating the intersection points of two triangles in 3-dimensions. The use of a series of single geometric tests to ascertain the way the two triangles are interacting simplifies the intersection point calculation and reduces the possibility of errors from floating point calculations.

The current and preceding sections of this Chapter have explored the geometric models and algorithms that underpin geometric fuzzy logic. The remainder of this Chapter presents the novel geometric approach to fuzzy systems. This work forms the theoretical core of this thesis.

3.3 Geometric Type-1 Fuzzy Logic Systems

Type-1 fuzzy systems are used in a wide range of application areas. Current discrete models are robust and efficient, if not always perfectly accurate. In this Section a geometric model of a type-1 FLS is defined. This offers an alternative, more accurate method of implementing type-1 fuzzy logic systems. This also represents a first step toward geometric models of interval type-2 and type-2 fuzzy logic systems. Most of the work in the preceding sections of this Chapter has been revised or reworked from other sources. The remained of this Chapter defines the novel geometric approach to type-1 fuzzy systems. The geometric approach to type-1 fuzzy systems was presented by the author in Coupland and John (2004*b*).

3.3.1 Geometric Type-1 Fuzzy Sets

Fundamental to any method of fuzzy reasoning is the representation of a fuzzy set. The geometric fuzzy reasoning approach explored in this work is built on a fuzzy set model consisting of geometric primitives, vertices and line segments. Any fuzzy set is defined by its membership function. A geometric type-1 fuzzy set is defined as a type-1 fuzzy set with a membership function that is a piecewise linear function.

Definition 3.2 *A type-1 geometric fuzzy set is characterised by a membership function consisting of a series of order vertices connected by line segments to form a function over a continuous domain. Each vertex of a fuzzy set A over the domain X consists of a value x such that* $x \in X$ *and a value y such that* $y \in [0,1]$ *i.e.,*

$$A = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n) \mid x_i \in X, y_i \in [0,1], x_i < x_{i+1} \forall i \in [1,n]\} \tag{3.17}$$

Three example type-1 geometric fuzzy sets are depicted in Figure 3.13. The Figures 3.13 (a), (b) and (c) respectively depict a triangular fuzzy set, a trapezoidal fuzzy set and an approximation of a Gaussian fuzzy set. The Gaussian fuzzy set depicted in Figures 3.13 (c) is approximated with nine vertices.

Definition 3.2 states that a geometric fuzzy set is characterised by a membership function that is a piecewise linear function. In Section 3.1 it was shown that all type-1 fuzzy sets can be approximated by a PLF. The point of departure for geometric type-1 fuzzy sets is that they must always be characterised by a PLF. The following Section draws upon the geometric methods given earlier in this Chapter to define the various operations needed for inferences to be made using geometric fuzzy sets.

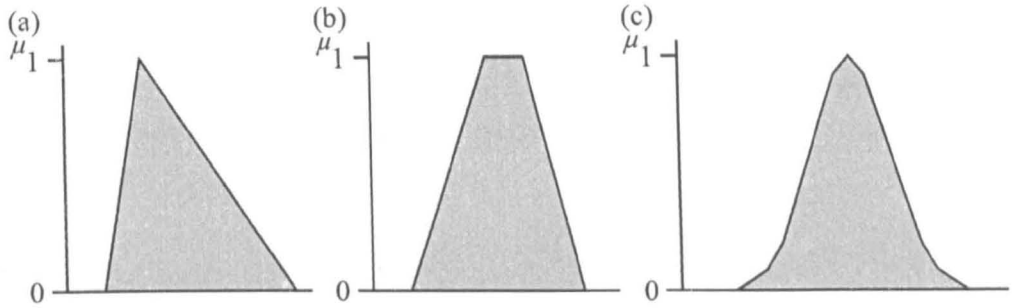


Fig. 3.13. (a) A Triangular Fuzzy Set, (b) A Trapezoidal Fuzzy Set and (c) An Approximation of a Gaussian Fuzzy Set.

3.3.2 Geometric Type-1 Fuzzy Inferencing

Fuzzy logic systems (see Figure 3.1.) are essentially rule based logic systems where the rules are fuzzy, that is, each rule fires to a fuzzy degree. The antecedent of a type-1 fuzzy rule has a value in $[0, 1]$. The consequent of each rule is one or more type-1 fuzzy sets. The combination of the rule consequents is a type-1 fuzzy set and must therefore be defuzzified if crisp output value is to be output.

3.3.2.1 Fuzzification

Fuzzification is the process of finding to what degree a particular value x belongs to a given fuzzy set A. The membership function of a geometric fuzzy set is a PLF, as such there are three possible situations that could occur when fuzzifying x in A. The value x could be outside the bounds of the fuzzy set, in which case x has a membership of zero in A. Should x be equal to a x-component of one of the vertices that make up A then the membership of x in A is the y-component of that vertex. When x lies between two vertices from A the membership of x in A is found by linear interpolation between the two vertices. More formally the membership grade μ_A for any particular value of x in a geometric fuzzy set A is given by:

$$\mu_A(x) = \begin{cases} 0 & ; x \leq x_1 \text{ or } x_n \leq x \\ y_i & ; x = x_i \\ y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) & ; x_i < x < x_{i+1} \end{cases} \tag{3.18}$$

where x_0, x_i and x_n are, respectively, the x-component of the first, i^{th} and last vertex and y_i is the y component of the i^{th} vertex. This is identical to equation 3.2 presented in the earlier discussion of piecewise linear functions. Each input is fuzzified in one or more fuzzy set as defined by the rules in the rule base. The antecedent of each rule may have a number of fuzzified values that are combined to give the antecedent value, the rules firing strength. Since this combination involves crisp numbers and not fuzzy sets the geometric model has no impact on this part of the inferencing

process. As with discrete fuzzy systems the conjunction of two fuzzified values is given with a t-norm and the disjunction with a t-conorm. This firing strength is then passed to the inference engine of the geometric type-1 fuzzy logic system for the calculation and combination of the rule consequents.

3.3.2.2 Implication

Implication is the process of calculating the value of a consequent in a given fuzzy rule according the firing strength of that rule. Consider the fuzzy rule, where the firing strength $\alpha = \mu_A(a) \star \mu_B(b)$:

IF a is A AND b is B THEN g is G

In a discrete fuzzy system the implication operation would involve taking a t-norm of the firing strength α and the membership grade of the consequent fuzzy set G at every point in G. As discussed earlier, the geometric model cannot model any t-norm, only the minimum. The geometric implication operation involves finding the minimum of a crisp number α and a PLF G at every point in G. To perform this operation a geometric model of α must first be constructed. The value of α across the domain of G can be modelled by a line segment fs which will now be defined. Let the first and the last vertices of the consequent fuzzy set G be denoted by (x_0, y_0) and (x_n, y_n) respectively. Let the start point of fs be $(x_0 - \delta, \alpha)$ and the end point of fs be $(x_n + \delta, \alpha)$, where δ is some positive number which is small with respect to $x_n - x_0$. The line segment fs has a y-component equal to α at value of x across G. Such a segment fs is depicted in Figure 3.14 (a) with a consequent fuzzy set G. The geometric implication of fs on G could then be given applying the

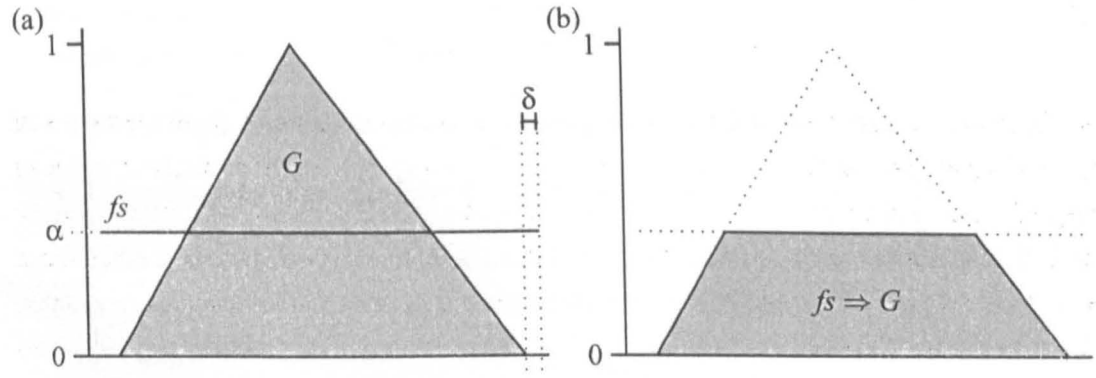


Fig. 3.14. (a) The Consequent Fuzzy Set G and the Line Segment fs. (b) The Result of $fs \Rightarrow G$.

Weiler-Atherton clipping algorithm to the geometric primitives fs and G. However, since fs is a perfectly horizontal line segment a simpler operation can be given. This geometric type-1 implication operation is given in Algorithm 3.4. This algorithm clips every line in G against a y value α, giving the minimum of μ_G and α at every point in G. The reason that the Weiler-Atherton clipping

Algorithm 3.4 The Geometric Type-1 Implication Operation.

- **Inputs:** antecedent value α and the geometric fuzzy set G;
- Let α be the computed antecedent value of a rule R. Let the consequent fuzzy set G of rule R be defined the vertices $(x_1,y_1)\ldots(x_n,y_n)$;
- for every value of i between 0 and $n-1$ let the line segment $(x_i,y_i)(x_{i+1},y_{i+1})$ be replaced by:
  - the segment $(x_i,y_i),(x_{i+1},y_{i+1})$ when y_i and y_{i+1} are both less than α;
  - the segment $(x_i,\alpha),(x_{i+1},\alpha)$ when both y_i and y_{i+1} are greater than α;
  - the two line segments $(x_i,y_i)(x_a,\alpha)$ and $(x_a,\alpha)(x_{i+1},\alpha)$ when y_i is less than α and y_{i+1} is greater than α, where

$$x_a = x_i + \frac{\alpha - y_i}{y_{i+1} - y_i}(x_{i+1} - x_i) \tag{3.19}$$

  - the two line segments $(x_i,\alpha)(x_a,\alpha)$ and $(x_a,\alpha)(x_{i+1},y_{i+1})$ when y_i is greater than α and y_{i+1} is less than α, where x_a is given by equation 3.19.
- **Outputs:** the geometric fuzzy set $\alpha \Rightarrow G$;

algorithm is not needed for this operation is that *alpha* is a constant value limiting the ways in which G and *fs* can interact. The y-component of any intersection point of *fs* and G will always be α. This reduces computation of the intersection points and the computation required to test of an intersect point. Any line segment in G can only intersect *fs* if α is between y_i and y_{i+1}. Because of this it is sensible to use an simplified algorithm based on Weiler-Atherton clipping provided by Algorithm 3.4.

The example implication given in Figure 3.14 (a) is now worked through.

- The consequent fuzzy set G consists of three vertices (x_1,y_1), (x_2,y_2) and (x_3,y_3);
- in the first segment y_1 is less than α and y_2 is greater than α. Therefore the two line segments $(x_1,y_1)(x_a,\alpha)$ and $(x_a,\alpha)(x_2,\alpha)$ are output. where

$$x_a = x_1 + \frac{\alpha - y_1}{y_2 - y_1}(x_2 - x_i) \tag{3.20}$$

- in the second segment y_2 is greater than α and y_3 is less than α. The two line segments $(x_2,\alpha)(x_b,\alpha)$ and $(x_b,\alpha)(x_3,y_3)$ are output, where

$$x_b = x_2 + \frac{\alpha - y_2}{y_3 - y_2}(x_3 - x_2) \tag{3.21}$$

The output from this algorithm is the PLF $fs \Rightarrow G$ depicted in Figure 3.14 (b). This Section has given a geometric interpretation of the type-1 fuzzy implication operation. The next Section gives a geometric model of the fuzzy 'or' operation.

3.3.2.3 Combination of Rule Consequents

Once each rule has fired all the implied consequent fuzzy sets have to be combined in some way. If computational resource is a limiting factor then it is possible to defuzzify each consequent and aggregate the results. Generally however, all the consequents are combined using the 'or' operator. This involves taking the t-conorm of the membership grades of all the consequents at each point in the domain. As discussed earlier, the only t-conorm that currently has a geometric interpretation is maximum. The Weiler-Atherton clipping algorithm gives the line segments that describe the minimum of all points across two PLFs. This algorithm can be easily modified to give the line segments that describe the maximum of all points across two PLFs. The modified Weiler-Atherton clipping algorithm is given in Algorithm 3.5.

Consider the two geometric fuzzy sets A and B depicted in Figures 3.15 (a) and (b). The application of the modified Weiler-Atherton clipping algorithm to these two fuzzy sets is now given.

- ensure vertices are correctly ordered;
- ensure PLF are in order A then B by x-component of they first vertex, and they are;
- test that A and B are not disjoint, and they are not;
- use the plane sweep algorithm to find any intersection points. Points i_1 and i_2 are identified;
- subject PLF: v, i_1, x, y, i_2, z;
- clip PLF: a, b, i_1, i_2, c, d;
- begin iterations:
    - follow clip PLF. Append a to C;
    - follow clip PLF. Append b to C;
    - intersection point found. Append i_1 to C. Switch to subject PLF;
    - follow subject PLF. Append x to C;
    - follow subject PLF. Append y to C;
    - intersection point found. Append i_2 to C. Switch to clip PLF;
    - follow clip PLF. Append c to C;

Algorithm 3.5 The Modified Weiler-Atherton Clipping Algorithm.

- **Inputs:** piecewise-linear functions A and B;
- Ensure all vertices in both A and B are ordered by their x-component;
- let A be the PLF with the start vertex with the lowest x-component and B be the other PLF;
- if the first vertex in B has an x-component that is greater than the x-component of the last vertex in A i.e. they are disjoint, then concatenate A and B to give C, output this result and exit the algorithm;
- use the Bentley-Ottmann plane sweep algorithm to identify all intersection points of A and B. If no intersection points are found then let $C = A$ and exit the algorithm;
- let the subject PLF be B with any intersection points included in order;
- let the clip PLF be A with any intersection points included in order;
- begin at the first point in the clip polygon, append this point to C;
- until all the last vertex in A or B is reached:
    - follow the clip PLF in order adding points to C, until an intersection point is come upon, and
    - follow the subject PLF in order adding points to C, until an intersection point is come upon then switch back to the clip PLF.
- **Outputs:** the piecewise-linear function C;

    - follow clip PLF. Append d to C;
- last vertex in A has been reached so algorithm terminates, and
- $C = a, b, i_1, x, y, i_2, c, d$.

The output from this example is depicted in Figure 3.15 (c) where the output geometric fuzzy set is labelled A or B. This output A or B is a composite geometric fuzzy set that has been arrived at through a geometric fuzzy inferencing process. This fuzzy set represents the decision made by the system based on the knowledge encoded in the fuzzy rules and inputs given to the system. The final stage in the geometric fuzzy inferencing process is defuzzification which arrives at a crisp value which is representative of the final fuzzy set. The following Section describes the defuzzification of a geometric type-1 fuzzy set.

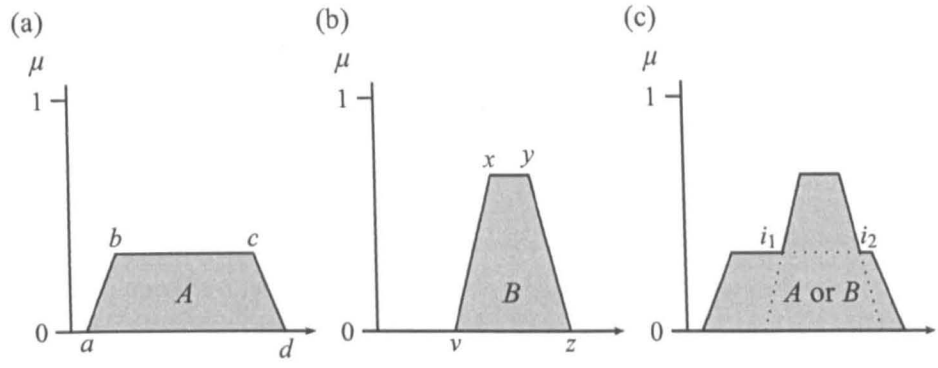


Fig. 3.15. (a) The Geometric Fuzzy Set A. (b) The Geometric Fuzzy Set B. (c) The Geometric Fuzzy Set A OR B.

3.3.2.4 Defuzzification

This final processing stage takes an inferred geometric fuzzy set and calculates a crisp value that is representative of this fuzzy set. The defuzzified value of a geometric fuzzy set C is the centre of the area (COA) of the C, denoted COA_C. The centroid of a polygon given in Section 3.2.5 can be used to find the COA of C. C is a PLF to transform it into a closed polygon the endpoints of the PLF must be connect with a line segment. This is done be appended an extra vertex to the end of C, the value of which equals the first vertex in C. This transforms C from a piecewise linear function into a closed polygon. The polygon COA method can then used find the centre of area of this closed polygon. The defuzzified value, the centroid of C is given by

$$COA_C = \frac{\sum_{i=1}^{n}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)}{3(\sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i)} \tag{3.22}$$

Adapted from Bourke (1988)

where C is $((x_1,y_1),(x_1,y_1),\ldots,(x_n,y_n))$ and $(x_n,y_n) = (x_1,y_1)$.

This Section has given a geometric equivalent to the discrete centre of area defuzzifier. This is the final component needed to have a complete geometric model of type-1 fuzzy logic. The next Section discuss issues that surround the use of discrete fuzzy sets.

3.4 Issues Surrounding Discretisation

Typically a fuzzy system that is deployed on either computer hardware or software will use discrete fuzzy sets. This is due to the way the fuzzy sets are represented in the computers memory. Fuzzy sets are typically represented as a number of discrete point with operations manipulating this discrete points algorithmically.

The level of discretisation decides the level of accuracy of the set model and amount of processing

resources it takes to execute a fuzzy system. The relationship between the level of discretisation in a type-1 fuzzy system and the execution speed is linear. The more complex relationship between type-2 fuzzy systems and discretisation is explored in this Section, however more work in is before firm recommendations about discretisation levels can be given.

The optimised join and meet operations reduce the computational problems of these operations when using the minimum and maximum t-norm and t-conorm and convex secondary membership functions. These operations are used throughout the inferencing process up to the output processing stage. A type-2 system that conforms to the imposed constraints will therefore be computationally efficient up to the point of type-reduction. Type-reduction has a high computational expense which increases significantly as the number of discrete points in the set being type-reduced increase. Type-reduction requires that all the type-2 embedded sets that are needed to model a type-2 fuzzy set be enumerated and processed. The number of embedded sets required increases as the number of points in the type-2 fuzzy set increase. Figure 3.16 plots the number of embedded sets required to model a type-2 fuzzy set that has between zero and ten points along the domain of the primary and secondary membership functions. The z-axis, the number of embedded sets, in Figure 3.16 is logarithmic. The novel extensions to the Karnik and Mendel optimised join and meet allow fast type-2 inferencing with the minimum t-norm. Type-reduction, in particular the explosion in the number of embedded sets, prior to this thesis, was the remaining obstacle to the implementation of fast and efficient type-2 fuzzy logic systems. With the advent of geometric type-2 fuzzy systems, presented for the first time in this thesis, this remaining obstacle to efficient implementation of a type-2 fuzzy logic system has been overcome.

Discretisation also impacts on the accuracy of the set model. This is also true of type-1 fuzzy sets. However, the computational problems presented above demonstrate that systems that have fuzzy sets with large levels of discretisation will require a large amount of processing when type-reduced. This is likely to have the effect on many system developers of using fewer discrete points than would be used in a type-1 system, reducing computation but also leading to a reduction in accuracy. Consider the type-2 fuzzy set $\widetilde{A}$ depicted in Figure 3.17(a). $\widetilde{A}$ is a non-symmetrical type-2 fuzzy set over a continuous domain with both primary and secondary membership functions taking a triangular shape. If $\widetilde{A}$ is to be used in a real system then the widely accepted practice is to discretise both the primary and secondary membership functions of $\widetilde{A}$. Consider two possible discrete versions of $\widetilde{A}$. Firstly $\widetilde{A}_3$, which has three discrete points in the domain of both the primary and secondary membership functions. Secondly $\widetilde{A}_6$, which has six discrete points in the primary and secondary membership functions. The discrete sets $\widetilde{A}_3$ and $\widetilde{A}_6$ are depicted in Figure 3.17(b) and Figure 3.17(c) respectively. These Figures show that the triangular shape of the secondary membership functions is preserved at both levels of discretisation. However, the level of discretisation in $\widetilde{A}_3$ is so coarse that the primary membership function is no longer triangular. The

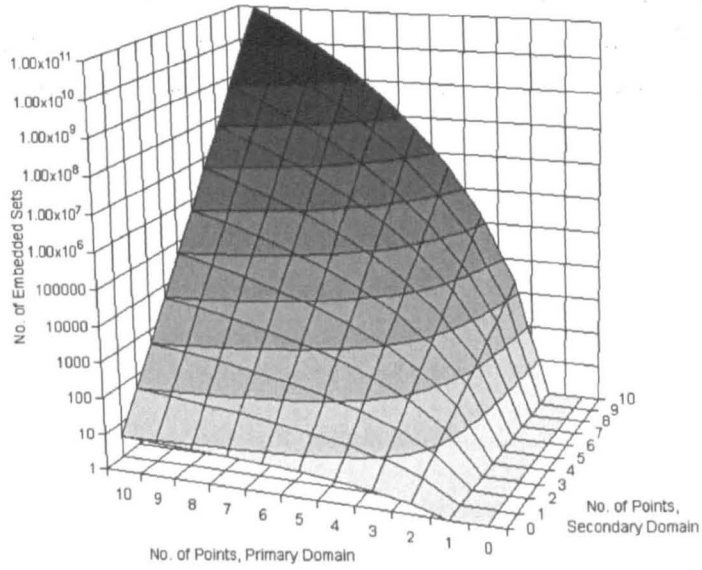


Fig. 3.16. Number of Embedded Sets Required to Model Type-2 Fuzzy Set of Primary and Secondary Discretisation Level Between Zero and Ten.

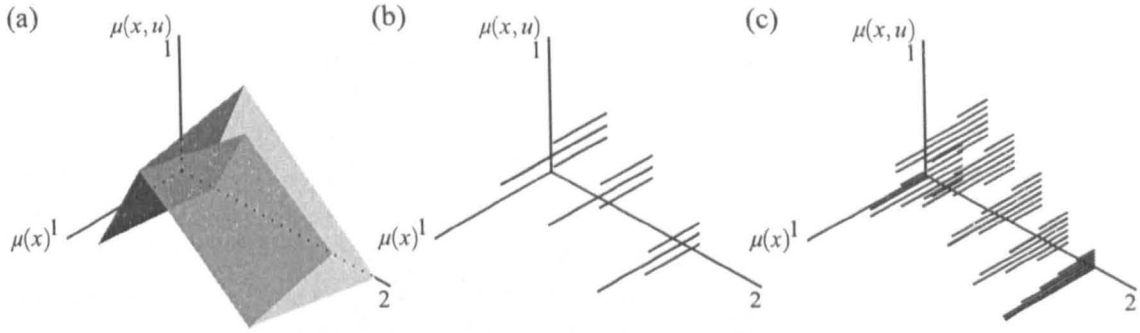


Fig. 3.17. (a) The Type-2 Fuzzy Set $\widetilde{A}$ with Triangular Primary and Secondary Membership Functions. (b) The Type-2 Fuzzy Set $\widetilde{A}_3$. (c) The Type-2 Fuzzy Set $\widetilde{A}_6$

primary membership function of $\widetilde{A}_6$ has retained the triangular form. The computational cost of type-reducing the sets is significantly different. The number of embedded type-2 sets that have to be enumerated in order to calculate a type-reduced set from $\widetilde{A}_3$ is 27. The number of embedded sets required to find the centroid of $\widetilde{A}_6$ is 46656. In this simple case quadrupling the number of discrete points has led to an increase of over 1700 times the amount of computation that is required to arrive at a type-reduced set. The final defuzzified values, the centroids of the type-reduced sets are also significantly different. The centroid of $\widetilde{A}_3$ is 1.375 and the centroid $\widetilde{A}_6$ is 1.55714. This difference of 0.18214 represents 9% of the support of type-2 fuzzy set $\widetilde{A}$.

This Section has explored the relationship between computational expense and the level of discretisation in type-2 fuzzy systems. The examples given also demonstrate that if discretisation levels are kept low then the accuracy of the set model can be adversely effected.

3.5 Discussion

This Chapter has presented a complete geometric model of a type-1 Mamdani style fuzzy logic system. This approach is novel, bringing together existing geometric operations and applying them to piecewise linear membership functions. The use of geometry to define logical operators brings about the following restrictions on geometric fuzzy logic.

- The membership functions in all the fuzzy sets must be piecewise linear functions;
- the implication operation must use the minimum t-norm, and
- the 'or' operation must use the maximum t-conorm.

The use of a PLF is not a significant restriction since many approaches already use some kind of linear function definition of a fuzzy set prior to discretisation. The restriction to the use of minimum t-norm when performing implication may be a slight restriction. Many engineering applications use the product rather than the minimum as the product results in a smoother control surface. The restriction on the 'or' operation is less important as the maximum is used in majority of fuzzy systems.

The underlying motivation behind the type-1 geometric model is to inform the interval and type-2 models. However the type-1 geometric fuzzy model has some important outcomes. The type-1 geometric model directly relates to the type-1 discrete model. Transforming between the two models is trivial. The geometric and discrete defuzzifiers produce equivalent results. This means that any variability between the two systems is wholly due to differences in set caused by discretisation. For piecewise linear membership functions the geometric fuzzy model is completely accurate. Discrete fuzzy models are never completely accurate, although the error level in a given application may not be significant in that particular system. This is useful since many popular classes of membership function such as triangular, trapezoidal and shoulder are piecewise linear functions. Other functions that are not piecewise linear, such as Gaussian, may also benefit from geometric modelling. An example of this was given by Coupland and John (2004) where a geometric model of a Z function was compared to a discrete model. The discrete and geometric fuzzy sets are reproduced in Figures 3.18(a) and 3.18(b) respectively. Both models had six points and the geometric model was shown to a give better degree of accuracy than the discrete model. The accuracy test was performed by comparing the centroid of the two sets with the centroid of a discrete version of those sets with one hundred discretisations. The results from Coupland and John (2004) are reproduced in Table 3.2

However neither this result nor the example given Section 3.1.1 take into account the computational complexity of the geometric system. The main advantage of the type-1 discrete model is that

Representation	Centroid (3 s.f.)	Actual Centroid (3 s.f.)
Discrete	1.70	1.73
Geometric	1.72	1.73

Table 3.2. A Comparison of the Centroids of the Discrete and Geometric Fuzzy Set Representations. From Coupland and John (2004).

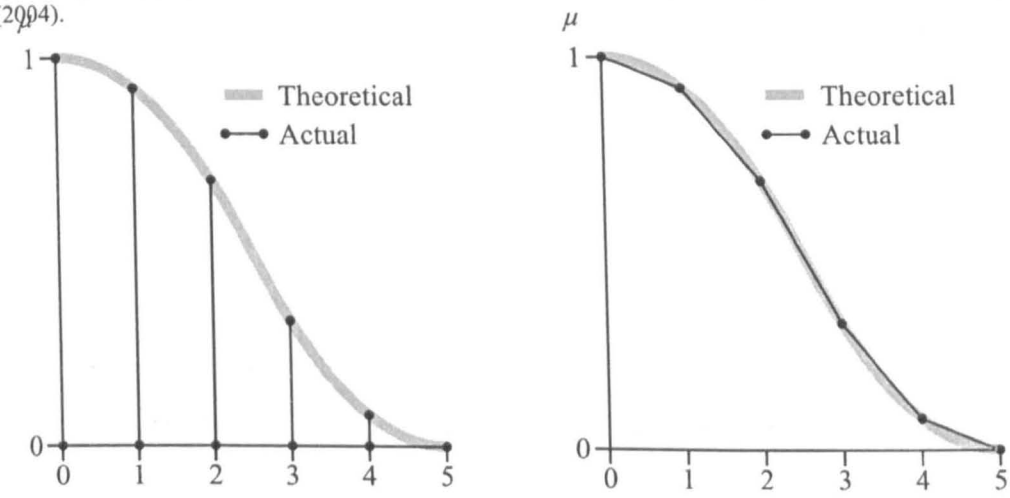


Fig. 3.18. (a) A Discrete Z Fuzzy Set. (b) A Geometric Z Fuzzy Set. From Coupland and John (2004).

has a very low computational burden. This is due to a simple model of the sets for which logical operators can be easily defined. The geometric model is more complex and requires more complicated algorithms to perform logical operations. A discrete set may require many more points than a geometric set to achieve the same level of accuracy. However to perform a logical operation each point in the discrete set requires relatively little computation compared to each point in a geometric fuzzy set where the logical operations are more elaborate. This leaves the system developer with a choice to balance the level accuracy with the ease and the amount of computation required. For some geometric type-1 fuzzy logic systems may provide an ideal solution. For others the existing discrete model may be perfectly adequate.

This Chapter also discussed problems associated with using discrete fuzzy sets. The problems of computational complexity are significantly more important from type-2 fuzzy sets The operations on type-2 fuzzy sets are derived using the extension principle. This is what causes the computational complexity of the type-2 operations to be so high. Consider the example of a binary operation $\oplus$ with a computational cost c. Let there be two type-1 fuzzy sets A and B, and two equivalent type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$. The computational cost of $A \oplus B$ at a single point x is c. Applying the extension principle to $\oplus$ results in much larger computational cost. The cost of $\widetilde{A} \oplus \widetilde{B}$ at the same point x is $(c+t)mn$, where t is the cost of a t-norm operator, m is the number of points in $\mu_{\widetilde{A}}(x)$ and n is the number of points in $\mu_{\widetilde{B}}(x)$. This example applies to any binary operator, including 'and'

and 'or'. Work by Karnik and Mendel and the novel work reported in Section 4.1 significantly reduce the computational burden of the join and meet operations. Type-reduction however, remains a significant barrier to the use of type-2 fuzzy logic in real world systems. Greenfield *et al* have given a method for type-reduction which can have a far lower computational burden. This method is however, non-deterministic and the accuracy of the method is, to some degree, dependent on chance.

The following Chapter defines the geometric model of type-2 fuzzy logic. This model directly extends the type-1 model presented in this Section. Geometric models of both type-2 interval and generalised type-2 fuzzy sets and systems are given.

Chapter 4

Type-2 Geometric Fuzzy Logic Systems

This Chapter presents the novel geometric approach to type-2 fuzzy logic systems. The method presented here builds directly on the those presented in the previous Chapter. The first part of this Chapter presents the geometric approach to type-2 interval systems before moving on to generalised geometric type-2 fuzzy systems. The research presented in this Chapter forms the theoretical core of the thesis.

4.1 New Work on Join and Meet

The Karnik and Mendel (2001*b*) definitions for the join and meet operations require that the secondary membership functions being processed are both normal and convex. However, type-2 fuzzy sets can contain secondary membership functions that are non-normal. A non-normal fuzzy set is one that does not contain a single element with a membership grade of one. Consider the type-2 fuzzy set $\widetilde{A}$ with triangular primary and secondary memberships depicted in Figure 4.1. It is quite possible that the first and last secondary memberships in such a set will be non-normal, as they are with $\widetilde{A}$. The issue of convexity is less of a problem. A secondary membership function can be thought of as a type-1 fuzzy number in $[0, 1]$, such as *about 0.34*. Thus, it is difficult to see why a non-convex secondary membership function would be used to describe a type-2 membership grade. A non-convex secondary would represent a number like *about 0.14 or about 0.42*. This suggests a more fundamental issue with representation of the concept with a single set. It may be that two or more sets are required to model that concept. This is a separate issue from the argument presented by Garibaldi *et al* (2004) who advocate the use of non-convex membership functions in specific applications. The use of non-convex primary membership functions is in line with Garibaldi *et al*. It is only the use of non-convex secondary membership functions which causes the optimised join and meet operations to fail.

For these reasons novel optimised methods for the join and meet of non-normal secondary mem-

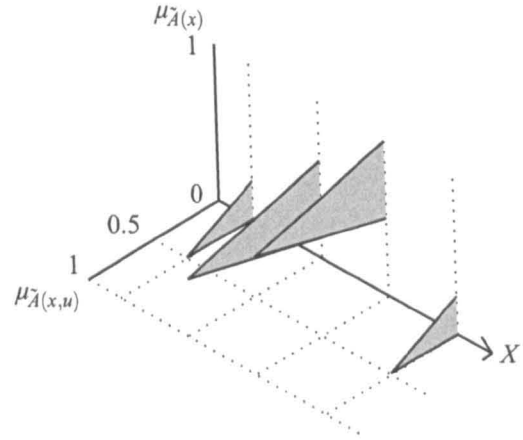


Fig. 4.1. The Type-2 Fuzzy Set $\tilde{A}$ with Non-Normal Secondary Membership Functions.

bership functions are now presented. These methods extend those presented by Karnik and Mendel (2001*b*). As with Karnik and Mendel these methods require that the possible operators must be limited to the minimum t-norm and maximum t-conorm. This work has been published by the author in a number of places (Coupland & John 2004*a*, Coupland & John 2005*b*, Coupland & John 2006*b*). Consider the two convex, but possibly non-normal secondary membership functions $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$. Firstly the join for a single point θ will be given.

For every pair of points (v,w), such that $\mu_{\tilde{A}}(v) \neq \emptyset$ and $\mu_{\tilde{B}}(w) \neq \emptyset$, take the maximum of v and w and the minimum of their membership grades, so that $(v \vee w) \mapsto (\mu_{\tilde{A}}(x,v) \wedge \mu_{\tilde{B}}(x,w)) \in \mu_{\tilde{A} \sqcup \tilde{B}}(x)$. When more than one pair of points (v,w) have the same maximum values take the pair with the maximum membership grade i.e, that is take the union. For any value θ such that $\theta \in [0,1]$ and $\theta \mapsto \mu_{\tilde{A} \sqcup \tilde{B}}(x,\theta) \in \mu_{\tilde{A} \sqcup \tilde{B}}(x)$, the set of all possible pairs (v,w) that can give θ as a result is $\{(v,w) \mid v \in (-\infty,\theta] \wedge w \in (-\infty,\theta]\}$. The value of $\mu_{\tilde{A} \sqcup \tilde{B}}(x,\theta)$ is given below:

$$\mu_{\tilde{A} \sqcup \tilde{B}}(x,\theta) = \left(\sup_{v \in (-\infty,\theta]} \mu_{\tilde{A}}(x,v) \wedge \mu_{\tilde{B}}(x,\theta) \right) \vee \left(\sup_{w \in (-\infty,\theta]} \mu_{\tilde{B}}(x,w) \wedge \mu_{\tilde{A}}(x,\theta) \right) \tag{4.1}$$

where $\sup_x \in [0,1]$ denotes the supremum operation over the points in $[0,1]$. θ is broken into three ranges: $\theta < \upsilon_1, \upsilon_1 \leq \theta < \upsilon_2$ and $\theta \geq \upsilon_2$. The point of departure from (Karnik & Mendel 2001*b*) is that $\mu_{\tilde{A}}(x,\upsilon_1)$ and $\mu_{\tilde{B}}(x,\upsilon_2)$ may not necessarily equal unity. Since both $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$ are convex when $\theta < \upsilon_1$, $\sup_{v \in (-\infty,\theta]} \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\theta)$ and $\sup_{w \in (-\infty,\theta]} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\theta)$. Substituting these values into 4.1 gives:

$$\mu_{\tilde{A} \sqcup \tilde{B}}(x,\theta) = \left(\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\theta)\right) \vee \left(\mu_{\tilde{B}}(x,\theta) \wedge \mu_{\tilde{A}}(x,\theta)\right) \tag{4.2}$$

which reduces to:

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\theta) \wedge \mu_{\tilde{A}}(x,\theta) \tag{4.3}$$

When $\upsilon_1 \leq \theta < \upsilon_2$, $\sup_{v\in(-\infty,\theta]} \cdot \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\upsilon_1)$ and $\sup_{w\in(-\infty,\theta]} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\theta)$. Substituting these values into 4.1 gives

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = \left(\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\theta)\right) \vee \left(\mu_{\tilde{B}}(x,\theta) \wedge \mu_{\tilde{A}}(x,\theta)\right) \tag{4.4}$$

Since $\widetilde{A}(x,\upsilon_1) \geq \widetilde{A}(x,\theta)$ this can be rewritten as:

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = \mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\theta) \tag{4.5}$$

When $\upsilon_2 \geq \theta$, $\sup_{v\in(-\infty,\theta]} \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\upsilon_1)$ and $\sup_{w\in(-\infty,\theta]} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\upsilon_1)$. Substituting these values into 4.1 gives:

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\theta)) \vee (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{4.6}$$

Since $\mu_{\tilde{A}}(x,\upsilon_1) > \mu_{\tilde{A}}(x,\theta)$ and $\mu_{\tilde{B}}(x,\upsilon_2) > \mu_{\tilde{B}}(x,\theta)$ this can be rewritten as (for demonstration of this see Appendix A.):

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\theta) \vee \mu_{\tilde{B}}(x,\theta)) \wedge (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{4.7}$$

Putting equations 4.3, 4.5 and 4.7 together gives a final efficient method for finding the join of two non-normal, convex secondary membership functions:

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = \begin{cases} \mu_{\tilde{A}}(x,u) \wedge \mu_{\tilde{B}}(x,u), & u < \upsilon_1, \\ \mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,u), & \upsilon_1 \leq u < \upsilon_2, \\ (\mu_{\tilde{A}}(x,u) \vee \mu_{\tilde{B}}(x,u)) \wedge (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\upsilon_2)), & u \geq \upsilon_2. \end{cases} \tag{4.8}$$

A similar progression can be made for the meet of two secondary membership functions. Again, the available t-norms and t-conorms are limited to minimum and maximum. The meet of two secondary membership functions $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$ for a single point θ will now be given.

For every pair of points (v,w), such that $\mu_{\tilde{A}}(v) \neq \emptyset$ and $\mu_{\tilde{B}}(w) \neq \emptyset$, take the maximum of v and w and the minimum of their membership grades, so that $(v \vee w) \mapsto (\mu_{\tilde{A}}(x,v) \wedge \mu_{\tilde{B}}(x,w)) \in \mu_{\tilde{A}\sqcap\tilde{B}}(x)$. When more than one pair of points (v,w) have the same maximum values take the pair with the higher membership grade i.e., take the union. For any value θ such that $\theta \in [0,1]$ and $\theta \mapsto \mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) \in \mu_{\tilde{A}\sqcap\tilde{B}}(x)$, the set of all possible pairs (v,w) that can give θ as a result is $\{(v,w) \mid v \in$

$[\theta,\infty) \wedge w \in [\theta,\infty)\}$. This gives the following:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = \left(\sup_{v\in[\theta,\infty)} \mu_{\tilde{A}}(x,v) \wedge \mu_{\tilde{B}}(x,\theta)\right) \vee \left(\sup_{w\in[\theta,\infty)} \mu_{\tilde{B}}(x,w) \wedge \mu_{\tilde{A}}(x,\theta)\right) \tag{4.9}$$

Again θ is broken down into the same three ranges: $\theta < \upsilon_1, \upsilon_1 \leq \theta < \upsilon_2$ and $\theta \geq \upsilon_2$. When $\theta < \upsilon_1$, $\sup_{v\in[\theta,\infty)} \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\upsilon_1)$ and $\sup_{w\in[\theta,\infty)} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\upsilon_2)$. Substituting these values into 4.9 gives:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\theta)) \vee (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{4.10}$$

Since $\mu_{\tilde{A}}(x,\upsilon_1) > \mu_{\tilde{A}}(x,\theta)$ and $\mu_{\tilde{B}}(x,\upsilon_2) > \mu_{\tilde{B}}(x,\theta)$ this can be rearranged as (demonstration of this of this is similar to that given in Appendix A:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\theta) \vee \mu_{\tilde{B}}(x,\theta)) \wedge (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{4.11}$$

When $\upsilon_1 \leq \theta < \upsilon_2$, $\sup_{v\in[\theta,\infty)} \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\theta)$ and $\sup_{w\in[\theta,\infty)} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\upsilon_2)$. Substituting these values into 4.9 gives:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\theta)) \vee (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{4.12}$$

Since $\mu_{\tilde{B}}(x,\upsilon_2) \geq \mu_{\tilde{B}}(x,\theta)$ which can be rewritten as:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = \mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\upsilon_2) \tag{4.13}$$

when $\upsilon_2 \leq \theta$, $\sup_{v\in[\theta,\infty)} \mu_{\tilde{A}}(x,v) = \mu_{\tilde{A}}(x,\theta)$ and $\sup_{w\in[\theta,\infty)} \mu_{\tilde{B}}(x,\theta) = \mu_{\tilde{B}}(x,\theta)$. Substituting these values into 4.9 gives:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\theta)) \vee (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\theta)) \tag{4.14}$$

which reduces to:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = \mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\theta) \tag{4.15}$$

Putting equations 4.11, 4.13 and 4.15 together gives a final efficient method for finding the meet of two non-normal, convex secondary membership functions:

$$\mu_{\tilde{A}\sqcap\tilde{B}}(x,\theta) = \begin{cases} (\mu_{\tilde{A}}(x,u) \vee \mu_{\tilde{B}}(x,u)) \wedge (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\upsilon_2)), & u < \upsilon_1, \\ \mu_{\tilde{A}}(x,u) \wedge \mu_{\tilde{B}}(x,\upsilon_2), & \upsilon_1 \leq u < \upsilon_2, \\ \mu_{\tilde{A}}(x,u) \wedge \mu_{\tilde{B}}(x,u), & u \geq \upsilon_2. \end{cases} \tag{4.16}$$

Equations 4.8 and 4.16 give an optimised method for performing the join and meet for non-normal secondary membership functions. Such methods will be practically useful, removing the require-

ment for normality, which many secondary memberships of discrete type-2 fuzzy sets fail to meet.

Expressions to compare the relative computational costs of novel join and meet operations with the other operations are now given. Earlier the computational cost of the non-optimised join and meet operations at a point x was given as $2MNt$ where M and N are the cardinality of the two secondary membership functions and t is the cost of a t-norm or t-conorm. To express the cost of the optimised techniques some additional terms need to be introduced. Let $\#A$ be the cardinality of a set A and the domain of a set A be given by $DOM\,A$. Let p be the preprocessing cost of finding apex points v_1 and v_2. Let α be the number of points in the secondary membership functions after v_2. Finally, let β be the number of points in the secondary membership functions before v_1. The computational cost of the optimised join operation can now be given as:

$$\mathsf{COST}_{join} = \left(\#(DOM\,\mu_{\tilde{A}} \cup DOM\,\mu_{\tilde{B}}) - \alpha\right) t + 3t\alpha + p \tag{4.17}$$

Similarly the computational cost of the optimised meet operation can be given as:

$$\mathsf{COST}_{meet} = \left(\#(DOM\,\mu_{\tilde{A}} \cup DOM\,\mu_{\tilde{B}}) - \beta\right) t + 3t\beta + p \tag{4.18}$$

The optimised join and meet reduce the computational cost by removing redundancies in the operations. The optimised join and meet only iterate over the combined domain of the secondary membership functions once. The standard join and meet iterate over this domain a number of times.

Both methods have advantages and disadvantages. The relative computational speed of the two is compared empirically in Chapter 5. The non-optimised methods have unchanging, easily calculated computational cost, and therefore a predictable, consistent execution speed. The optimised methods are likely to be faster, although the actual speed will depend on the form of secondary membership functions being processed. In some cases where it is useful to be able to know the execution speed beforehand, such as a slow but real-time system, the non-optimised methods may represent a better choice. However, generally speaking, the optimised method will be faster, and therefore more useful in the development of practical applications.

This Section has given novel extensions to the optimised join and meet which allow those operations to be conducted on non-normal secondary membership functions. This is enables the use of these methods in real world applications. The following Section defines geometric type-2 interval fuzzy systems.

4.2 Geometric Type-2 Interval Fuzzy Logic Systems

Type-2 interval fuzzy systems have been shown to outperform type-1 systems in a number of applications (Hagras 2004, Karnik & Mendel 1999, Liang & Mendel 2000*a*, Liang & Mendel 2001), giving smoother, more robust, more correct control performance, increased classification levels or improve predictive ability. This may be because interval type-2 fuzzy sets increase the capacity of a fuzzy models to handle uncertainty.

This Section introduces the geometric type-2 interval fuzzy sets and systems. Like the type-1 geometric fuzzy set the geometric interval model provides increased accuracy. A geometric defuzzification algorithm is provided which eliminates the need for type-reduction. The following Section gives the definition of a geometric type-2 interval fuzzy set. This work was presented by the author in Coupland and John (2005*a*).

4.2.1 Geometric Type-2 Interval Fuzzy Sets

To begin handling uncertainty in a geometric fuzzy system a fuzzy set that provides some model of uncertainty is needed. The simplest discrete fuzzy set that models uncertainty is the type-2 interval fuzzy set. As was seen in the previous Chapter, type-2 interval fuzzy sets model uncertainty with a uniform distribution. Type-2 interval fuzzy logic systems only need to model the end points of the interval sets as upper and lower membership functions. The geometric type-2 interval fuzzy set models the upper and lower membership functions as piecewise linear functions.

Definition 4.1 *A geometric interval type-2 fuzzy set $\widetilde{A}$ is characterised by two membership functions, upper and lower denoted by $\overline{\widetilde{A}}$ and $\underline{\widetilde{A}}$ respectively. Each membership function consists of a series of ordered vertices of length n connected by line segments to form a function over a continuous domain. Every vertex $\overline{(x,y)}$ or $\underline{(x,y)}$ from either of the functions of the fuzzy set $\widetilde{A}$ over the continuous domain X consists of a value x such that $x \in X$ and a y value such that $y \in [0,1]$, i.e.*

$$\mu_{\widetilde{A}}(x) = [\underline{y}, \overline{y}] \quad ; \mu_{\underline{\widetilde{A}}}(x) = \underline{y} \quad ; \mu_{\overline{\widetilde{A}}}(x) = \overline{y} \tag{4.19}$$

$$\underline{\widetilde{A}} = \{\underline{(x_1,y_1)}, \underline{(x_2,y_2)}, \ldots, \underline{(x_n,y_n)} \mid x_i \in X, y_i \in [0,1], x_i < x_{i+1} \, \forall i \in [1,n]\} \tag{4.20}$$

$$\overline{\widetilde{A}} = \{\overline{(x_1,y_1)}, \overline{(x_2,y_2)}, \ldots, \overline{(x_n,y_n)} \mid x_i \in X, \forall y_i \in [0,1], \forall x_i < x_{i+1}\} \tag{4.21}$$

Where $\overline{(x_i,y_i)}$ is the i^{th} vertex from the PLF that represents the upper membership function of $\widetilde{A}$ and $\underline{(x_i,y_i)}$ is the i^{th} vertex from the PLF that represents the lower membership function of $\widetilde{A}$.

Two example geometric interval type-2 fuzzy sets are given in Figures 4.2 (a) and (b). Figure 4.2 (a) depicts a triangular set with uncertain endpoints. Figure 4.2 (b) depicts a Gaussian set with

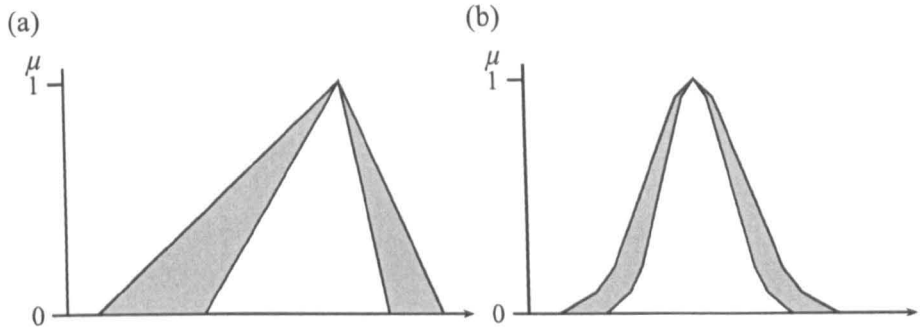


Fig. 4.2. (a) A Triangular Geometric Interval Fuzzy Set. (b) A Gaussian Geometric Interval Fuzzy Set.

an uncertain standard deviation using nine vertices in each of the upper and lower membership functions.

Definition 4.1 states that a geometric interval type-2 fuzzy set is defined by upper and lower membership functions that are piecewise linear. In effect upper and lower membership functions of such sets can each be viewed as geometric type-1 fuzzy sets. In the next Section the logical operations needed for inferencing to be performed with geometric interval type-2 fuzzy sets are defined. The analogy with geometric type-1 fuzzy sets are exploited when giving these operations.

4.2.2 Geometric Type-2 Interval Fuzzy Inferencing

Geometric type-2 interval fuzzy inferening is a direct extension of geometric type-1 fuzzy inferencing. An interval type-2 fuzzy set can be modelled as two independent type-1 fuzzy sets. These two type-1 sets represent the upper and lower bound of the interval type-2 set. Geometric interval type-2 FLS can be modelled by two geometric type-1 FLS. The logical operators for a geometric type-2 interval fuzzy set can be derived from the geometric type-1 operations on these upper and lower bound type-1 fuzzy sets.

4.2.2.1 Fuzzification

Fuzzification finds the value of the membership grade of a given input in a fuzzy set. The value of a point in an type-2 interval set is a crisp interval set, given by the end points of that crisp set. With a geometric type-2 interval fuzzy set these end points are found by taking the membership grade of the input in the upper and lower bounds. The membership grade of a geometric type-2

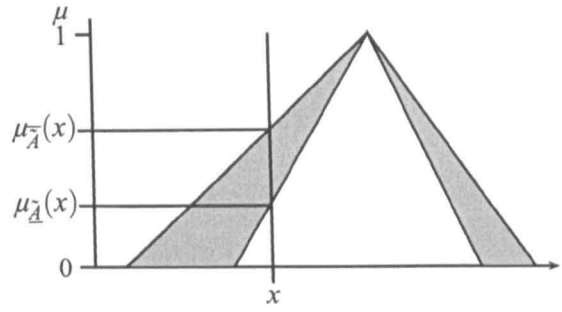


Fig. 4.3. The Membership Grade of a Point x in the Geometric Interval Fuzzy Set $\widetilde{A}$.

interval fuzzy set at a point x in the set $\widetilde{A}$ is given by $[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)]$ where

$$\mu_{\underline{\widetilde{A}}}(x) = \begin{cases} 0 & ; x \leq \underline{x_1} \text{ or } \underline{x_n} \leq x \\ \underline{y_i} & ; x = \underline{x_i} \\ \underline{y_i} + \frac{x - \underline{x_i}}{\underline{x_{i+1}} - \underline{x_i}} (\underline{y_{i+1}} - \underline{y_i}) & ; \underline{x_i} < x < \underline{x_{i+1}} \end{cases} \tag{4.22}$$

and

$$\mu_{\overline{\widetilde{A}}}(x) = \begin{cases} 0 & ; x \leq \overline{x_1} \text{ or } \overline{x_n} \leq x \\ \overline{y_i} & ; x = \overline{x_i} \\ \overline{y_i} + \frac{x - \overline{x_i}}{\overline{x_{i+1}} - \overline{x_i}} (\overline{y_{i+1}} - \overline{y_i}) & ; \overline{x_i} < x < \overline{x_{i+1}} \end{cases} \tag{4.23}$$

From Coupland and John (2005*a*)

Figure 4.3 depicts the value of $[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)]$ in an example geometric type-2 interval fuzzy set $\widetilde{A}$. Having found the value of a membership grade of a point in a geometric type-2 interval fuzzy set, the next stage is to combine one or more of these values with logical operators.

4.2.2.2 Combination of Antecedents

The logical combination of antecedents in a geometric type-2 interval fuzzy rule is identical to the non-geometric case. The conjunction of two interval sets $[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)]$ $[\mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{B}}}(x)]$is given by the meet ($\sqcap$) operation from (Mendel 2001*b*).

$$[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)] \sqcap [\mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{B}}}(x)] = [\mu_{\underline{\widetilde{A}}}(x) \star \mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{A}}}(x) \star \mu_{\overline{\widetilde{A}}}(x)] \tag{4.24}$$

Where $\star$ is a t-norm, usually the minimum or the product. The disjunction of two interval sets $[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)]$ $[\mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{B}}}(x)]$is given by the join ($\sqcup$) operation also from (Mendel 2001*b*).

$$[\mu_{\underline{\widetilde{A}}}(x), \mu_{\overline{\widetilde{A}}}(x)] \sqcup [\mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{B}}}(x)] = [\mu_{\underline{\widetilde{A}}}(x) \bullet \mu_{\underline{\widetilde{B}}}(x), \mu_{\overline{\widetilde{A}}}(x) \bullet \mu_{\overline{\widetilde{A}}}(x)] \tag{4.25}$$

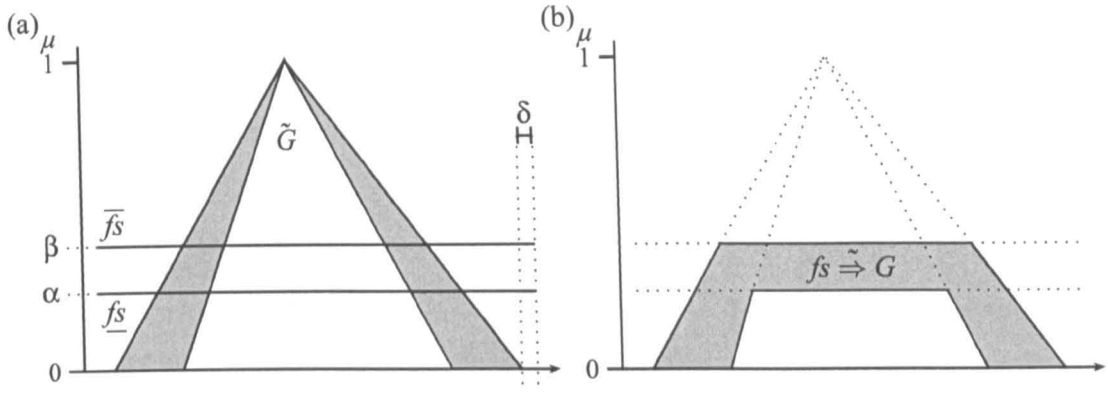


Fig. 4.4. (a) The Consequent Interval Type-2 Fuzzy Set $\tilde{G}$ and the Line Segments $\underline{fs}$ and $\overline{fs}$. (b) The Result of $fs \Rightarrow \tilde{G}$.

Where • is a t-conorm usually the maximum. Like geometric type-1 fuzzy logic this is the only processing stage where a choice of t-norms and t-conorms is available. All other processing stages require that only minimum and maximum are used. Once the value of the rule antecedent has been calculated, the implication of this value on a given consequent must be found.

4.2.2.3 Implication

Each geometric interval fuzzy rule has an antecedent value $[\alpha, \beta]$ with a lower (α) and upper (β) value and a consequent with corresponding upper and lower membership functions $\mu_{\underline{\tilde{G}}}$ and $\mu_{\overline{\tilde{G}}}$. To calculate the implied rule consequent $[\alpha, \beta] \Rightarrow \tilde{G}$ two type-1 fuzzy implications operations are performed concurrently. The lower bound of $[\alpha, \beta] \Rightarrow \tilde{G}$ is calculated by finding type-1 geometric fuzzy implication of α on $\mu_{\underline{\tilde{G}}}$. The upper bound is given by the type-1 geometric fuzzy implication of β on $\mu_{\overline{\tilde{G}}}$. Let $\underline{fs}$ be a line segment $(\underline{x_0} - \delta, \alpha)(\underline{x_n} + \delta, \alpha)$ and let $\overline{fs}$ be a line segment $(\overline{x_0} - \delta, \beta)(\overline{x_n} + \delta, \beta)$. The geometric type-1 implication algorithm can then by applied to $\underline{fs}$ and $\mu_{\underline{\tilde{G}}}$ to give $\mu_{\underline{fs \tilde{\Rightarrow} G}}$ and to $\overline{fs}$ and $\mu_{\overline{\tilde{G}}}$ to give $\mu_{\overline{fs \tilde{\Rightarrow} G}}$. The two geometric type-1 fuzzy sets $\mu_{\underline{fs \tilde{\Rightarrow} G}}$ and $\mu_{\overline{fs \tilde{\Rightarrow} G}}$ combine to give the geometric type-2 interval fuzzy set $\widetilde{fs \tilde{\Rightarrow} G}$. An example implication operation is depicted in Figure 4.4. Once all the consequent sets have been calculated they must be combined to give a final geometric type-2 interval fuzzy set which can be defuzzified.

4.2.2.4 Combination of Consequents

To combine two or more consequents the disjunction is taken. Geometric type-1 fuzzy systems use the modified Weiler-Atherton clipping algorithm for this. Geometric type-2 interval systems use the modified Weiler-Atherton algorithm to give the disjunction of upper and lower membership functions independently. Let there be two geometric type-2 interval fuzzy sets $\tilde{A}$ and $\tilde{B}$. The disjunction of the two $\tilde{A} \cup \tilde{B}$ has upper and lower membership functions $\overline{\tilde{A} \cup \tilde{B}}$ and $\underline{\tilde{A} \cup \tilde{B}}$. The

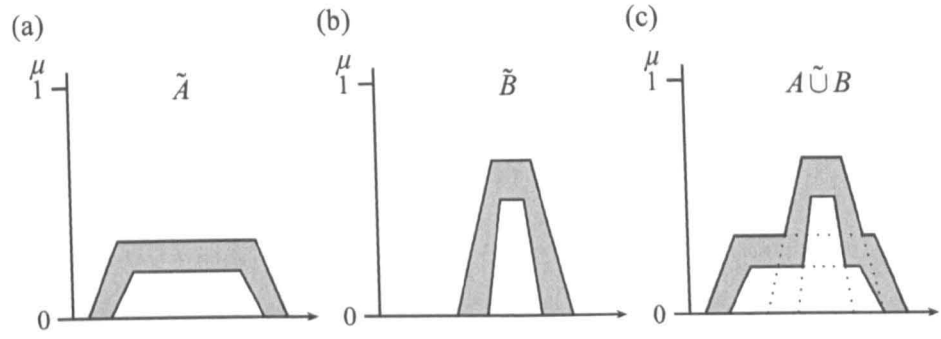


Fig. 4.5. (a) The Geometric Interval Type-2 Fuzzy Set $\tilde{A}$. (b) The Geometric Interval Type-2 Fuzzy Set $\tilde{B}$. (c) The Geometric Interval Type-2 Fuzzy Set $A \tilde{\cup} B$.

upper membership $\overline{A \tilde{\cup} B}$ is given by applying the modified Weiler-Atherton clipping algorithm to the upper membership functions $\overline{\tilde{A}}$ and $\overline{\tilde{B}}$. The lower membership $\underline{A \tilde{\cup} B}$ is given by applying the modified Weiler-Atherton clipping algorithm to the lower membership functions $\underline{\tilde{A}}$ and $\underline{\tilde{B}}$. Figure 4.5 depicts the disjunction of two example sets $\tilde{A}$ and $\tilde{B}$. Once the consequent sets have all been combined to final resultant set, that set must be defuzzified.

4.2.2.5 Defuzzification

Output processors in discrete type-2 interval fuzzy logic systems typically use the type-reduction method to arrive at an interval set. This interval set represents the spread of possibilities of the crisp output value. The centre of this interval set is normally used to give the final crisp output value. The type-reduction procedure may be derived by applying the extension principle to type-1 defuzzification algorithms.

Type-reduction cannot be directly applied to geometric systems. Type-reduction, whether using the iterative method or not, finds two embedded sets that have the lowest and highest defuzzified values. Since there are an infinite number of embedded sets in a geometric type-2 interval fuzzy set the search for the two with the highest and lowest defuzzified values would be infinite. One solution to this problem is to discretise the final combined consequent set and type-reduce it, however the computational expense of type-reduction would be a significant impairment on such a system. The novel solution suggested in this thesis is to extend the geometric type-1 defuzzier defined in Section 3.3.2.4.

A geometric type-2 interval fuzzy set can be thought of as a closed, non self-intersecting polygon. The polygon is formed by ordering the membership function vertices $\overline{(x_0,y_0)}$, $\overline{(x_1,y_1)}$, ..., $\overline{(x_n,y_n)}$, $\underline{(x_n,y_n)}$, $\underline{(x_{n-1},y_{n-1})}$, ..., $\underline{(x_0,y_0)}$. The polygon centroid algorithm given by equation 3.13 can be used to give the centroid of the geometric type-2 interval fuzzy set without the need for type-

reduction. This geometric algorithm also has a discrete equivalent given by equation 4.26.

$$c_{\tilde{A}} = \frac{\sum_{i=0}^{m} \bar{\mu}_{\tilde{A}}(x_i)x_i + \sum_{i=0}^{n} \underline{\mu}_{\tilde{A}}(x_i)x_i}{\sum_{i=0}^{m} \bar{\mu}_{\tilde{A}}(x_i) + \sum_{i=0}^{n} \underline{\mu}_{\tilde{A}}(x_i)} \tag{4.26}$$

Adapted from Coupland and John (2006*b*)

where m and n are respectively the number of discrete points in the upper and lower membership functions of the discrete type-2 interval fuzzy set $\tilde{A}$. This equation is the first use of the geometric approach in defining a discrete fuzzy operation. The usefulness of the novel discrete operation is explored in more detail by Coupland and John (2006).

4.2.3 Discussion

This Section has given the details of geometric type-2 interval fuzzy logic sets and systems. The geometric model exploits the fact that an interval set is defined by upper and lower boundary points. In effect the upper and lower boundaries are each modelled independently by geometric type-1 fuzzy sets. These two boundaries remain independent until the defuzzification stage. This means the boundary sets can be processed concurrently. Increasing the scope for parallel processing gives the potential for increased hardware execution speeds. The type-2 interval geometric fuzzy model is constrained by the same three limitations as the type-1 geometric fuzzy model, these are:

- The upper and lower membership functions must be piecewise linear;
- the implication operator must use the minimum t-norm, and
- the 'or' operator must use the maximum t-conorm.

As with the type-1 geometric model, these do not represent significant restrictions.

The output processing for geometric type-2 interval fuzzy systems represents a completely different approach from discrete systems. Type-reduction is derived from type-1 methods using the extension principle and has a high computational cost, however, the iterative method greatly reduces the redundancy in this technique and hence the computational cost. Although the iterative method is fast, the amount of computation needed is not known before the algorithm is performed. This contrasts with the geometric defuzzifier where the amount of computation is a directly proportional to the number of vertices in the membership functions. Again this feature may prove to be advantageous for hardware implementation. One of the advantages to using type-reduction is that the result gives a measure of the uncertainty propagated through the system, an interval

Metric	Geometric Defuzzifier	Iterative Method	Type-Reduction
Computational Cost	Low. Known beforehand.	Low to medium. Upper boundary known beforehand.	High. Known beforehand.
Uncertainty Measure	None.	Interval set.	Interval set.

Table 4.1. A Summary of the Differing Qualities of Type-Reduction and Geometric Defuzzification

possibility distribution of the defuzzified value. Since this is a uniform distribution the interval set end points can be thought of as the boundaries of the uncertainty associated with the result. A drawback of the geometric defuzzifier is that it gives no measure or indication of the amount of uncertainty that has come through the system - a single defuzzified value is all that is produced. How much of a limitation this is will depend on the application area. In a decision making system such as a credit rating application or a medical diagnosis system, knowledge of uncertainty is likely to be of significant benefit. For control applications uncertainty distributions are likely to be of little use as only a single defuzzified value can be utilised by the actuators. These points are summarised in Table 4.1.

The geometric defuzzifier represents a significant departure from the type-reduction approach. Type-reduction is based on the notion that there is a single type-1 fuzzy system that under perfect conditions (with zero uncertainty present in the inputs, outputs or knowledge of the system) would perform exactly the same as a type-2 fuzzy system. The membership functions of fuzzy sets in this single type-1 fuzzy system are assumed to be between the lower and upper boundaries of the interval sets. The guiding principle behind this method is that without uncertainty a type-2 system performs as a type-1 system would:

> "When all sources of uncertainty disappear, a type-2 FLS must reduce to a comparable type-1 FLS."
>
> (Mendel 2001*b*) Page 11

Geometric type-2 interval fuzzy systems break a literal interpretation Mendel's fundamental design requirement. When uncertainty is zero the upper and lower bounds of an type-2 interval fuzzy set are identical. When this happens the geometric defuzzifier will not work. The geometric defuzzifier finds the centroid of the polygon formed by the upper and lower bounds of the set. If the upper and lower bounds are identical then this polygon has no area and therefore has no centroid. This breaks a literal interpretation of Mendel's design requirement. When the upper and lower bounds of a type-2 interval fuzzy set are identical the centroid is given by applying the type-1 geometric defuzzifier to either the upper or lower boundary. A more liberal interpretation of Mendel's principle may consider this to approach to meet this design requirement. This liberal interpretation of Mendel's principle may be read:

"When all sources of uncertainty disappear, type-1 methods should be applied in the place of the type-2 methods."

The geometric defuzzifier also produces some unexpected results where compared directly to type-reduction. The end points of the type-reduced interval of uncertainty correspond to two embedded type-1 fuzzy sets contained in the type-2 interval fuzzy set. The two embedded sets used to calculate these endpoints tend not to be characteristic of the set being defuzzified. Consider the two embedded sets $\widetilde{F_e^{C_l}}$ and $\widetilde{F_e^{C_r}}$ depicted in Figures 4.6(a) and (b). These are the two embedded sets

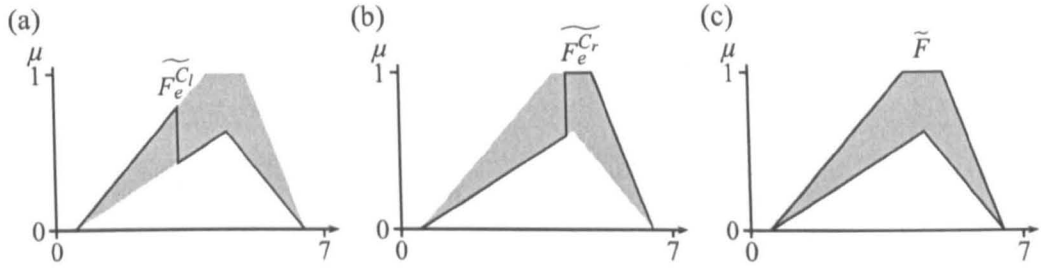


Fig. 4.6. (a) The Embedded Set $\widetilde{F_e^{C_l}}$. (b) The Embedded Set $\widetilde{F_e^{C_r}}$. (c) The Type-2 Interval Fuzzy Set $\widetilde{F}$.

used to calculate the centroid of the type-2 interval fuzzy set $\widetilde{F}$ depicted in Figure 4.6(c). The set $\widetilde{F}$ has a clear triangular form. Neither embedded set used to calculate the type-reduced set share this triangular form. The linguistic meaning of the embedded sets would therefore be quite different from the linguistic meaning of $\widetilde{F}$. The geometric approach does not consider these two embedded type-1 fuzzy sets when calculating the centroid. The geometric approach considers the entire area encompassed by the upper and lower bounds. This can lead to the geometric centroid of a geometric set lying outside the interval of uncertainty of the discrete equivalent. Consider the type-2 interval fuzzy sets $\widetilde{A}$, $\widetilde{B}$ and $\widetilde{C}$ depicted in Figures 4.7 (a), (b) and (c) respectively. The sets $\widetilde{A}$ and $\widetilde{B}$ represent typical outputs from a type-2 interval fuzzy logic system. The set $\widetilde{C}$ represents an unlikely but theoretically possible interval set. Table 4.2 gives the type-reduced, geometric and discrete centroids of these three sets. The discrete centroid is given by equation 4.26. Type-reduction was performed on the discrete set where each set was discretised along the domain at intervals of 0.5 beginning at 0.5. The geometric centroid of the more typical sets falls close to the type-reduced centroid as does the discrete centroid. The geometric centroid of $\widetilde{C}$ is not only significantly different from the type-reduced centroid but falls outside the type-reduced interval of uncertainty. This may cause some developers to reject the geometric defuzzifier, however the criticisms of the type-reduction given in this Section should also be taken into account. The discrete centroid of $\widetilde{C}$ also falls outside the interval of uncertainty. The discrete centroid is greater then the interval whereas the geometric is below. This is surprising since the discrete centroid is based on the geometric operation. The geometric defuzzifier is fast, predictable and makes use of all the information used to describe an type-2 interval fuzzy set. These qualities make the geometric

Set	TR Interval	TR Centroid	Geometric Centroid	Discrete Centroid
$\widetilde{A}$	[1.72,2.35]	2.03	2.10	2.01
$\widetilde{B}$	[1.76,2.19]	1.98	1.94	2.05
$\widetilde{C}$	[1.4,1.56]	1.48	0.96	1.68

TR = Type-reduced.

Table 4.2. A Comparison of the Defuzzified Values of $\widetilde{A}$, $\widetilde{B}$ and $\widetilde{C}$.

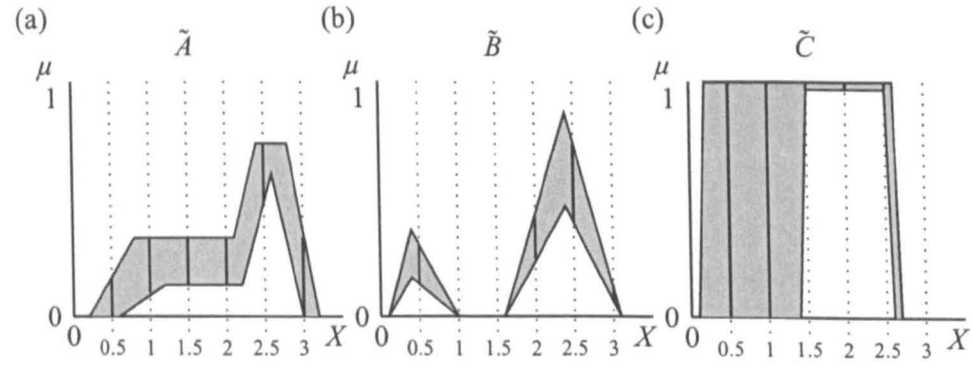


Fig. 4.7. Three Geometric Interval Type-2 Fuzzy Sets (a) $\widetilde{A}$, (b) $\widetilde{B}$ and (c) $\widetilde{C}$.

defuzzifier a useful operation.

This Section has given a complete description of a geometric type-2 interval fuzzy logic system, the differences between the geometric and discrete operations have been explored. The next Section discusses geometric type-2 fuzzy logic sets and systems.

4.3 Geometric Type-2 Fuzzy Logic Systems

This Section builds on the previous two sections, utilising the 3-dimensional geometry discussed earlier to give a complete geometric model of a type-2 FLS. Of particular interest is the type-2 geometric defuzzifier. This operation significantly reduces the computational cost of defuzzification of a type-2 fuzzy set, considerably decreasing the overall computational cost of a type-2 FLS. The work presented in this Section represents a major contribution to the field of type-2 fuzzy logic. To this end, De Montfort University have applied to the UK patent office for a patent protecting the intellectual property rights of the work given in this Section. A copy of the patent application is provided on the supplementary materials CD provided with this thesis.

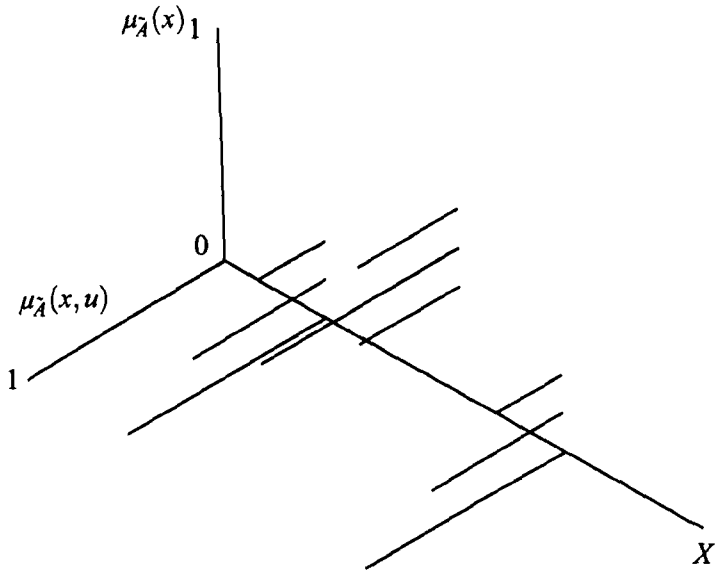


Fig. 4.8. The Type-2 Fuzzy Set $\tilde{A}$.

4.3.1 Geometric Type-2 Fuzzy Sets

A discrete type-2 fuzzy set is made up of a number of points in 3-dimensional space. Each point in the domain (x-dimension) of a type-2 fuzzy set has an associated secondary membership function. This secondary membership function maps primary membership grades (y-dimension) to secondary membership grades (z-dimension). Both primary and secondary grades take a value in between zero and one. When a type-2 fuzzy set is discretised a number of points from a three dimensional surface are used to approximate that surface. Consider the example type-2 fuzzy set $\tilde{A}$ depicted in Figure 4.8 which consists of nine discrete points. The three dimensional surface from which $\tilde{A}$ was discretised is depicted in Figure 4.9.

Geometric type-2 fuzzy sets also approximate the 3-dimensional surface of a type-2 fuzzy set. Recall from Section 3.2.6 that any 3-dimensional surface can be approximated by a mesh of connected triangles. Geometric type-2 fuzzy sets approximate type-2 fuzzy sets as triangular meshes. The 3-dimensional surface is separated into two parts, an upper and a lower surface. Each of these are modelled independently as upper and lower triangular meshes. The point where the upper and lower surfaces meet are the apex points from each of the secondary memberships. This is done to simplify the logical operations that will be defined in later Sections of this Chapter.

Definition 4.2 *A geometric type-2 fuzzy set consists of two surfaces, an upper surface ($\overline{\tilde{A}}$) and a lower surface ($\underline{\tilde{A}}$). Each surface is modelled by a mesh of triangular facets each consisting of three 3-dimensional vertices in a counter clockwise order, i.e.,*

$$\tilde{A} = (\underline{\tilde{A}}, \overline{\tilde{A}}) \tag{4.27}$$

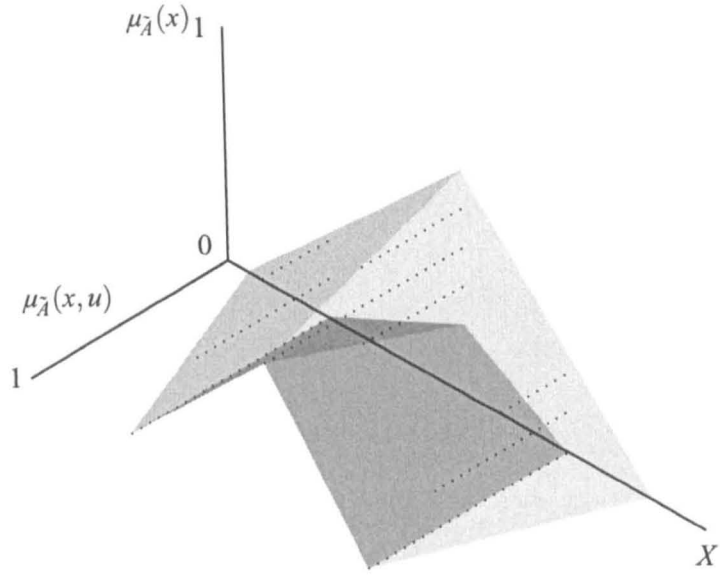


Fig. 4.9. The Surface of the Type-2 Fuzzy Set $\widetilde{A}$.

where

$$\underline{\widetilde{A}} = \{(\underline{p_1}, \underline{q_1}, \underline{r_1}), (\underline{p_2}, \underline{q_2}, \underline{r_2}), \ldots, (\underline{p_n}, \underline{q_n}, \underline{r_n})\} \tag{4.28}$$

and

$$\overline{\widetilde{A}} = \{(\overline{p_1}, \overline{q_1}, \overline{r_1}), (\overline{p_2}, \overline{q_2}, \overline{r_2}), \ldots, (\overline{p_n}, \overline{q_n}, \overline{r_n})\} \tag{4.29}$$

where $p_i = (p_i.x, p_i.y, p_i.z)$, $q_i = (q_i.x, q_i.y, q_i.z)$ *and* $r_i = (r_i.x, r_i.y, r_i.z)$ *such that* $p_i.x, q_i.x, r_i.x \in X$, $p_i.y, q_i.y, r_i.y, p_i.z, q_i.z, r_i.z \in [0,1]$ *and* $|(\mathbf{q_i} - \mathbf{p_i}) \times (\mathbf{r_i} - \mathbf{p_i})| > 0$.

An example geometric fuzzy set $\widetilde{A}$ is depicted in Figure 4.10. Both the upper and lower surface of $\widetilde{A}$ contain four triangular facets. The greater the number of facets used for the approximation the closer the approximation is to the actual surface.

4.3.2 Geometric Type-2 Fuzzy Logic

A geometric type-2 FLS has the same processing stages as any other FLS, fuzzification, combination of antecedents, implication, combination of consequents and defuzzification. Each of these will now be discussed in turn.

4.3.2.1 Fuzzification

The first stage in fuzzy inferencing is to find the membership grades of the inputs in the respective fuzzy sets. The membership grade of a type-2 fuzzy set is a type-1 fuzzy number with a domain

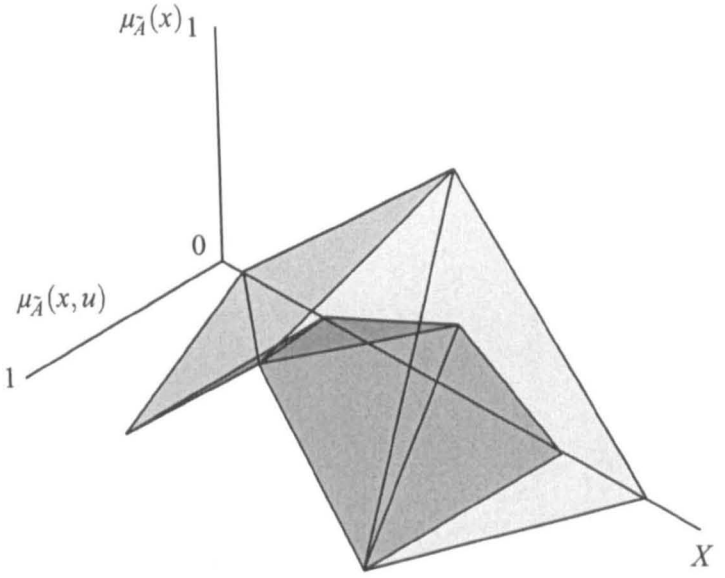


Fig. 4.10. The Geometric Type-2 Fuzzy Set $\widetilde{A}$.

which falls within the interval $[0,1]$. The membership grade of a geometric type-2 fuzzy set is a geometric type-1 fuzzy number with a domain which falls within the interval $[0,1]$, a geometric secondary membership function. To find this value a cross section is taken, a vertical slice through the upper and lower surface of the geometric type-2 fuzzy set. The lower surface of the set gives the left hand side and the upper surface gives the right hand side of the secondary membership function. The vertices in a geometric secondary are stored in two lists, one storing all points to the left of, and including, the apex point and one storing all points to the right of, and including, the apex point. This definition assumes that the secondary memberships are always convex. This allows the join and meet operations to be more easily defined. Limiting secondary memberships to be convex is not a significant limitation. As discussed in Chapter 2, it is difficult to envisage a context where a non-convex secondary membership function would be useful.

Triangles, in 3-dimensions, can be thought of being bounded by a cubic volume. Let each triangle in a type-2 fuzzy set by enclosed by a bounding cube defined by the two vertices l and r where $l = (min(p.x,q.x,r.x),\ min(p.y,q.y,r.y),\ min(p.z,q.z,r.z))$ and $r = (max(p.x,q.x,r.x),\ max(p.y,q.y,r.y),\ max(p.z,q.z,r.z))$. The bounding cube gives simple cubic description of the 3-dimensional area that the triangle lies in. For any geometric entity to interact with a triangle at least some part of the entity must lie inside that triangle's bounding cube. A triangle and its bounding cube is depicted in Figure 4.11(a). If the given domain value x falls outside this bounding cube for a particular triangle t then t does not contribute to the membership grade. If $x \in [l.x,r.x]$ then x must lie within the bounds of two of the three line segments. Once these two segments have been identified their parametric description is used to find the corresponding y and z values. These

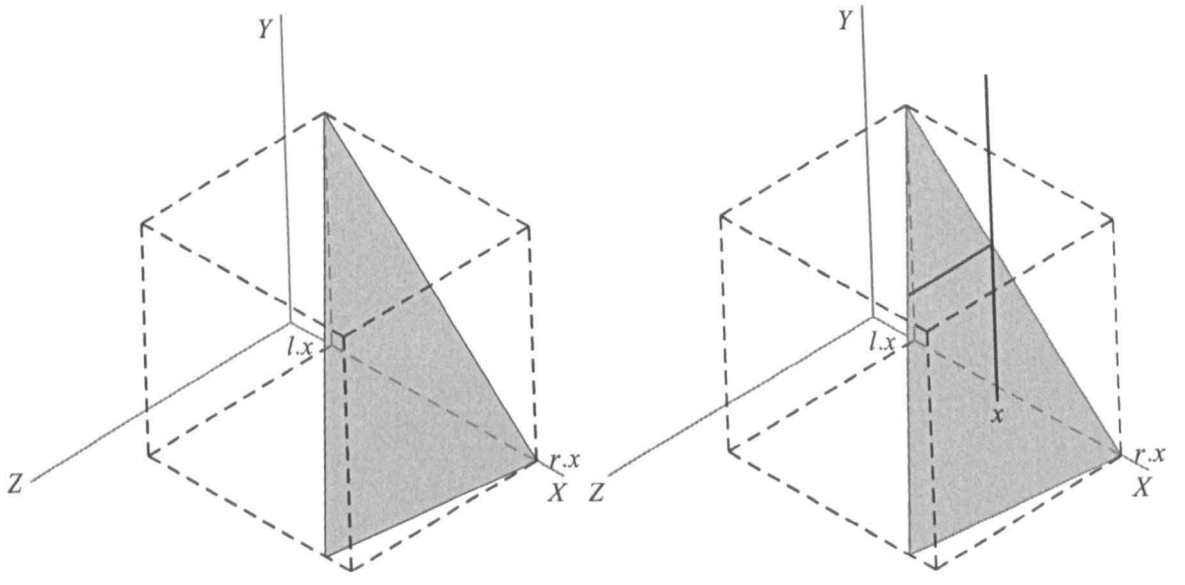


Fig. 4.11. (a) A Triangle and Bounding cube. (b) The Membership of x in a Triangle.

values on a 3-dimensional co-ordinate system are then transferred to a 2-dimensional co-ordinate system that is used when describing the a secondary membership function. The values obtained from the y-axis on the 3D model become values on the x-axis on the 2D model. The values obtained from the z-axis on the 3D model become values on the y-axis on the 2D model. All the triangles in the both surfaces are processed in this way. The resultant vertices form the secondary membership functions. Vertices identified in the lower surface form the left side vertices the secondary membership function. Vertices identified in the upper surface form the right side vertices the secondary membership function. It is important that secondary membership functions contain no vertical line segment. If any vertical segments are identified then the line should be altered very slightly so it is no longer exactly vertical. The algorithms that are applied to the membership functions will fail if the function contains vertical lines. Figure 4.11(b) depicts the segment formed by finding the vertices that lie on a triangle's edges a given point x. Figure 4.12 depicts this process when performed on an entire geometric type-2 fuzzy set. Algorithm 4.1 gives a formal definition of the fuzzification algorithm for a geometric type-2 fuzzy set that has been explored in this Section. The result of this algorithm is a geometric secondary membership function. The next Section describes how two geometric secondary memberships can be logically combined.

Algorithm 4.1 The Fuzzification Algorithm for a Geometric Type-2 Fuzzy Set.

- **Inputs:** domain value x, geometric type-2 fuzzy $\widetilde{A}$ the bounding vertices l and r;
- For every triangle in the lower surface of $\widetilde{A}$ test whether $l.x < x < r.x$. If so, append two vertices (x_1,y_1) and (x_2,y_2) to the list of vertices L where:
  - if $p.x < x$ and $p.x < min(q.x,r.x)$:
    - if $x < min(q.x,r.x)$:
      - $x_1 = p.y + \frac{x-p.x}{r.x-p.x}(r.y-p.y), y_1 = p.z + \frac{x-p.x}{r.x-p.x}(r.z-p.z)$;
      - $x_2 = p.y + \frac{x-p.x}{q.x-p.x}(q.y-p.y), y_2 = p.z + \frac{x-p.x}{q.x-p.x}(q.z-p.z)$;
    - else if $q.x < r.x$:
      - $x_1 = p.y + \frac{x-p.x}{r.x-p.x}(r.y-p.y), y_1 = p.z + \frac{x-p.x}{r.x-p.x}(r.z-p.z)$;
      - $x_2 = q.y + \frac{x-q.x}{r.x-q.x}(r.y-q.y), y_2 = q.z + \frac{x-q.x}{r.x-q.x}(r.z-q.z)$;
    - else if $r.x < q.x$:
      - $x_1 = p.y + \frac{x-p.x}{q.x-p.x}(q.y-p.y), y_1 = p.z + \frac{x-p.x}{q.x-p.x}(q.z-p.z)$;
      - $x_2 = q.y + \frac{x-q.x}{r.x-q.x}(r.y-q.y), y_2 = q.z + \frac{x-q.x}{r.x-q.x}(r.z-q.z)$;
  - if $q.x < x$ and $q.x < min(p.x,r.x)$:
    - if $x < min(p.x,r.x)$:
      - $x_1 = q.y + \frac{x-q.x}{r.x-q.x}(r.y-q.y), y_1 = q.z + \frac{x-q.x}{r.x-q.x}(r.z-q.z)$;
      - $x_2 = q.y + \frac{x-q.x}{p.x-q.x}(p.y-q.y), y_2 = q.z + \frac{x-q.x}{p.x-q.x}(p.z-q.z)$;
    - else if $p.x < r.x$:
      - $x_1 = q.y + \frac{x-q.x}{r.x-q.x}(r.y-q.y), y_1 = q.z + \frac{x-q.x}{r.x-q.x}(r.z-q.z)$;
      - $x_2 = p.y + \frac{x-p.x}{r.x-p.x}(r.y-p.y), y_2 = p.z + \frac{x-p.x}{r.x-p.x}(r.z-p.z)$;
    - else if $r.x < p.x$:
      - $x_1 = q.y + \frac{x-q.x}{p.x-q.x}(p.y-q.y), y_1 = q.z + \frac{x-q.x}{p.x-q.x}(p.z-q.z)$;
      - $x_2 = r.y + \frac{x-r.x}{p.x-r.x}(p.y-r.y), y_2 = p.z + \frac{x-p.x}{p.x-r.x}(p.z-r.z)$;
  - if $r.x < x$ and $r.x < min(p.x,q.x)$:
    - if $x < min(p.x,q.x)$:
      - $x_1 = r.y + \frac{x-r.x}{p.x-r.x}(p.y-r.y), y_1 = r.z + \frac{x-r.x}{p.x-r.x}(p.z-r.z)$;
      - $x_2 = r.y + \frac{x-r.x}{q.x-r.x}(q.y-r.y), y_2 = r.z + \frac{x-r.x}{q.x-r.x}(q.z-r.z)$;
    - else if $p.x < r.x$:
      - $x_1 = r.y + \frac{x-r.x}{q.x-r.x}(q.y-r.y), y_1 = r.z + \frac{x-r.x}{q.x-r.x}(q.z-r.z)$;
      - $x_2 = r.y + \frac{x-r.x}{p.x-r.x}(p.y-r.y), y_2 = r.z + \frac{x-r.x}{p.x-r.x}(p.z-r.z)$;
    - else if $r.x < p.x$:
      - $x_1 = r.y + \frac{x-r.x}{p.x-r.x}(p.y-r.y), y_1 = r.z + \frac{x-r.x}{p.x-r.x}(p.z-r.z)$;
      - $x_2 = q.y + \frac{x-q.x}{p.x-q.x}(p.y-q.y), y_2 = q.z + \frac{x-p.x}{p.x-q.x}(p.z-q.z)$;
- sort the output vertices by x and remove duplicates. This gives the left side of the output, and
- repeat this for the upper surface of $\widetilde{A}$. This gives the right side of the output.
- **Outputs:** list of vertices L;

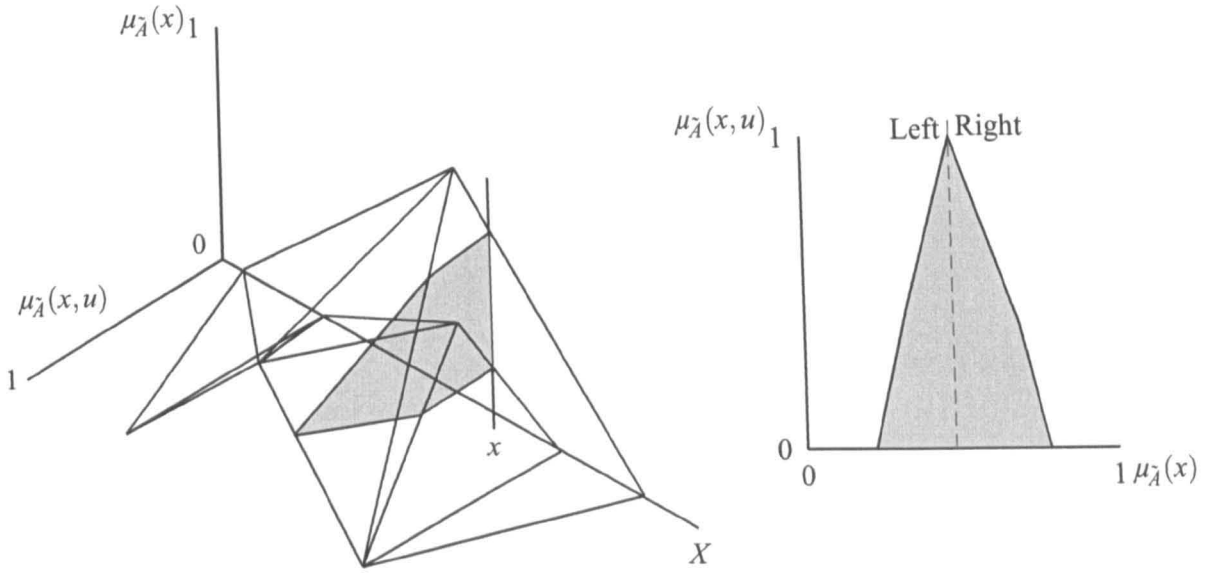


Fig. 4.12. The Membership Grade of x in the Geometric Type-2 Fuzzy Set $\widetilde{A}$.

4.3.2.2 Combination of Antecedents

The antecedent of a geometric type-2 fuzzy rule is a geometric secondary membership function. The logical connectives for such functions can be given by extending the type-1 fuzzy logical operations. As Section 3.3.2 explained the conjunction of two geometric type-1 fuzzy sets is given by the application of Weiler-Atherton clipping algorithm and the disjunction by the modified Weiler-Atherton clipping algorithm. The conjunction or meet and disjunction or join of two secondary memberships can also be found using the modified and unmodified Weiler-Atherton clipping algorithms. In both operations the left and right vertex lists are processed independently and can therefore be processed in parallel. The meet is given by applying the modified Weiler-atherton clipping algorithm to the left and right lists independently. The join is given by applying the Weiler-Atherton clipping algorithm to the left and right lists independently. Figure 4.13(a) depicts two geometric secondary memberships A and B. The join of A and B is depicted in Figure 4.13(b) and the meet in Figure 4.13(c). The relationship between the geometric join and meet and the discrete join and meet operations is now explored.

Firstly, consider the meet operation. Section 4.1 showed that the meet of the secondary memberships can be simplified when only using the *minimum* t-norm and *maximum* t-conorm. As was shown earlier, the geometric model only uses *minimum* and *maximum* since it is based on the optimised method. Recall from equation 4.9 that, for computational purposes, the domain of the resultant secondary membership function is split into three portions - before either apex, between apexes and after both apexes. However, geometric secondary memberships are split into two por-

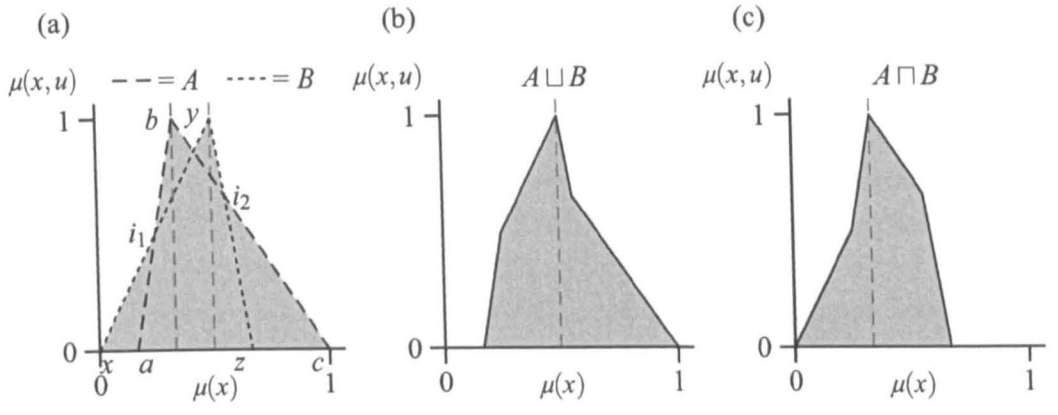


Fig. 4.13. (a) The Geometric Secondary Membership Functions A and B. (b) The *Join* of A and B. (c) The *Meet* of A and B.

tions - before and after the apex point. This difference in the models can be resolved, as explained below.

The application of the Weiler-Atherton and modified Weiler-Atherton clipping algorithms to the left and right side vertex lists yields different results:

- Applying Weiler-Atherton clipping to the left side $((x,y),(a,b))$ gives line segments that describe the minimum values across the left side of the geometric secondary membership function, giving (a,i_1,y).
- Applying Weiler-Atherton clipping to the right side $((b,c),(y,z))$ gives line segments that describe the maximum values across the right side of the geometric secondary membership function, giving (y,i_2,c).
- Applying modified Weiler-Atherton clipping to the left side $((x,y),(a,b))$ gives line segments that describe the maximum values across the left side of the geometric secondary membership function, giving (x,i_1,b).
- Applying modified Weiler-Atherton clipping to the right side $((b,c),(y,z))$ gives line segments that describe the minimum values across the right side of the geometric secondary membership function, giving (b,i_2,z).

In Figure 4.13 the apex points are b and y.

The meet operation requires that the maximum of the two secondary membership functions is taken up to the first apex point. This result is given by applying the modified Weiler-Atherton clipping algorithm to the left side vertex list. The meet requires that the minimum of the two secondary membership functions is taken after the second apex point. This result is given by

applying the modified Weiler-Atherton clipping algorithm to the right side vertex list. This also gives the maximum values between the two apex points as required by the meet operation.

The join algorithm can be explained in much the same way. The join operation given by equation 4.1 splits the domain into the same three portions, before either apex, between the apexes and after both apexes. The join operation requires that the minimum of the two secondary membership functions is taken up to the first apex point. This result is given by applying the Weiler-Atherton clipping algorithm to the left side vertex list. This also gives the minimum values between the two apex points as required by the join operation. The join requires that the maximum of the two secondary membership functions is taken after the second apex point. This result is given by applying the Weiler-Atherton clipping algorithm to the right side vertex list.

Once the antecedent values have been calculated their implication in the consequent must be found. This is explored in the next Section.

4.3.2.3 Implication

The geometric type-1 implication operation given in Section 3.2 works by clipping one collection of line segments against the other. The geometric type-2 clipping operation works in an analogous way by clipping one surface against another. The consequent set already consists of two surfaces, an upper and a lower. An example consequent $\widetilde{C}$ is depicted in Figure 4.15. The surfaces against which $\widetilde{C}$ will be clipped are derived from the rule antecedent, a geometric secondary membership function. The antecedent has two lists of vertices, left and right. The left list is extruded across the domain to form the lower surface and the right is extruded to give the upper surface. This extrusion is performed as follows. Let δ be a small distance in terms of the domain of $\widetilde{C}$. Let the extrusion start point be the x value of the first point in $\widetilde{C}-\delta$ denoted $min.x$. The end point of the extrusion is the x value of the last point in $\widetilde{C}+\delta$ denoted $max.x$. For each pair of adjacent vertices in the antecedent $(x_1,y_1),(x_2,y_2)$ form two triangles t_1 and t_2. Where $t_1 = ((min.x,x_1,y_1), (min.x,x_2,y_2), (max.x,x_1,y_1))$ and $t_2 = ((min.x,x_1,y_1), (max.x,x_1,y_1), (max.x,x_2,y_2))$. If the original antecedent pair are from the left list then t_1 and t_2 are added to the lower surface otherwise they are added to the upper surface. Figure 4.14 depicts an example extruded antecedent $\widetilde{A}$.

The next step in the implication process is to clip the respective surfaces against one another. Four clipping operations are performed independently, these are:

- the lower antecedent against the lower consequent;
- the lower consequent against the lower antecedent;
- the upper antecedent against the upper consequent, and

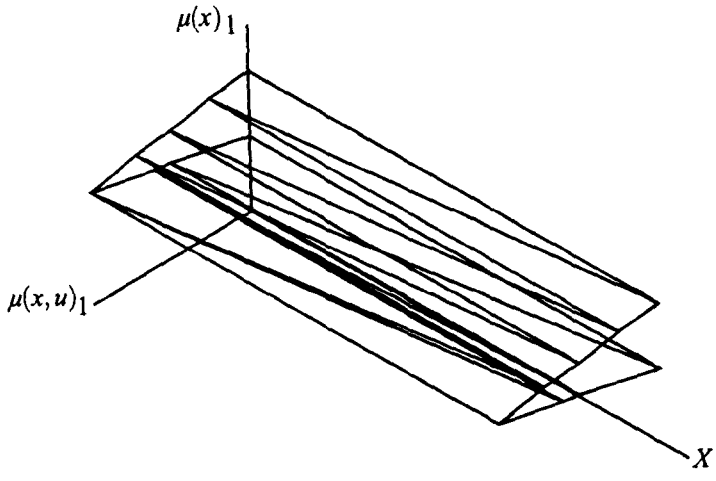


Fig. 4.14. The Extruded Antecedent $\tilde{A}$.

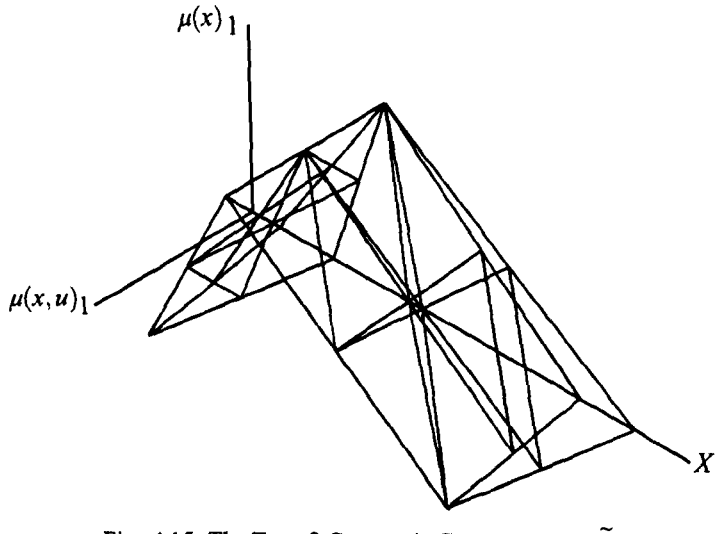


Fig. 4.15. The Type-2 Geometric Consequent Set $\tilde{C}$.

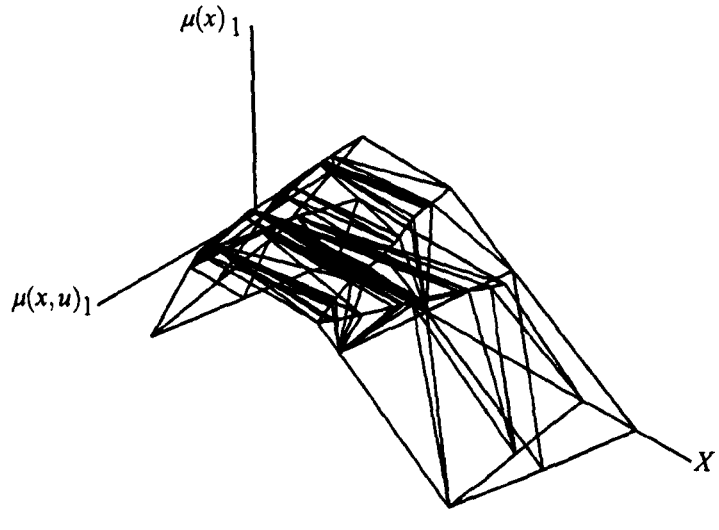
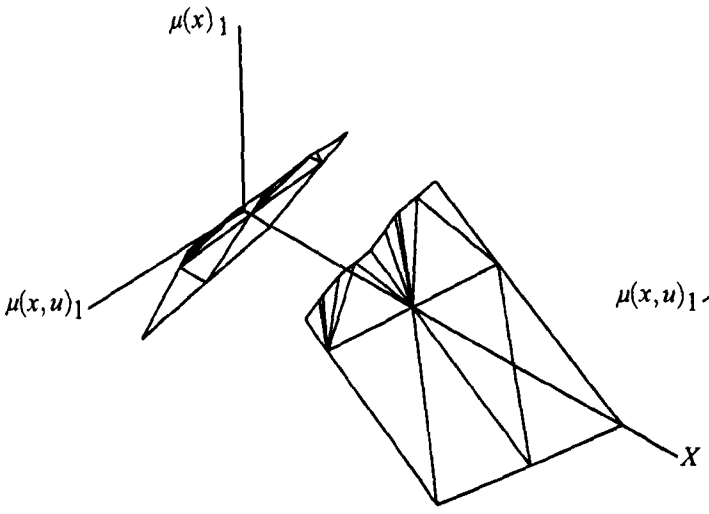
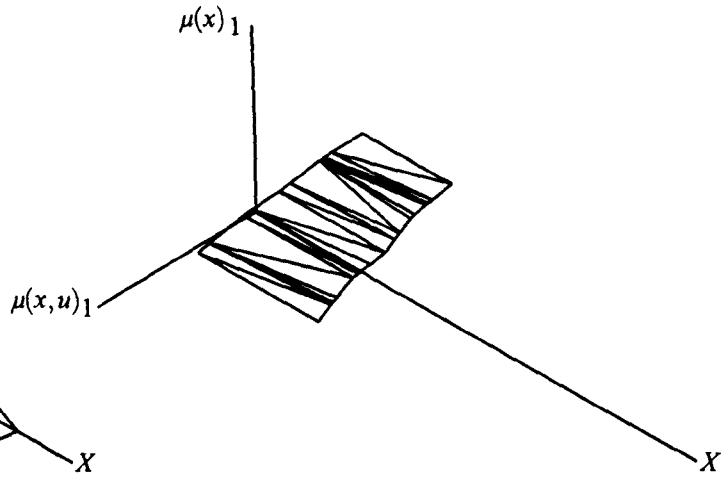


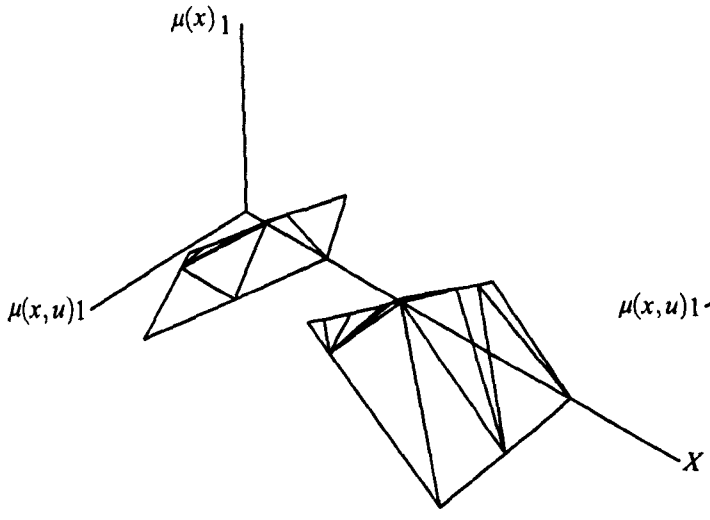
Fig. 4.16. The Final Output from a Geometric Type-2 Fuzzy System.



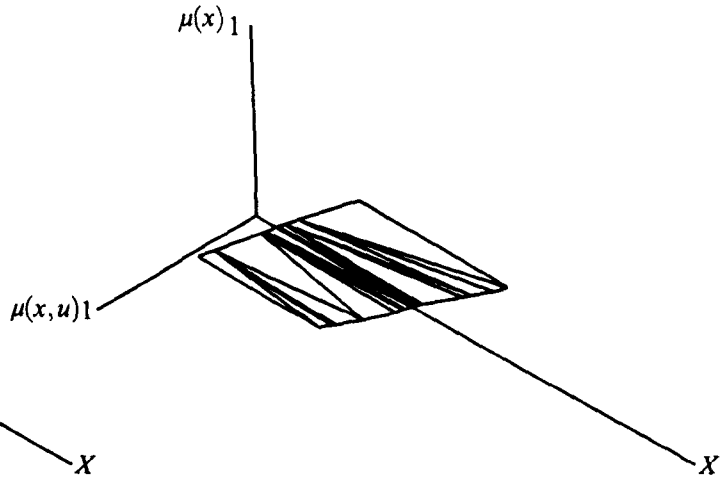
Upper Consequent Against Upper Antecedent



Upper Antecedent Against Upper Consequent



Lower Consequent Against Lower Antecedent



Lower Antecedent Against Lower Consequent

Fig. 4.17. The Four Clipped Surfaces.

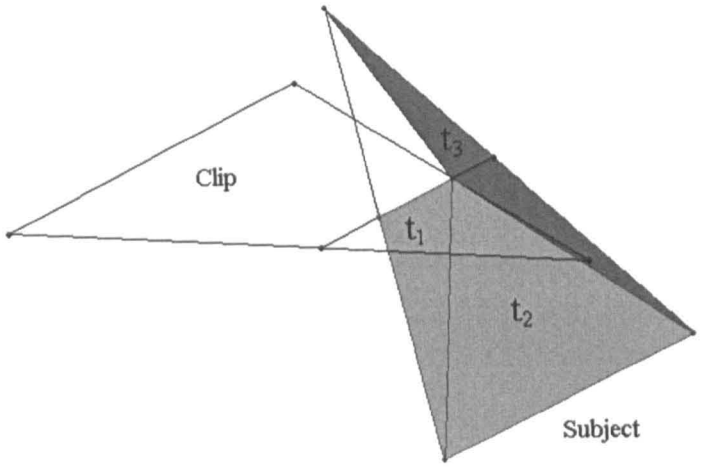


Fig. 4.18. A Clipped Triangle and the Resultant Triangles t_1, t_2 and t_3.

- the upper consequent against the upper antecedent.

To perform the clipping operations the novel surface clipping algorithm is used. The surface clipping algorithm is given in Algorithm 4.2. This algorithm works by finding the minimum of the two surfaces. This is done by testing whether a triangle is below other triangles which is done by measuring vertex to plane distances. When triangles are found to intersect the minimum possible surface that can be constructed from the intersecting triangles must be found. Once the four surface clipping operations have been performed then the results from all four operation must be merged to give a final resultant geometric type-2 fuzzy set. Figure 4.17 depicts the result of the four clipping operation of $\widetilde{A} \Rightarrow \widetilde{C}$. Figure 4.16 depicts the final result of $\widetilde{A} \Rightarrow \widetilde{C}$.

This geometric implication operation extends the geometric meet operation given in the previous Section. The membership grade at a single point in a geometric type-2 fuzzy set is a geometric secondary membership function. For the moment let us only consider the lower surface of the set and therefore only consider the left side of the geometric secondary membership functions. For a single secondary membership the meet for the left side would be given by the application of the modified Weiler-Atherton clipping algorithm. This algorithm essentially finds the minimum of the two left side functions. The surface clipping algorithm performs exactly the same task but with a 3-dimensional surface rather than a 2-dimensional line. Consider the two surfaces each made up of two triangles depicted in Figure 4.19(a). The dashed and dotted lines depict four possible secondary membership functions contained within in these two surfaces. Figure 4.19(b) depicts the meet of these two surfaces. The dashed lines depict the result of the meet of the four secondary membership functions. The surface clipping algorithm has in essence performed the meet on every possible secondary membership function contained within these surfaces. This

Algorithm 4.2 The Surface Clipping Algorithm

- **Inputs:** the surfaces *subject* and *clip*. *clip* is the surface that is being clipped against.
- Let there also be two lists of triangles - *currentsubject, currentclip*. These lists hold triangles where intersections are possible;
- order all triangles in both the surfaces by *min.x*;
- until off both the surfaces *subject* and *clip*:
    - take a triangle t from the head of *subject* or *clip*, whichever has the lower value of *min.x*;
    - add t to respective current list;
    - check whether any triangles can be removed from current lists i.e, check whether the *max.x* value for that triangle is less than the *min.x* of t;
    - if t is from *subject* :
        - if t is below all triangles in the clip list, then append t to *clipped*, and;
        - if t intersects with any of the triangles in the clip list then area below the intersection line must be output:
            - If only one vertex is below the intersection line then only one triangle is output. This triangle consist of that vertex and the end points of the triangle intersection line.
            - If two vertices lie below the intersection line the two triangles need to be added. One triangle consists the intersection line endpoints and one of the vertices. The other triangle consists of one of the intersection line vertices and the vertices that are below the intersection line.
            - If either of the intersection line endpoints is inside t then t must be replaced by triangles that cover the area of t that did not intersect.

      Figure 4.18 depicts two intersecting triangles and the resultant triangles t_1 and t_2 which are appended to *clipped* and t is replace by t_3 in the respective lists.
- **Outputs:** the surface *clipped*;

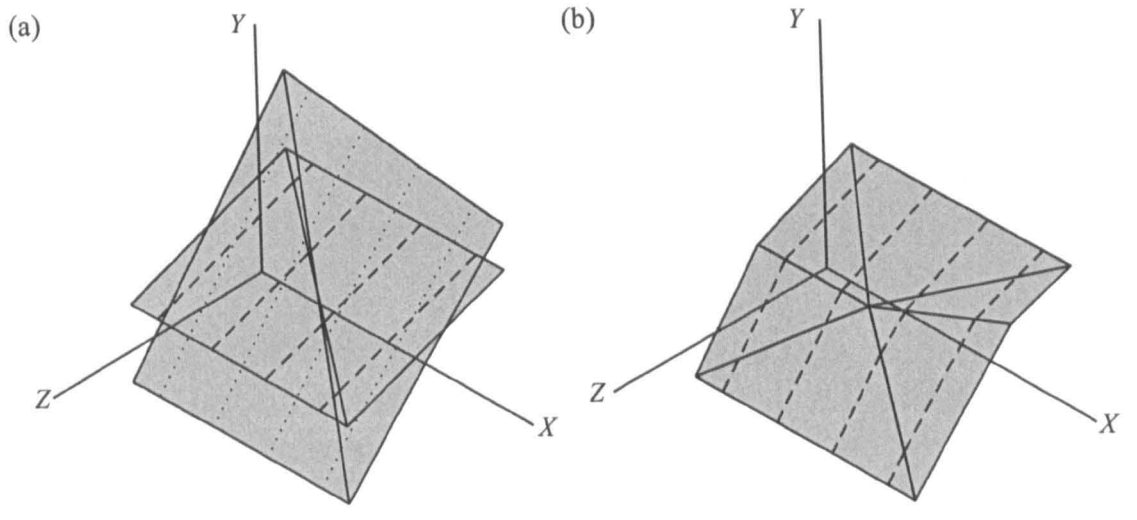


Fig. 4.19. (a) Two Surfaces. (b) The Minimum of those Two Surfaces.

methods works by considering each individual 2-dimensional secondary membership to be a single point on a 3-dimensional surface. The surface clipping algorithm extends the 2-dimensional line clipping algorithm into a 3-dimensional surface clipping algorithm allowing an entire surface to be calculated in one single process.

Once all the implied consequents from all the rules have been calculated they must be combined into a final geometric type-2 fuzzy set. This is explored in the next Section of this Chapter.

4.3.2.4 Combination of Consequents

Once all the rules have fired the rule consequents, which are geometric type-2 fuzzy sets, are then combined with the 'or' operator. The implication operation given in the previous Section essentially combined an extruded antecedent with a consequent using the 'and' operator. The 'or' operator works in a similar way using four independent surface clipping operations to arrive at a result. Consider the two geometric type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ depicted in Figure 4.21. To find $\widetilde{A \; or \; B}$ the following surface clipping operations are needed:

- The upper surface of $\widetilde{A}$ against the upper surface of $\widetilde{B}$;
- the upper surface of $\widetilde{B}$ against the upper surface of $\widetilde{A}$;
- the lower surface of $\widetilde{A}$ against the lower surface of $\widetilde{B}$, and
- the lower surface of $\widetilde{B}$ against the lower surface of $\widetilde{A}$.

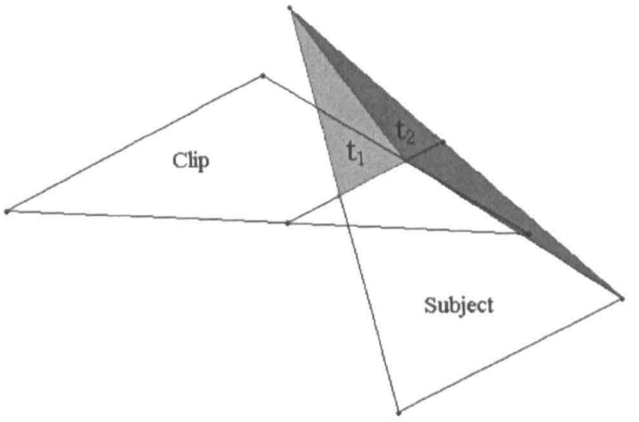


Fig. 4.20. A Clipped Triangle and the Resultant Triangles t_1 and t_2.

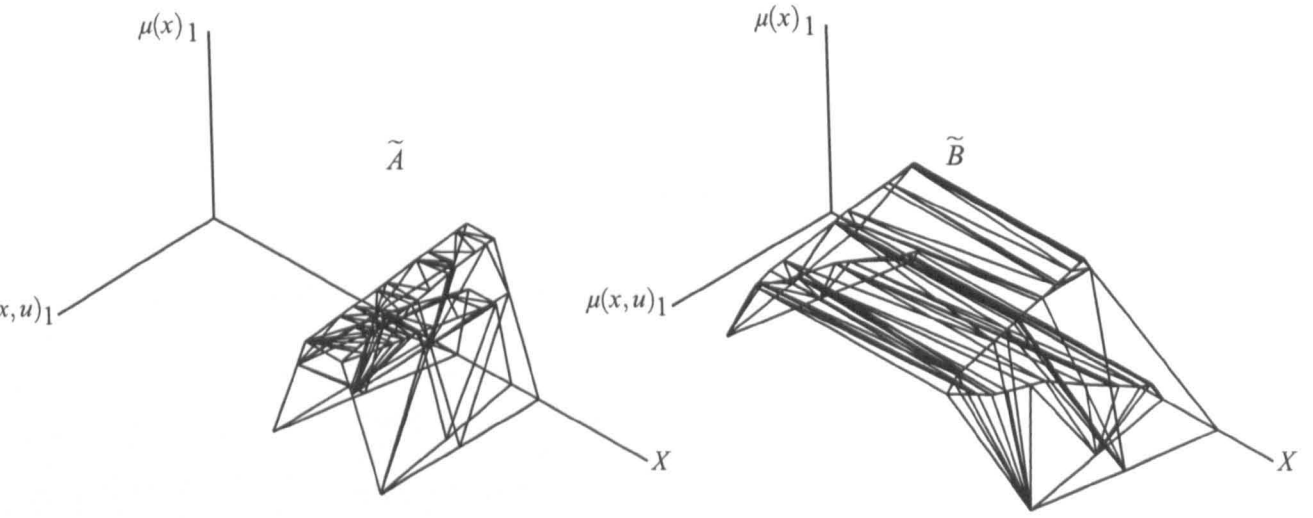


Fig. 4.21. The Geometric Type-2 Fuzzy Sets $\widetilde{A}$ and $\widetilde{B}$.

The implication operation used the surface clipping algorithm to give the *meet* across the surface of the geometric fuzzy sets. The disjunction operation requires the surface clipping algorithm to give the *join* across the surface of the geometric fuzzy sets. To achieve this, the surface clipping algorithm must be modified as it has to return the maximum of two given surfaces. The modified algorithm tests whether the triangles are above other triangles and when triangles intersect the highest possible surface is given. This modified surface clipping algorithm is given in 4.3.

The modified algorithm essentially finds the maximum of two 3-dimensional surfaces. This is analogous to the modified Weiler-Atherton clipping algorithm finding the maximum of two PLF. By applying this algorithm to the left and right surfaces of two geometric type-2 fuzzy sets the intersection, the join, is obtained for every point in the fuzzy set. The application of the modified

Algorithm 4.3 The Modified Surface Clipping Algorithm

- **Inputs:** the surfaces *subject* and *clip*. *clip* is the surface that is being clipped against.
- Let there also be two lists of triangles - *currentsubject*, *currentclip*. These lists hold triangles where intersections are possible;
- order all triangles in both the surfaces by *min.x*;
- until off both the surfaces *subject* and *clip*:
    - take a triangle t from the head of *subject* or *clip*, whichever has the lower value of *min.x*;
    - add t to respective current list;
    - check whether any triangles can be removed from current lists i.e, check whether the *max.x* value for that triangle is less than the *min.x* of t;
    - if t is from the subject polygon;
        - if t is above all triangles in the clip list, then append t to *clipped*, and;
        - if t intersects with any of the triangles in the clip list then area above the intersection line must be output.
            - If only one vertex is above the intersection line then only one triangle is output. This triangle consists of that vertex and the end points of the triangle intersection line.
            - If two vertices lie above the intersection line the two triangles need to be added. One triangle consists the intersection line endpoints and one of the vertices. The other triangle consists of one of the intersection line vertices and the vertices that are above the intersection line.
            - If either of the intersection line endpoints is inside t then t must be replaced by triangles that cover the area of t that did not intersect.

      Figure 4.20 depicts two intersecting triangles and the resultant triangle t_1 which is appended to *clipped* and t is replace by t_2 in the respective lists.
- **Outputs:** the surface *clipped*;

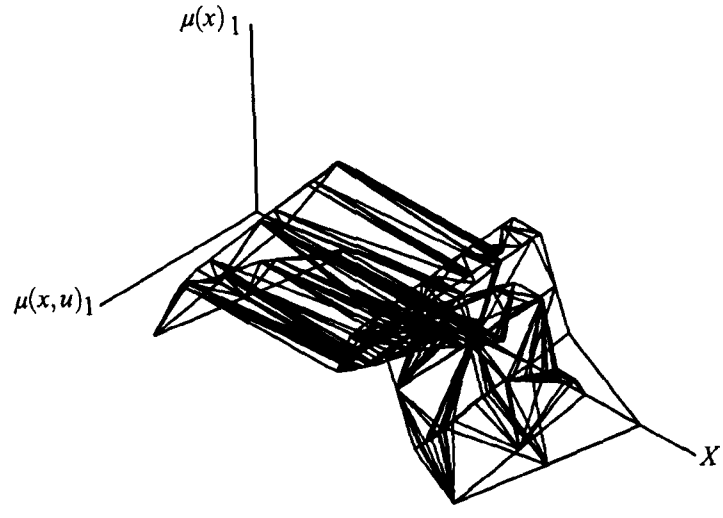


Fig. 4.22. The Type-2 Geometric Fuzzy Set $\widetilde{A \text{ or } B}$.

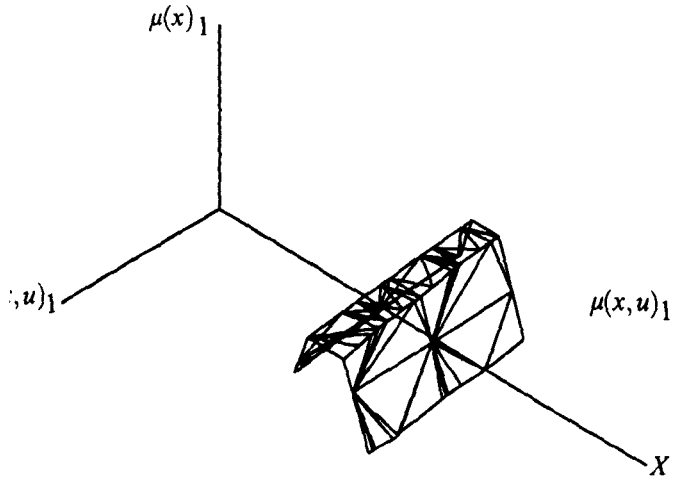
clipping algorithm to the four clipping operation stated above yields four new surfaces. These four resultant surfaces are depicted in Figure 4.23. To arrive at final resultant set $\widetilde{A} or \widetilde{B}$ these four surfaces are merged into two surfaces, one upper and one lower. This final inferred set is depicted in Figure 4.22. The final stage in the fuzzy system is to arrive at a representative crisp value by defuzzifying the final composite fuzzy set.

4.3.2.5 Defuzzification

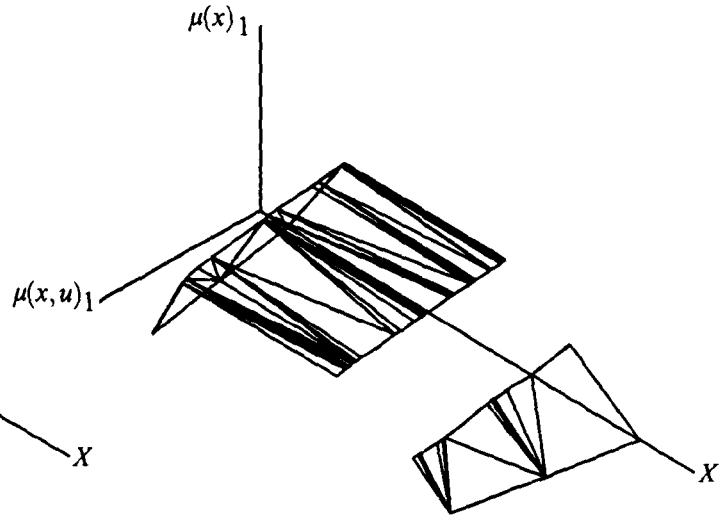
In order to arrive at a final crisp value discrete type-2 fuzzy sets must be type-reduced. This inferencing stage requires a large amount of computational resources. One of the main motivations behind the geometric approach is to eliminate a great deal of this computational load. This Section describes how a geometric defuzzifier can eliminate the need for type-reduction and in doing so reduce computational load.

Geometric type-2 interval systems are defuzzified by finding the centre of the polygonal area that represents that set. Geometric type-2 fuzzy sets are represented by 3-dimensional volumes rather than a 2-dimensional area. In order to defuzzify a geometric type-2 fuzzy set the x component of the centre of this volume must be calculated. To do this the complete 3-dimensional volume that represents a geometric type-2 fuzzy set must be modelled. The upper and lower surfaces contain all the 3-dimensional information needed to model a fuzzy set. However, these surfaces do not directly model the set's entire volume. The volume must be a closed space formed by geometric surfaces. There are three 2-dimensional areas that must be added to the upper and lower surfaces to form this volume:

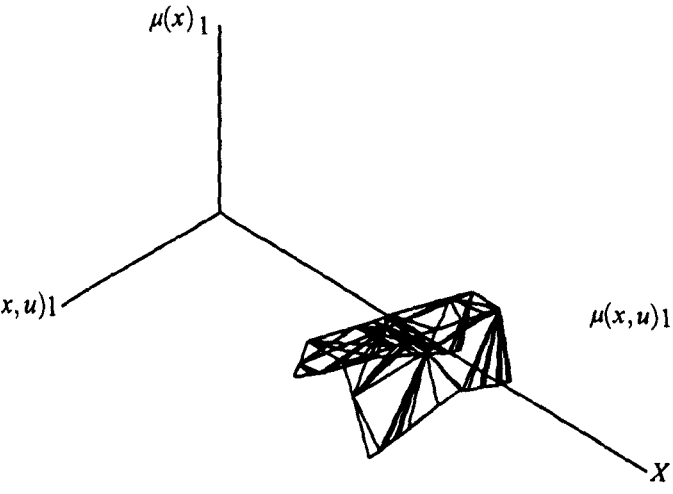
- The back face, the *FOU* the fuzzy set, polygon *a*;



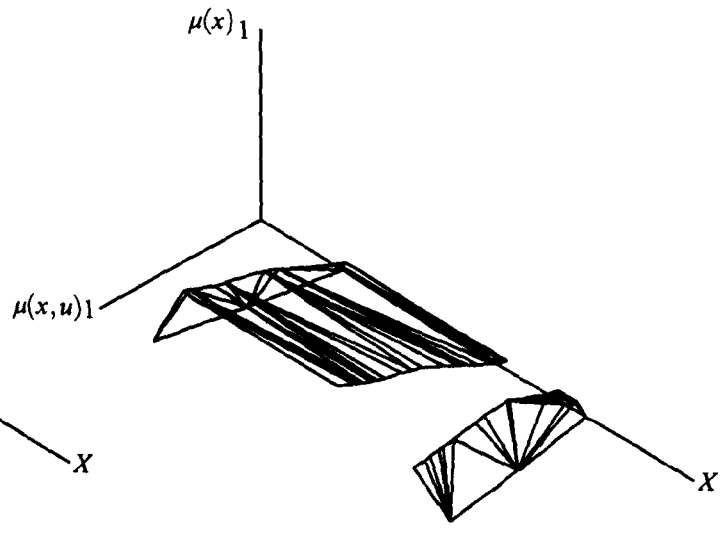
Upper $\widetilde{A}$ Against Upper $\widetilde{B}$



Upper $\widetilde{B}$ Against Upper $\widetilde{A}$



Lower $\widetilde{A}$ Against Lower $\widetilde{B}$



Lower $\widetilde{B}$ Against Lower $\widetilde{A}$

Fig. 4.23. The Four Surface Clipping Operations that give $\widetilde{A \text{ or } B}$.

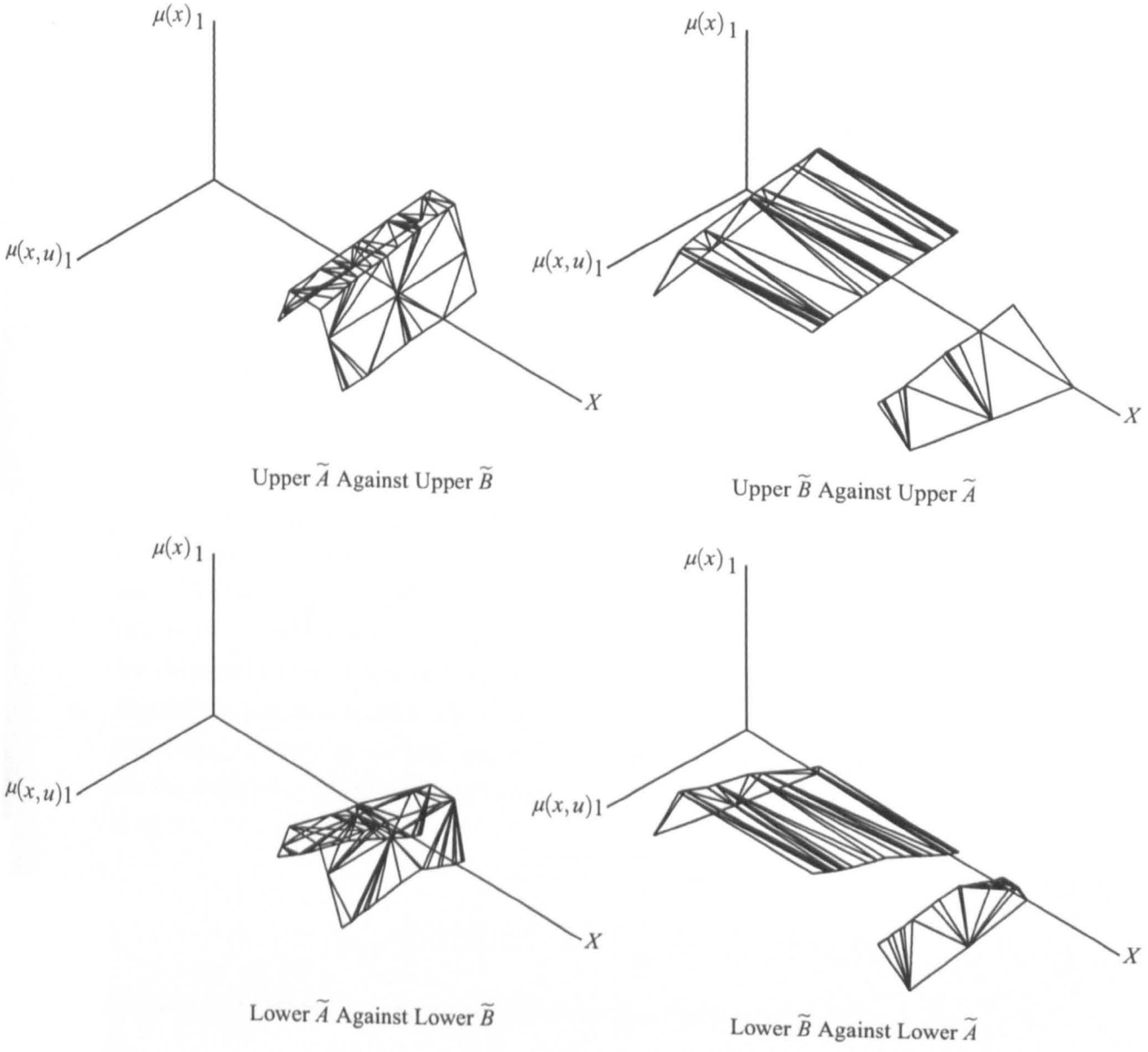


Fig. 4.23. The Four Surface Clipping Operations that give $\widetilde{A \text{ or } B}$.

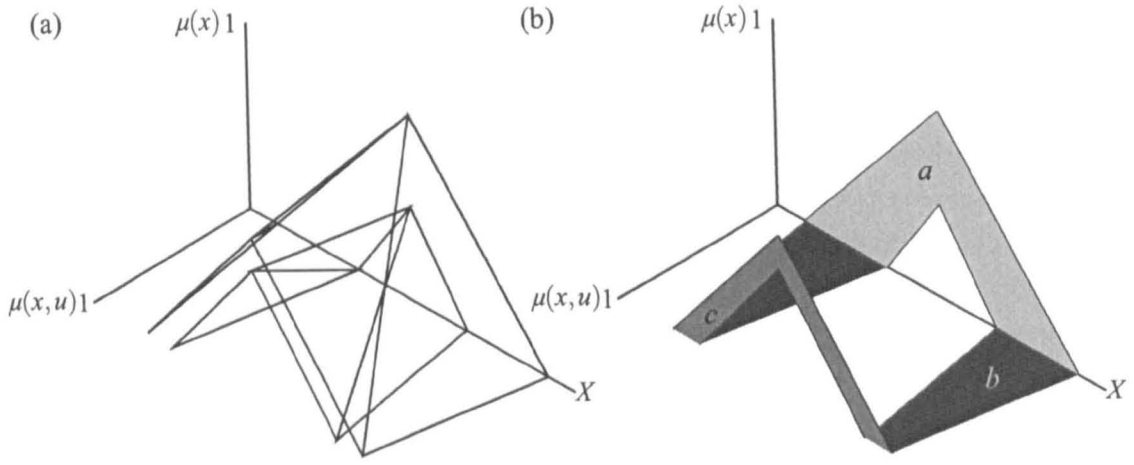


Fig. 4.24. (a) A Geometric Type-2 Fuzzy Set $\widetilde{A}$. (b) The Additional Surfaces Needed to Defuzzify $\widetilde{A}$.

- The footprint over the $x-z$ space, polygon b, and;
- Any gap between the upper and lower surfaces along the $x-y$ plane at z value of 1, for example when trapezoidal secondary memberships are used, polygon c.

Figure 4.24(a) depicts the surfaces of a geometric type-2 fuzzy set $\widetilde{F}$. Figure 4.24(b) depicts the 2-dimensional polygons a, b and c required to create a closed volume for defuzzification. The area and centroid of the a, b and c can all be calculated using two dimensional methods as given in Section 3.2. The overall centroid of a geometric fuzzy set $\widetilde{F}$ is given as a weighted average of the centroid of each triangle in $\widetilde{F}$ and the polygons a, b and c against the area of each triangle in $\widetilde{F}$ and the polygons a, b and c. The x component of the centroid of each triangle t in $\widetilde{F}$ is given by equation 4.30 where the vertices t are P, Q and R. The signed area of each triangle is given by half the determinant of the cross product of the two edge vectors of the triangle as given by equation 4.31.

$$C_t = \frac{P.x + Q.x + R.x}{3} \tag{4.30}$$

$$A_t = \frac{(Q.x - P.x)(R.y - P.y) - (Q.y - P.y)(R.x - P.y)}{2} \tag{4.31}$$

Equation 4.32 gives the centroid of a geometric type-2 fuzzy set $\widetilde{F}$.

$$C_{\widetilde{F}} = \frac{\sum_{i=0}^{N} C_{t_i} A_{t_i} + C_a A_a + C_b A_b + C_c A_c}{\sum_{i=0}^{N} A_{t_i} + A_a + A_b + A_c} \tag{4.32}$$

This method is identical to the one used by Bourke (1988) when calculating the centroid of a 3-dimensional object. By using the value given by equation 4.32 as the centroid, it is no longer

necessary to use type-reduction to defuzzify a type-2 fuzzy set, significantly reducing computational cost.

4.3.3 Discussion

This Section has presented a complete, new, geometric model of a type-2 fuzzy logic system. This model has built upon the geometric models of type-1 and type-2 interval systems presented earlier in this Chapter. The geometric model of a type-2 fuzzy set is the corner stone of geometric type-2 systems. Geometric type-2 fuzzy sets model the surface of a type-2 fuzzy set with two triangular meshes. This allows type-2 logical operations to be defined using the geometric knowledge of triangles presented at the start of this Chapter. A novel surface clipping algorithm has been presented. This algorithm extends the Weiler-Atherton clipping algorithm which clips line segments to allow the clipping of triangular mesh surfaces. As with the other geometric models presented in this Chapter some restrictions are placed on the operators that may be used, these are;

- Membership functions must be modelled by two triangular mesh surfaces;
- Secondary membership functions must be modelled by two piecewise linear functions;
- The only t-norm that can be used is the minimum, and
- The only t-conorm that can be used is the maximum.

Like the type-2 interval geometric model the geometric type-2 fuzzy logic systems do not conform to Mendel's design principle that when uncertainties disappear the system behaves like a type-1 system. If no uncertainties are present the volume of the surfaces modelling the fuzzy sets will be zero. The secondary membership grades in such set will be a vertical line from a single point. Then the geometric model will fail to correctly find the join or the meet of such secondary memberships. One of the underlying assumptions of the geometric model is that no two points in a secondary membership function have an equal x value. If this assumption is not met then it is possible that the Weiler-Atherton clipping algorithm will reverse the order of the two vertices with the equal x values corrupting the set model.

Defining a type-2 fuzzy set in terms of surfaces disconnects the definition from the discrete version which is based on discrete points. Unlike the geometric definitions of type-1 and type-2 interval fuzzy sets the membership grade of a geometric type-2 fuzzy set is not given by an equation but instead is the result of an algorithm. The 'and' and 'or' operations are also given by algorithms, although the defuzzifier is given by a simple equation. This algebraic separation from discrete

methods may make the geometric model less accessible to existing type-2 practitioners. The complexity of the 3-dimensional operations in the type-2 geometric model also represents a significant learning burden making the model less appealing for systems developers.

Like the type-2 interval geometric model the type-2 geometric model uses a direct defuzzification algorithm eliminating the need for type-reduction. The type-2 interval iterative method significantly reduces the computational burden of type-reduction. There is no equivalent method for type-2 type-reduction. This means that type-reduction for type-2 fuzzy system is a computationally significant burden. Thus, the geometric defuzzifier significantly reduces the computational burden of type-2 fuzzy logic by eliminating the need for type-reduction. This reduction in computation comes at a price. The geometric defuzzifier produces no measure of the uncertainty that has been propagated through the system. For control applications this will not be significant.

This Section has given a complete description of a geometric type-2 fuzzy logic systems. Such systems have the potential to give the improved performance of a type-2 fuzzy system but with a greatly reduced computational burden. The next Section discusses the potential for hybridisation of discrete and geometric type-2 system to give the lowest possible computational burden.

4.4 Hybridised Type-2 Fuzzy Logic Systems

Rather than using either the discrete or geometric model it may be computational expedient to produce a hybridised system which uses the most efficient components from each model. A hybrid type-2 fuzzy system uses the most efficient inferencing component from the geometric and discrete models, resulting in a computational minimal system. The system is computationally minimal in the sense that the amount of computation required is the lowest possible with the methods that have been defined to date. This Section discusses such a system.

The discrete and geometric model of a type-2 fuzzy set are completely inter-transformable. This means it is possible to take any geometric type-2 fuzzy set and transform it into a discrete type-2 fuzzy set. Similarly it is possible to take any discrete type-2 fuzzy set and transform it into a geometric type-2 fuzzy set. These transformation procedures make it possible to build a hybridised discrete/geometric fuzzy system from a set of components that can be 'plugged' together to form a complete and functioning system. The configuration of the components for a particular system will depend on the requirements for that system. If computational speed is the top priority then the geometric defuzzifier will be used. If the measure of the uncertainty propagated through the system is important then the discrete defuzzifier will be used. The two operations that transform discrete type-2 fuzzy sets to geometric ones and vice versa are now explored.

4.4.1 Discrete to Geometric Transform

A discrete type-2 fuzzy set consists of a number of discrete domain points each with an associated discrete secondary membership function. In order to transform such sets into geometric sets the apex point of each of the secondary memberships must be found. Let the function $M(x)$ map each point in the domain to the index number of the apex point in the secondary membership function at x and $N(x)$ map each point in the domain to the cardinality of the secondary membership function. The upper surface of the geometric set is given by equation 4.33.

$$\overline{\widetilde{A}} = \bigcup_{i=1}^{n-1} \bigcup_{j=N(x_i)}^{M(x_i)} t_1^{ij} t_2^{ij} \tag{4.33}$$

where

$$t_1^{ij} = ((x_i, u_j, \mu_{\widetilde{A}}(x_i, u_j)), (x_{i+1}, u_j, \mu_{\widetilde{A}}(x_{i+1}, u_j)), (x_i, u_{j-1}, \mu_{\widetilde{A}}(x_i, u_{j-1}))) \tag{4.34}$$

and

$$t_2^{ij} = ((x_i, u_{j-1}, \mu_{\widetilde{A}}(x_i, u_{j-1})), (x_{i+1}, u_j, \mu_{\widetilde{A}}(x_{i+1}, u_j)), (x_{i+1}, u_{j-1}, \mu_{\widetilde{A}}(x_{i+1}, u_{j-1}))) \tag{4.35}$$

where n is the number of points in the domain of the discrete type-2 fuzzy set $\widetilde{A}$. t_1 and t_2 are triangles which form the resultant surface. At every point in discrete set, the next point in both the primary and secondary domains are identified along with a another point which is adjacent to both of these two identified points. These four points are then used to define two triangles which describe the surface that covers these points. Consider Figure 4.25(a) which depicts a simple discrete type-2 fuzzy set. Figure 4.25(b) depicts two triangles that are used to connect four of the points in the upper surface as described in equation 4.33.

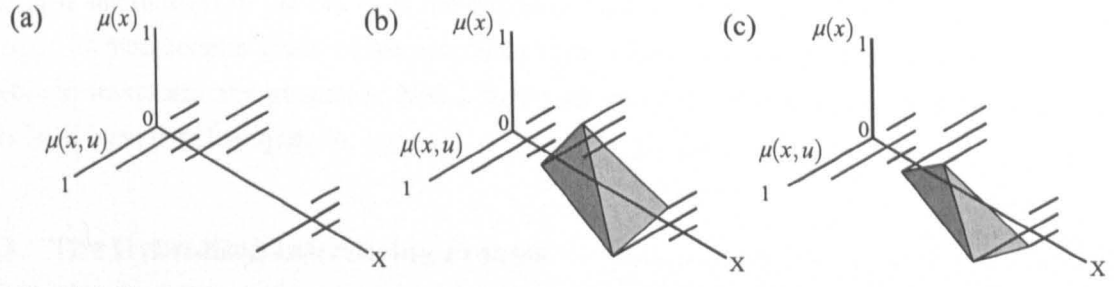


Fig. 4.25. (a) A Simple Discrete Type-2 Fuzzy Set. (b) The Geometric Transformation of Four Points from the Upper Surface. (c) The Geometric Transformation of Four Points from the Lower Surface.

The lower surface is constructed in exactly the same way. Let the function $T(x)$ map each point in the domain to the number of points in the secondary membership function at x. The lower surface

of the geometric set is given by equation 4.36.

$$\underset{\sim}{\widetilde{A}} = \bigcup_{i=1}^{n-1} \bigcup_{j=0}^{M(x_i)} t_1^{ij} t_2^{ij} \tag{4.36}$$

where

$$t_1^{ij} = ((x_i, u_j, \mu_{\widetilde{A}}(x_i, u_j)), (x_i, u_{j+1}, \mu_{\widetilde{A}}(x_i, u_{j+1}), (x_{i+1}, u_{j+1}, \mu_{\widetilde{A}}(x_{i+1}, u_{j+1})) \tag{4.37}$$

and

$$t_2^{ij} = ((x_i, u_j, \mu_{\widetilde{A}}(x_i, u_j)), (x_{i+1}, u_{j+1}, \mu_{\widetilde{A}}(x_{i+1}, u_{j+1})(x_{i+1}, u_j, \mu_{\widetilde{A}}(x_{i+1}, u_j)) \tag{4.38}$$

Figure 4.25(c) depicts two triangles that describe the surface of four points from the discrete fuzzy set depicted in Figure 4.25(a). These two equations define how any discrete type-2 fuzzy set can be transformed into a geometric type-2 fuzzy set.

4.4.2 Geometric to Discrete Transform

The transformation from a geometric type-2 fuzzy set to a discrete one is quite straightforward. As with any discrete type-2 fuzzy set the number of points m in the discrete primary domain must be decided upon. For each of these discrete domain points the number of points n in the domain of the secondary membership function must be decided upon. The resulting fuzzy can then be given by equation 4.39.

$$\widetilde{A} = \sum_{i=0}^{m} \sum_{j=0}^{n} \mu_{\widetilde{A}}(x_i, u_j) \; ; \quad x \in X, \; u \in J_x \tag{4.39}$$

where X is the domain of the set, J_x is the domain of the secondary membership function $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{A}}(x)$ is membership grade of the geometric type-2 fuzzy set $\widetilde{A}$ at x. This equation makes it possible to transform any geometric type-2 fuzzy set into a discrete one once the discretisation levels have been decided upon.

4.4.3 The Hybridised Inferencing Process

There are five processing stages any fuzzy logic system:

- Fuzzification;
- combination of antecedents;
- implication;
- combination of consequents, and

- defuzzification.

It is possible to transform between the geometric and discrete model at any of these processing stages. This means for any particular application the overall computational cost of a hybridised type-2 system may be minimised by transforming from one model to the other at one or more of the processing stages. For example, say for a specific application, the discrete model has the lowest computational cost up to the defuzzification stage. Type-reduction however, has a higher computational cost than geometric defuzzification, for this application. At this point a single final composite fuzzy set as been reached which represents the decision output from the system. This set can easily be transformed into a geometric type-2 fuzzy set and geometrically defuzzified. This represents a hybridisation of the discrete and geometric model to achieve the lowest possible computational cost of the system.

This Section has provided the techniques to enable transformations between discrete and geometric type-2 fuzzy sets. This allows hybridised type-2 FLS to be constructed.

4.5 Discussion

This Chapter has explored the geometric model of Mamdani FLS of type-2 interval and generalised type-2. These models, together with the type-1 geometric model presented in the previous Chapter, represent useful additions to developer choice, offering different levels of accuracy and execution speed to existing technologies. However, the significant outcomes of this work are the type-2 defuzzifiers, both type-2 interval and type-2. The increases in execution speed provided by these methods (which will be seen in the next Chapter) improve the applicability of type-2 fuzzy technology. The iterative method enables fast execution of type-2 interval systems. The disadvantages of this method have been explored here. The geometric defuzzifier has the advantage over the iterative method of have a pre-calculable level of computation, which is important for hardware implementation. The geometric defuzzifier also has a far lower computational cost than the standard type-reduction method. The geometric defuzzifier for type-2 fuzzy sets makes a significant contribution to type-2 fuzzy systems. The computational cost of type-reduction is massively higher than the cost of geometric defuzzification. The next Chapter demonstrates this advantage being exploited in a practical application. Each of the three geometric models is now discussed in turn.

The type-2 interval geometric model is based on the type-1 geometric model presented in the previous Chapter. As such the inferencing system provides the same high level of accuracy up to the point of defuzzification. This is where the geometric type-2 interval model departs significantly from the discrete type-2 interval model. Type-reduction is eliminated and replaced with

the geometric centroid of the set which is extended from the type-1 definition. Doing this raises issues the Mendel's design principle, which have been noted, however, the model still extends the type-1 centroid defuzzifier. The geometric defuzzifier provides low and consistent execution cost. The execution cost of type-reduction may be low, but this cannot be guaranteed. Type-reduction arrives at a result by using the two most extreme embedded sets, everything else is ignored. The geometric defuzzifier takes account of the entire set. To summarise, the discrete and geometric type-2 interval model perform in a very similar fashion up to the point of defuzzification where the geometric model exploits a geometric solution to reduce computational cost.

The type-2 geometric model departs entirely from the discrete model. The type-2 geometric model uses 3-dimensional geometry to provide logical operations over a continuous domain. This is a significant departure from the discrete approach which deals with each point in the domain separately. The computation involved in the 3-dimensional geometry is complex and may well be more computationally expensive than the optimized discrete methods. The discrete model however requires the computationally problematic type-reduction operation. It was shown that hybridised systems may be built that use the fastest parts of discrete and geometric type-2 inferencing, minimising computation. The type-2 geometric defuzzifier also break Mendel's design principle but is extended from type-1 and type-2 interval defuzzifiers. To summarise, the geometric type-2 model is significantly different from the discrete model, and instead is based on 3-dimensional geometry. This geometry is complex although the geometric defuzzifier is far more efficient than type-reduction. Hybridised systems take the most efficient components from each model to give a system with minimal computational cost. The reduction in computational complexity that results from use of novel methods presented in this Chapter is a major contribution to the field of type-2 fuzzy logic. A far wider set of applications of type-2 fuzzy logic can now be developed. The next Chapter investigates the deployment and performance of such a system. A geometric type-2 fuzzy logic controller for a mobile robot.

Chapter 5

Fuzzy Mobile Robot Controllers - An Investigation

The previous Chapter gave a complete theoretical description of geometric fuzzy logic. This Chapter describes the design and implementation of a number of fuzzy logic controllers (FLCs) for mobile robot navigation. A total of six controllers were built, all of which were adapted from an initial discrete type-1 FLC. The other controllers that are also presented in this Chapter are a geometric type-1, discrete and geometric type-2 interval and two hybrid type-2 controllers. The testing regime that these software based controller were subjected to is presented in Appendix C. This Chapter presents the design of, and the results from, an experiment which compares these six controllers. Each of the controllers will be numbered so that each controller can be referred to by name or number.

- Controller 1 is the discrete type-1 controller;
- Controller 2 is the geometric type-1 controller;
- Controller 3 is the discrete type-2 interval controller;
- Controller 4 is the geometric type-2 interval controller;
- Controller 5 is the hybrid type-2 controller with configuration H1, and
- Controller 6 is the hybrid type-2 controller with configuration H2.

The configuration of the hybridised type-2 controllers is explored in Section 5.3. The following Section discusses the task to be performed by all the controllers.

5.1 Task Selection

The motivation behind type-2 geometric fuzzy logic is to enable the application of type-2 fuzzy

logic to a wider range of problems. There are currently no reported systems (except Coupland *et al* (2006) which reports this experiment) where generalised type-2 fuzzy logic has been applied to a control application. This Chapter describes the first such application which has been made possible with the introduction of geometric type-2 fuzzy logic. As discussed in Chapter 2, type-2 fuzzy logic systems should be able to cope with the uncertainties inherent in control applications. To best evaluate geometric type-2 fuzzy logic in a control application a difficult mobile robot navigation problem was designed. Type-2 interval fuzzy logic has already been applied to such an application by Hagras (Hagras 2004). This study demonstrated improved performance in navigation tasks when using type-2 interval rules rather than type-1 under environmental uncertainties. One of the limitations of this study was that the robot only performed eight runs and therefore, it is difficult to state the significance, if any, of this performance improvement. However, the Hagras study demonstrated that mobile robot navigation is a useful application area for exploring the potential of type-2 fuzzy logic in control applications.

The task of mobile robot navigation represents a significant challenge for a type-2 FLC. The control system has to operate in real time on limited hardware resources. The definition of real time and the level of available resources is dependent on the specific robotic platform. The environment which the robot has to operate in is challenging. The sensors on the robot are operating in the real world and are prone to noise and error. For example the accuracy of a sonar sensor is likely to be reduced the further away an object is. Background noise in the environment may also effect the sonar reading. The level of traction between the wheels and the floor depends on the type of flooring, type pressures and the speed at which the wheels are moving.

The task to be completed by the FLC in this Chapter is to navigate a mobile robot around the curved edge of a wall like obstacle maintaining a distance of 0.5 metres between the centre of the robot and the obstacle at all times. A diagram of the robot, the obstacle and the ideal path that the robot should follow around the obstacle is given in Figure 5.1. The initial position of the robot puts the obstacle at a right angle to the left wheel of the robot. The initial distance between the obstacle and the centre of the robot is set at 0.5 metres. The robot is facing the correct direction to begin navigation of the obstacle. This start position places the robot just below the start point of the ideal path that should be taken by the robot around the obstacle. Once the centre of the robot crosses the dotted start line tracking begins. Once the centre of the robot crosses the dotted finish line tracking stops. The task of the FLC is essentially to minimise the deviation from the ideal path between the start and finish lines. The environment was static, that is the environment did not contain any moving objects. The reason for using a static environment is to aid the repeatability of the experiments. By its very nature a dynamic environment is nearly impossible to replicate.

The system was deployed on the commercially available pioneer 2 robot (depicted in Fig 5.2) built

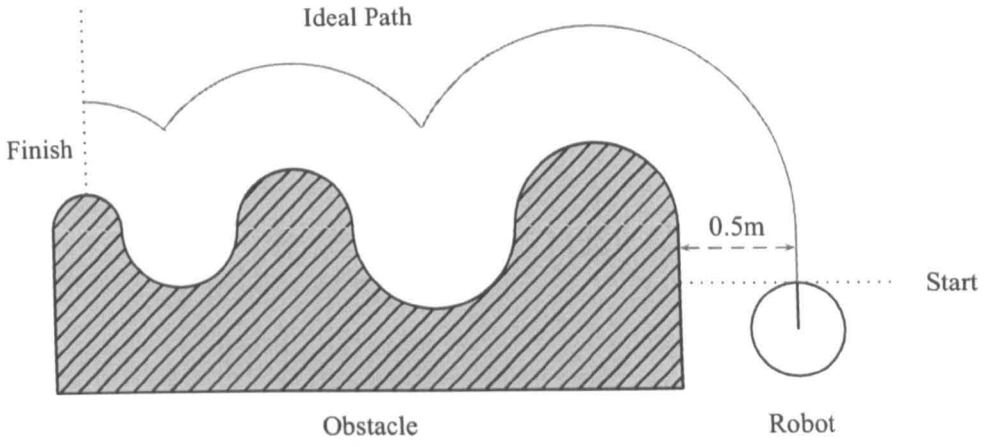


Fig. 5.1. Mobile Robot and Obstacle.

by ActivMedia*. The robot has an on board personal computer which the software based FLC was implemented on. This PC links to a microcontroller which is directly connected to the sensors and actuators. An array of eight sonar sensors each with a range of 3 metres provides sensory capability. Two independently driven wheels give mobility to the robot. The ARIA software library supplied with the robot provides a variety of methods for pre-processing the sonar input values and a variety of motor driving methods. This allows for the reduction in the number of inputs into the FLC which may reduce the rule base size. The FLC developed for this work had four inputs, d_1, θ_1, d_2 and θ_2:

- The angle θ_1 is the angle to the closest object detected by all eight sensors. θ_1 is given as a value between -110^o and 110^o;
- the angle θ_2 is the angle to the closest object detected by the middle four sensors. θ_2 takes a value between -40^o and 40^o;
- the distance d_1 is the distance to the nearest object detected by all eight sensors, and
- the distance d_2 is the distance to the nearest object detected by the middle sensors.

The only output from the system is the change in direction (δh) of the heading of robot. Since only the direction of the robot is being altered the speed of the robot is kept constant at 0.1ms^{-1}. The robot travels at this speed when moving in a straight line. However, when turning a component of this speed is taken up as rotational velocity. The robot is always moving forwards and can never go backwards.

Using the Pioneer 2 robot brings some additional challenges. The sonar configuration on this model of robot only provides sensory ability to the front of the robot. The rear of the robot has no

*ActivMedia Robotics, 19 Columbia Drive, Amherst, NH 0303, USA. Website: www.mobilerobots.com.

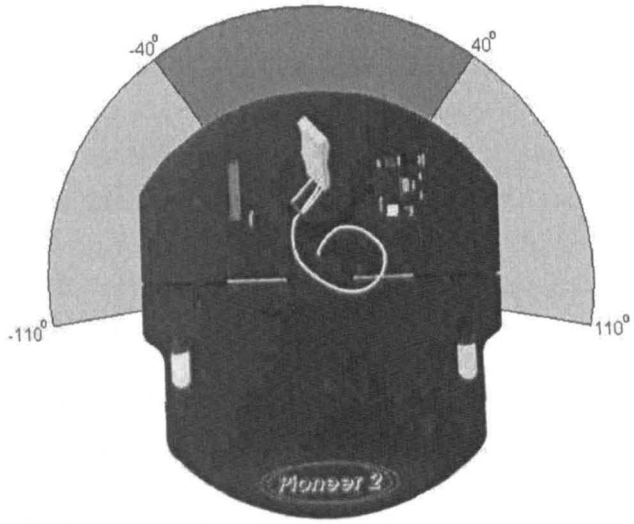


Fig. 5.2. The Pioneer 2 Mobile Robot.

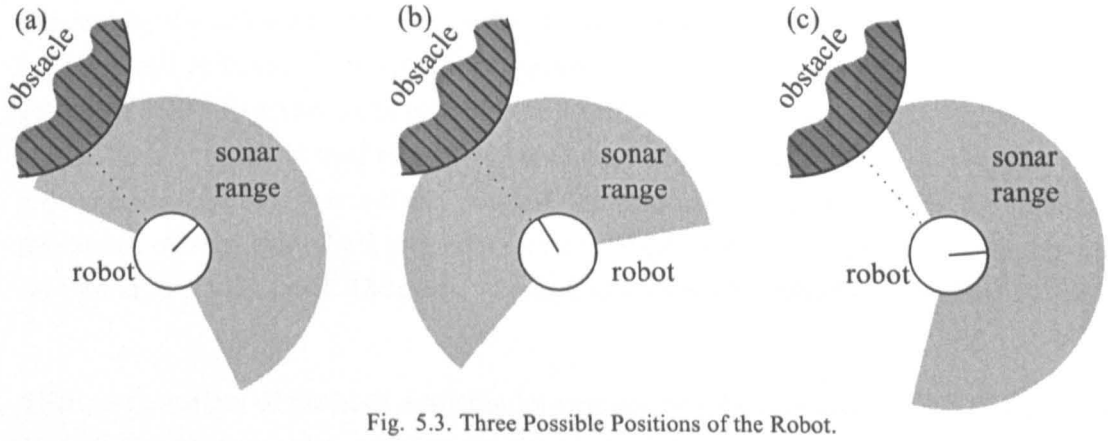


Fig. 5.3. Three Possible Positions of the Robot.

sensory capability. This causes additional difficulties when trying to maintain a constant distance to a wall. If the robot is facing away from the wall there is no way of the robot perceiving the walls existence. If the robot is actually facing the wall the sonar readings may still be incorrect. Consider the three robot positions relative to a wall depicted in Figures 5.3 (a), (b) and (c) where the dotted lines depict the shortest distance from wall to robot. In Figure 5.3 (a) the robot is following the path of the wall correctly. In Figure 5.3 (b) the robot has turned to face the wall. In both cases the sonar sensors are able to give a correct distance and angle from the robot to the wall. In Figure 5.3 (c) the robot is moving away from the wall. The sonar sensors are not able to sense the portion of the robot that is closest to the wall. The robot senses that the distance to the wall is longer than the actual distance to the wall and senses that the angle to the closest point on the wall is more acute than they actual angle. These erroneous readings would be fed into the FLC and erroneous actions taken.

The Aria software library provided with the robot requires that control commands are executed within a tenth to a quarter of a second window. This is the definition of real time for this robot, command execution within a quarter of a second or the robot operations will shutdown. This is quite a low requirement by control standards. It is however a significant challenge to perform type-2 fuzzy inferencing on limited hardware within a quarter of a second.

5.2 Controller Design

The previous Section described the task to be performed by a FLC deployed on a robot. This Section presents the design and rationale for the architecture of the FLC.

The process of designing a FLC involves making a number of choices and decisions as well as producing a rule base. These choices include which rule type to use, which form of operators to use, the level of set discretisation and the defuzzification method. Many of these choices are informed by the task being undertaken, the information available about this task and the available hardware and software. Throughout this thesis only Mamdani rules have been discussed. The geometric approach presented in the previous currently only models Mamdani style rules. For this reason, this rule type was used exclusively in all the FLCs presented in this Chapter. The geometric approach also requires that the only t-norms and t-conorms that can be used are minimum and maximum. All the controllers presented in this Section used minimum and maximum operators for t-norms and t-conorms. The centre of area defuzzification method was used for all controllers.

There are a variety of methods available for constructing the rule base for the FLC. Sugeno and Nishida (1985) suggest four methods for deriving rule bases:

1. The operator's experience;
2. The control engineer's knowledge;
3. Fuzzy modelling of the operator's control actions, and
4. Fuzzy modelling of the process.

Sugeno and Nishida advocate the use of the third option for a robot navigation task using TSK rules. Their example involved recording data from the sensors on the robot and recording the matching control actions begin taken by a human operator. A learning algorithm was then used to tune a TSK rules base (Mamdani rules could also be used) which modelled the actions of the human operator. This method could not have been used in conjunction with the geometric approach since learning algorithms are yet to be developed for such systems. The fourth option,

fuzzy modelling of the process, requires a mathematical model of the process which can be built upon. No such model is available for this task so this method was not used. Basing the fuzzy system on the knowledge of a control engineer was not considered an appropriate approach either. This would require either eliciting knowledge from a control engineer or having an existing crisp system (probably PID) on which to base the fuzzy system. Since neither were available this approach could not have been taken. The final option is to base the system on the experience of a robot operator. Experience of how to drive the robot around the obstacle was gained by the author. A joystick was connected to the robot over a wireless network. This joystick was then used to manoeuvre the robot around the obstacle. This process was repeated until the author was competent at driving the robot around the obstacle. The rules were based on this experience of driving the robot around the obstacle manually with a joystick.

The FLC also used an idea from control theory, the change in error over time, the derivative δe. This added an element of proportionality to the controller (Reznik 1997). To obtain the value of δe the gradient of a best fit line placed through the last four error measurements e, where $e = 500 - d_1$ was taken. Taking δe is useful as it gives a measure of whether the robot is moving toward the ideal path or away from it. This is particularly useful with this configuration of pioneer robots as they do not have any sonar sensors at the rear to detect whether the robot is moving toward or away from an object. The FLC only implements one behaviour with a single rule base rather than fusing two or more smaller, simpler behaviours. This decision was taken on the basis that the robot only has one task to perform, edge following. Other behaviours are implicit in that task. An example of an implicit behaviour is obstacle avoidance. When following the edge of the obstacle it is important not to drive into the obstacle and must therefore avoid it. By following an edge at a set distance, the robot is implicitly not crashing into that edge.

The overall structure of the FLC is depicted in Figure 5.4. The fuzzifier, inference engine and defuzzification components vary with the fuzzy technology being used, i.e. discrete, geometric or hybrid. All the controllers present here have identical rules. The fuzzy sets used in the rules varied depending on the controller:

- Discrete FLCs (controllers 1 and 3) used discrete fuzzy sets;
- geometric FLCs (controllers 2 and 4)used geometric fuzzy sets;
- type-1 FLCs (controllers 1 and 2) used type-1 fuzzy sets;
- type-2 interval (controllers 3 and 4) FLCs systems used type-2 interval fuzzy sets, and
- hybrid FLCs (controllers 5 and 6) used discrete and hybrid type-2 fuzzy sets;

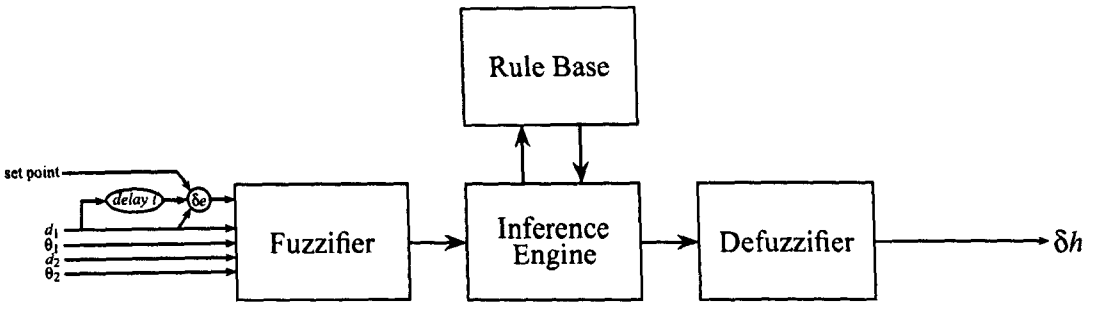


Fig. 5.4. The Structure of the FLC.

Rule 1:	IF θ_1 IS *left* AND d_1 IS *near* THEN δh IS *right*
Rule 2:	IF θ_1 IS *left* AND d_1 IS *correct* THEN δh IS *no change*
Rule 3:	IF θ_1 IS *left* AND d_1 IS *far* THEN δh IS *left*
Rule 4:	IF θ_1 IS *hard left* AND d_1 IS *far* THEN δh IS *hard left*
Rule 5:	IF d_1 IS *very far* THEN δh IS *left*
Rule 6:	IF θ_1 IS *left* AND d_1 IS *far* AND δe is *negative* THEN δh IS *no change*
Rule 7:	IF θ_1 IS *slight left* AND d_1 IS *correct* THEN δh IS *right*
Rule 8:	IF θ_1 IS *slight left* AND d_1 IS *near* THEN δh IS *hard right*
Rule 9:	IF θ_2 IS *centre* AND d_2 IS *far* THEN δh IS *right*
Rule 10:	IF θ_2 IS *centre* AND d_2 IS *correct* THEN δh IS *hard right*
Rule 11:	IF θ_2 IS *centre* AND d_2 IS *near* or *very near* THEN δh IS *hard hard right*
Rule 12:	IF θ_2 IS *right* AND d_2 IS *near* or *very near* THEN δh IS *hard hard right*

Table 5.1. Rule Base for the Edge Following Behaviour.

The rule base created for the edge following behaviour is given in table 5.1. Three distinct functions are contained within these rules:

1. When too far from the edge turn toward the edge.
2. When too close to the edge turn away from the edge.
3. When directly facing the wall turn to the right.

These tasks combine to give the single behaviour of edge following.

The simplicity of the rule base, the low number of rules, lends itself to type-2 FLC implementation where the computation levels are a critical factor. For systems where the size of the rule base is a problem Hagras (2004) suggests using a hierarchical behavioural architecture to optimize the rule base. The most trivial way to reduce the computational expense of any rule based system is to use fewer rules. A hierarchical architecture was not used as the rule bases were already small and the FLC only implemented a single behaviour.

The performance of the FLCs being designed here were all compared with one another. To make the rule bases comparable the membership functions of all the fuzzy sets were extended from an initial type-1 fuzzy rule base. The type-1 rules were based on human knowledge about the task

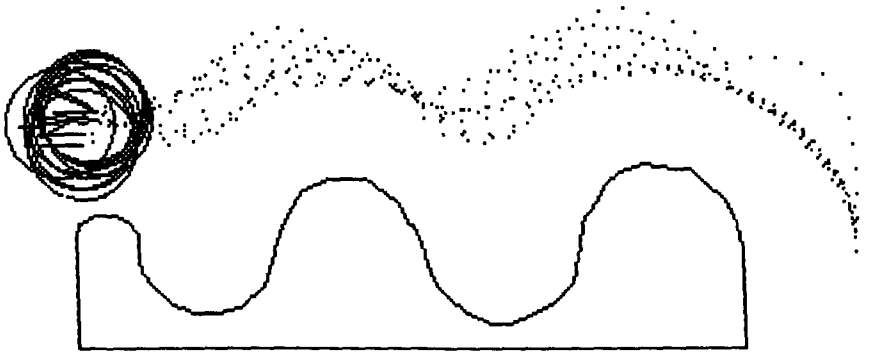


Fig. 5.5. Initial Ten Paths Recorded Using the Simulator.

gained from manually controlling of the robot. The rules were then tuned by hand with the use of a robot simulator. The control system was run on the simulator. The parameter of the rules were then adjusted, using an iterative, trial and error approach. This process continued until the controller gave satisfactory and robust performance. Figure 5.5 depicts the tracked path of the simulated robot using the discrete type-1 FLC over ten runs. This figure shows the FLC is capable is performing the task repeatedly, providing sufficient performance to allow experiments to be conducted using this FLC.

The type-2 interval and type-2 rule bases were defined by substituting the fuzzy sets in the type-1 rule base with type-2 interval and type-2 fuzzy sets. To arrive at the type-2 interval sets the membership functions of the type-1 fuzzy sets were widened to incorporate a level of uncertainty:

- The degree and position of widening was again tuned with the use of the simulator.
- The lower and upper bounds were set symmetrically around the type-1 functions.

The type-2 membership functions were based on a combination of the type-1 and type-2 interval fuzzy sets:

- The interval sets gave the FOUs of the type-2 sets.
- These gave the supports for the secondary membership functions.
- Each secondary was defined as a triangular membership function.
- The start and end points of the triangles were set as the lower and upper bounds of the type-2 interval fuzzy sets.
- The apex point of each secondary was set as the membership grade from the type-1 fuzzy sets.

For illustrative purposes an example type-1 fuzzy set *straight ahead* is given Figure 5.6, the interval version of *straight ahead* is given in Figure 5.7 where the dotted is the membership function of the equivalent type-1 fuzzy set. The type-2 version of *straight ahead* is depicted in Figure 5.8. Appendix B gives complete descriptions of all the fuzzy sets employed in all the FLCs presented in this Chapter.

The type-1 and the interval FLC both use a standard Mamdani architecture. In order to achieve the best possible performance from the type-2 systems the fastest FLC components must be selected giving a hybrid type-2 FLC. The process of selecting these components is presented in the next Section.

5.3 Offline Timing of Type-2 Fuzzy System Components

This section is only concerned with components used in hybrid fuzzy controllers numbered 5 and 6. When designing a hybrid type-2 fuzzy system the computational speed of each component should be tested. The fastest configuration of discrete and geometric components can then be used in the hybrid system. In this Section the execution speed of geometric and discrete type-2 fuzzy system components is compared. This comparison informs the design of two hybrid type-2 FLC whose execution speed is also given. The fastest type-2 systems components are combined in two hybrid type-2 fuzzy systems, controllers 5 and 6, which are experimentally analysed later in this Chapter.

The execution speed for a given problem of the five components of a discrete and a geometric type-2 fuzzy system are compared. These five components are:

- The fuzzifier;
- Antecedent combination;
- Rule implication;
- Consequent combination, and
- The defuzzifier.

The domain of the discrete fuzzy sets chosen for this experiment contained ten discrete points. Each secondary membership contains five discrete points. This gives a reasonably low but usable level of discretisation. The discrete implementation of a type-2 fuzzy logic system limits the possible level of discretisation. The maximum number of embedded sets it is possible to enumerate is 2,147,483,647. This is largest number it is possible to represent with a unsigned *int* data type in

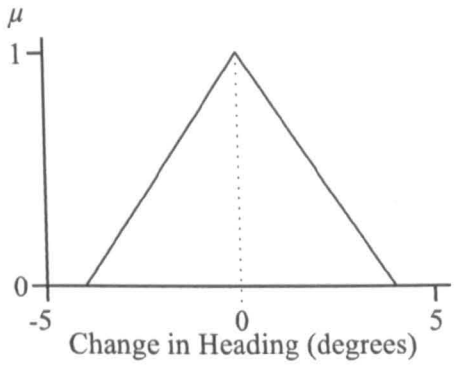


Fig. 5.6. The Type-1 Fuzzy Set *straight ahead*.

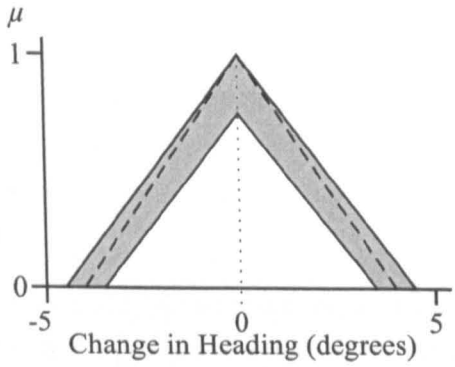


Fig. 5.7. The Type-2 Interval Fuzzy Set *straight ahead*.

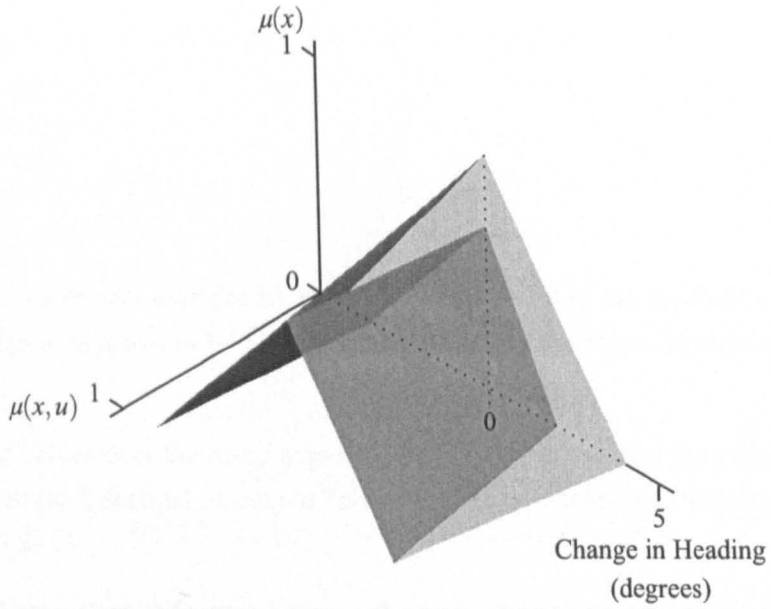


Fig. 5.8. The Generalised Type-2 Fuzzy Set *straight ahead*.

	Discrete	Geometric	H1	H2
Fuzzification	8.23×10^{2}	3.59×10^{3}	8.12×10^{2}	1.93×10^{3}
Combination of Antecedents	9.48×10^{1}	3.33×10^{3}	9.11×10^{1}	5.46×10^{2}
Implication	8.94×10^{2}	7.42×10^{4}	8.74×10^{2}	1.25×10^{4}
Combination of Consequents	3.22×10^{3}	8.88×10^{4}	3.52×10^{3}	4.11×10^{3}
Defuzzification	1.30×10^{8}	1.72×10^{3}	5.81×10^{2}	2.67×10^{2}
Total	1.30×10^{8}	1.72×10^{5}	5.88×10^{3}	1.94×10^{4}

Table 5.2. The Mean Timings (in microseconds) of Fuzzy System Components Over 30 Runs. Given to 3 s.f.

C++, the language of the Aria software library. This limit the size of data structure used to hold the enumerated embedded sets. Ideally, the discretisation levels would have been 25 and 25, the same as the fuzzy sets in the controllers. However this would require 1.78×10^{34} embedded sets to be enumerated. This was well beyond the data structure limit. A lower level needed to be chosen. Discretisation levels of 10 and 10 would require 5.00×10^{9} embedded sets to be enumerated, again above the imposed limit. The discretisation level of 10 and 5 were settled upon as they require 9,765,625 embedded sets to be enumerated. This is a realistically large number, but is however within the data structure limit.

The discrete system used the optimized join and meet operations presented in Section 4.1. Both systems used the rule base given in the previous Section. Each component of the system was timed over a single inference operation. This was repeated thirty times to demonstrate repeatability. The following inputs used when timing each system:

- $d_1 = 577$;
- $\theta_1 = 84$;
- $d_2 = 954$, and
- $\theta_2 = 8$.

A fixed value of -0.86 was used for δe, the derivative element of the controller. These values are typical of the inputs that would be given by the sensors on the robot, ensuring that some of the rules fire.

The mean timed values over the thirty experiments for the discrete and the geometric fuzzy components are given (to 2 decimal places) in Table 5.2. The coefficient of variance for these times is given in Table 5.3.

These results show a clear difference between the two approaches. Apart from the defuzzification stage the discrete fuzzy system is significantly faster than the geometric fuzzy system. The time

	Discrete	Geometric	H1	H2
Fuzzification	0.05	0.02	0.03	0.01
Combination of Antecedents	0.06	0.09	0.05	0.03
Implication	0.03	0.12	0.01	0.01
Combination of Consequents	0.10	0.02	0.02	0.02
Defuzzification	0.01	0.07	0.02	0.01

Table 5.3. The Coefficient of Variance for the Times Given in Table 5.2. Given to 2 d.p.

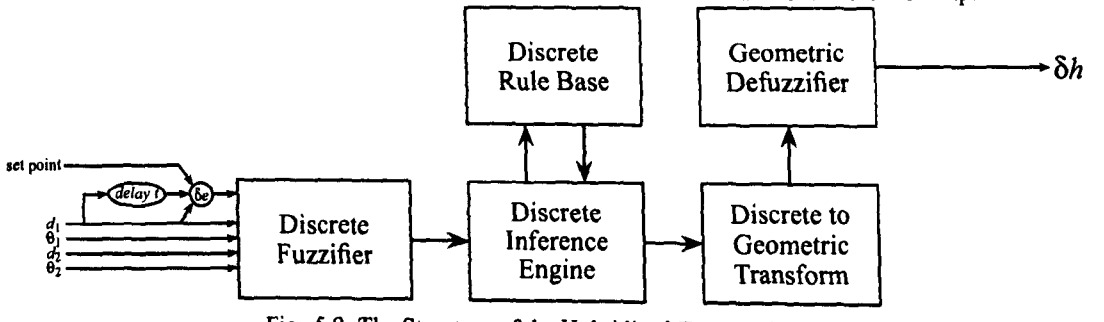


Fig. 5.9. The Structure of the Hybridised Type-2 FLC H1.

taken for defuzzification in the discrete system is 99.9% of the overall time taken by the discrete system. It is this defuzzification stage that makes the geometric system significantly faster overall despite all the other components in the geometric system being significantly slower than their discrete counterparts.

These findings suggest an obvious hybridised system (H1) which consists of all discrete components up to the defuzzification stage. At this point the single, final composite discrete fuzzy set is transformed into a geometric fuzzy set using the methods presented in Section 4.4. This geometric fuzzy set is then defuzzified using the geometric defuzzifier. The structure of the H1 system is depicted in Figure 5.9.

The 3-dimensional geometric operations being performed by the geometric system are complex and require a large number of floating point calculations to be performed. The second hybridised system (H2) attempts to improve on the performance the geometric system by eliminating the 3-dimensional geometric operations and replacing them with 2-dimensional ones. This is done by replacing all components, except the defuzzifier in the geometric system, with hybrid components. The sets are all hybrid sets which have discrete domains identical to the discrete set but have geometric secondary membership functions. The architecture of this system is identical to the discrete system. The difference between H2 and a discrete system is that the join and meet operations are geometric as defined in Section 4.3.2.2. This type of fuzzy system was presented by the author in Coupland and John(2005a) where it was called a partially discrete fuzzy system. In that publication this configuration of hybrid type-2 fuzzy system was shown to be significantly

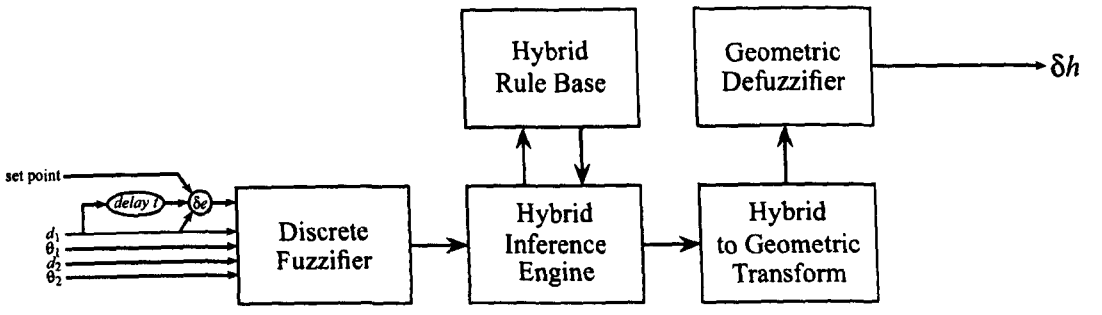


Fig. 5.10. The Structure of the Hybridised Type-2 FLC H2.

faster at performing the join and meet than the standard approach. The system was, however, not as fast as the operations given in Section 4.1. Before defuzzification can be performed the final composite hybrid type-2 fuzzy set must be transformed into a geometric fuzzy set. The discrete to geometric transform presented in Section 4.4.1 is used to do this. Although the secondary membership functions in this system are geometric, they are defined by a collection of discrete points, a piecewise linear function. As such applying the discrete to geometric transform to such set gives a geometric type-2 fuzzy set. The architecture of the type-2 FLC H2 is depicted in Figure 5.10.

This Section compared the computational speed of discrete and geometric components of a type-2 fuzzy system. These results of this comparison were used to inform the design of two hybrid FLC H1 and H2. The following Section presents the design of an experiment which compares the control performance of these two FLCs with type-1 and type-2 interval FLCs.

5.4 Experiment Design

The previous sections have presented the design of six FLCs. In this Section an experiment to compare the performance of these controllers is designed. Great care has been taken to ensure that the results can be effectively analysed with statistical methods.

5.4.1 Tracking the Robots Position

To get meaningful results about the ability of a robot to navigate around the curved obstacle, knowledge of the path the robot took around the obstacle is needed. In order to track the position of the robot a camera was mounted directly above the obstacle. The camera fed a VGA signal to a computer via a frame grabber card. A piece of software was developed that could identify the x,y co-ordinates of the brightest pixel in each frame. A red light emitting diode was placed on the top of the robot at its centre point. This experimental setup is depicted in Figure 5.11. The Experiments were conducted in a darkened laboratory. The pixel that corresponded to the location of the LED was consistently identified as the brightest pixel in the frame. This established that the

tracking system could consistently identify the position of the robot as a x,y co-ordinate within the camera frame. The software only begins recording the tracked position of the robot once it moves past the start line and finishes recording the path once the robot moves passed the finish line.

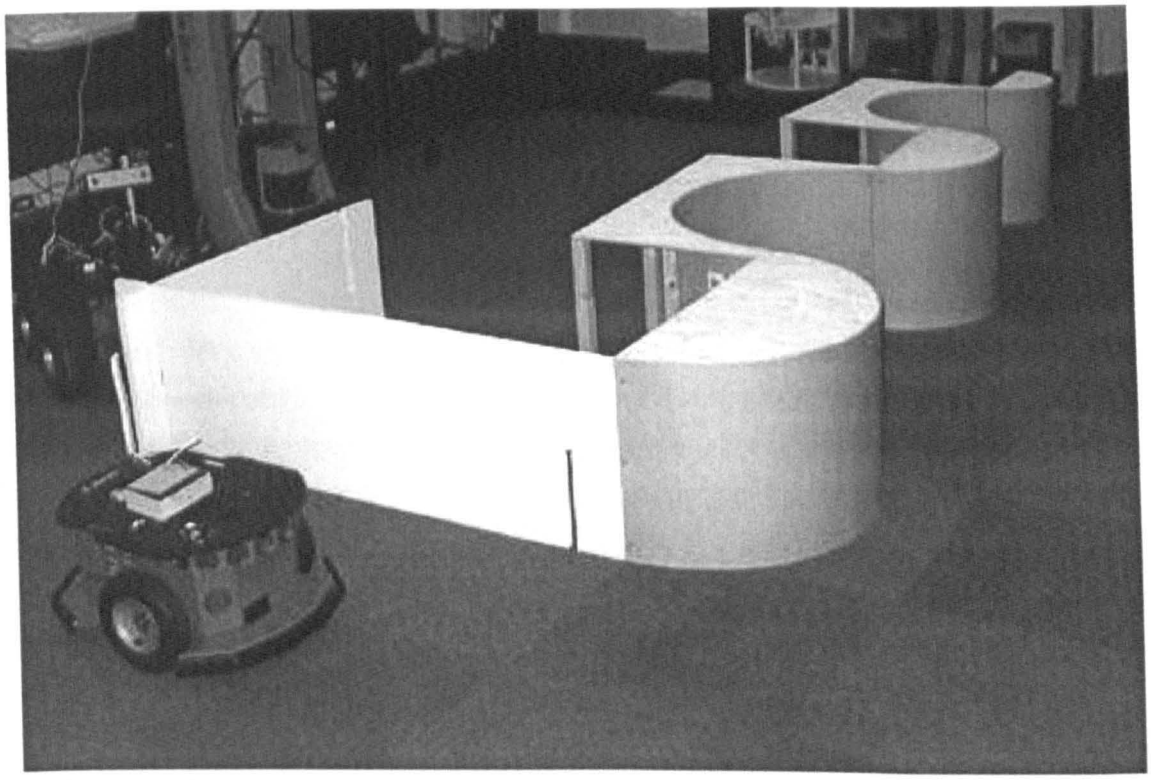


Fig. 5.11. Experimental Setup for Robot Path Capture.

In order to find the position of the robot relative to a path the position of the ideal path within the camera frame had to be found. To do this the ideal path that should be taken by the robot was scribed on the laboratory floor. The robot was moved manually around this path whilst tracking the position of the robot. This manually gathered data is used to represent the position of the ideal path in the camera frame. Gathering the ideal path and the experimental data in the same way meant that any errors due to any optical effects from the camera would be minimised. This gives a method for tracking the position of the robot relative to an ideal path. The drawback with taking this approach is that the results are only relative as accurate distance measurements in metres cannot be guaranteed. As such the raw data is only useful for this particular experiment, although comparable results are useful for later studies. The Hagras study did calculate the error level with absolute distances. This was achieved by attaching a pen to the robot which allowed the robot to scribe the path taken on the laboratory floor. The drawback with this technique is the potential for errors when drawing the path and measuring the drawn path. Also the amount of work involved in measuring a single path is not helpful when a large number of experimental runs to be performed.

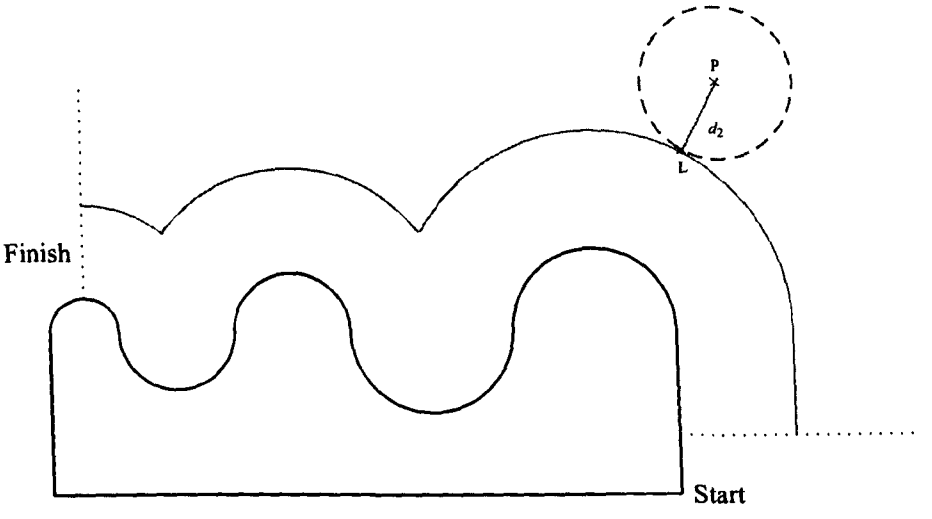


Fig. 5.12. Assessing the Error in the Position of the Robot.

The technique employed in this work only suffers from optical effects from the camera lens which were identical for all runs and the ideal path. The software tracking system makes path analysis far simpler for a large number of experimental runs.

The deviation from the ideal path, the error e of the position of the robot at point P, is the shortest distance between the ideal path and the point P in a straight line. To find the point on the ideal path closest to P an expanding circle technique was used. A circle with a radius of zero is placed at P. The radius of the circle is increased until a point on the circle intersects with a point on the ideal path. The radius of the circle at this point is the error e in the position of the robot relative to the ideal path. This technique is depicted in Figure 5.12.

5.4.2 Statistical Methods and Data Analysis

The purpose of conducting these experiments is to show any statistically significant difference between the performance of the six controllers and where any differences lie. Each experimental run gives a number of discrete data points which make up the path of the robot over the individual run. Each of these data points has an associated error, the amount of deviation from the ideal path. For each run the square root of the mean of the square of the error associated with each data point, the *RMSE* is taken. This gives a single measure of the FLC performance for that particular run. The variation in the *RMSE* gives a measure of the consistency of controller performance. So, the following two metrics were used to assess the properties of the control performances:

- *RMSE*. The ability to follow the ideal path.
- the standard deviation of *RMSE*. The consistency of the paths taken.

To investigate the significance of performance differences between the FLC, it was intended that an analysis of variance (ANOVA) would be performed on the results from a number of experimental

runs. The Bonferroni procedure was used to calculate the number of experimental runs that are required. Applying this technique allowed the type-1 error rate to be controlled in a multiple comparison ANOVA. To do this the following pieces of information must be known, the acceptable error rates and the expected standard deviation amongst the samples. Standard error rates of 5% for type-1 error and 10% for type-2 error are to be used for the anaylsis. A statistically significant difference between the controllers shows a performance increase which is repeatable approximately 95% of the time. An initial set of 20 experimental runs were conducted using the type-1 FLC. The standard deviation of the *RMSE* amongst these 20 runs was 0.98. Equation 5.1 is used to determine the sample size n required for the experimental results to be significant.

$$n = \frac{2(Z_{\alpha/2} + Z_{\beta})^2 \sigma^2}{E^2} \tag{5.1}$$

From (Ott & Longnecker. 2001) page 14

Where Z_{α} and Z_{β} are the type-1 and type-2 error levels respectively, σ is the standard deviation within the samples and E is error resolution. Using standard statistical tables $Z_{\alpha/2}$ was determined to be 2.67 and Z_{β} to be 1.28. The standard deviation of the *RMSE* from the initial 20 runs gave the value for *sigma* of 0.98. E has a value of 1 since that is the highest possible resolution, a single pixel. This gives a rounded value for n of 29. Each experiment was repeated 50 times. This number was chosen as its comfortably above the threshold of 29 suggested by the Bonferroni procedure. Fifty repetitions allows the experiments to be conducted over a short time period, minimizing the effect of variations in experimental conditions.

5.4.3 Reducing Experimental Error

All measurements taken during experiments are subject to errors. This Section describes the measures that were taken to minimise such errors and to minimise the effect of confounding variables when conducting the robot navigation experiments.

To minimize any performance variation due to battery drainage each controller was run in turn. The first seven experimental runs were run in the following order:

- Discrete type-1;
- Geometric type-1;
- Discrete type-2 interval;
- Geometric type-2 interval;
- Type-2 H1, and

- Type-2 H2.

This order continued until each controller had performed fifty runs. As mentioned earlier the laboratory was blacked out to optimise the performance of the path tracking system. At the end of each run the recorded path was visually checked for any obvious outlying data points. Runs containing outlying points would suggest a failure in the tracking system and would have been repeated. However, no outlying data points occurred, suggesting the tracking system was robust. Before each run took place the robot was placed carefully in the start position. The floor was marked to ensure that the robot began in the same position and was facing the same direction for every run. This meant the initial x,y coordinate position and heading of the robot were the same for every experimental run. The experiments were conducted over two consecutive days. The first 150 experiments were conducted on the first day and the remaining 150 on the second. The batteries that power the robot were recharged between these two sessions.

5.5 Results

5.5.1 Experimental Data

The path each robot FLC took around the obstacle was tracked fifty times. These tracked paths are depicted in Figures 5.13(a) to 5.13(f). The error for each point in this tracked path relative to an ideal path was calculated. The RMSE for each tracked run around the obstacle was then calculated. The mean, median, standard deviation and coefficient of variance over the fifty runs was then calculated for each robot FLC. These results are given in table 5.4. Each tracked path is made up of a number of discrete points. An approximate complete path may be plotted by connecting all the discrete points in one path together in the order in which they were sampled. These approximate complete paths are depicted in Figures 5.14(a) to 5.14(f). All of tracked paths are reproduced in a larger size in Appendix D.

Robot FLC	Mean Error	Median Error	St Dev of Error	Co Var of Error
Type-1 Discrete	13.5852	13.4185	1.0995	0.0809
Type-1 Geometric	14.8254	15.3726	2.4173	0.1631
Interval Discrete	12.5394	11.9779	2.0543	0.1638
Interval Geometric	10.1399	9.6363	1.8744	0.1848
Type-2 H1	9.8171	9.7783	1.0185	0.1038
Type-2 H2	14.2515	14.3913	2.7969	0.1963

Table 5.4. The Mean, Median, Standard Deviation and Coefficient of Variance of Error for the Six Robot FLC Over Fifty Runs. All numbers quoted to 4 d.p.

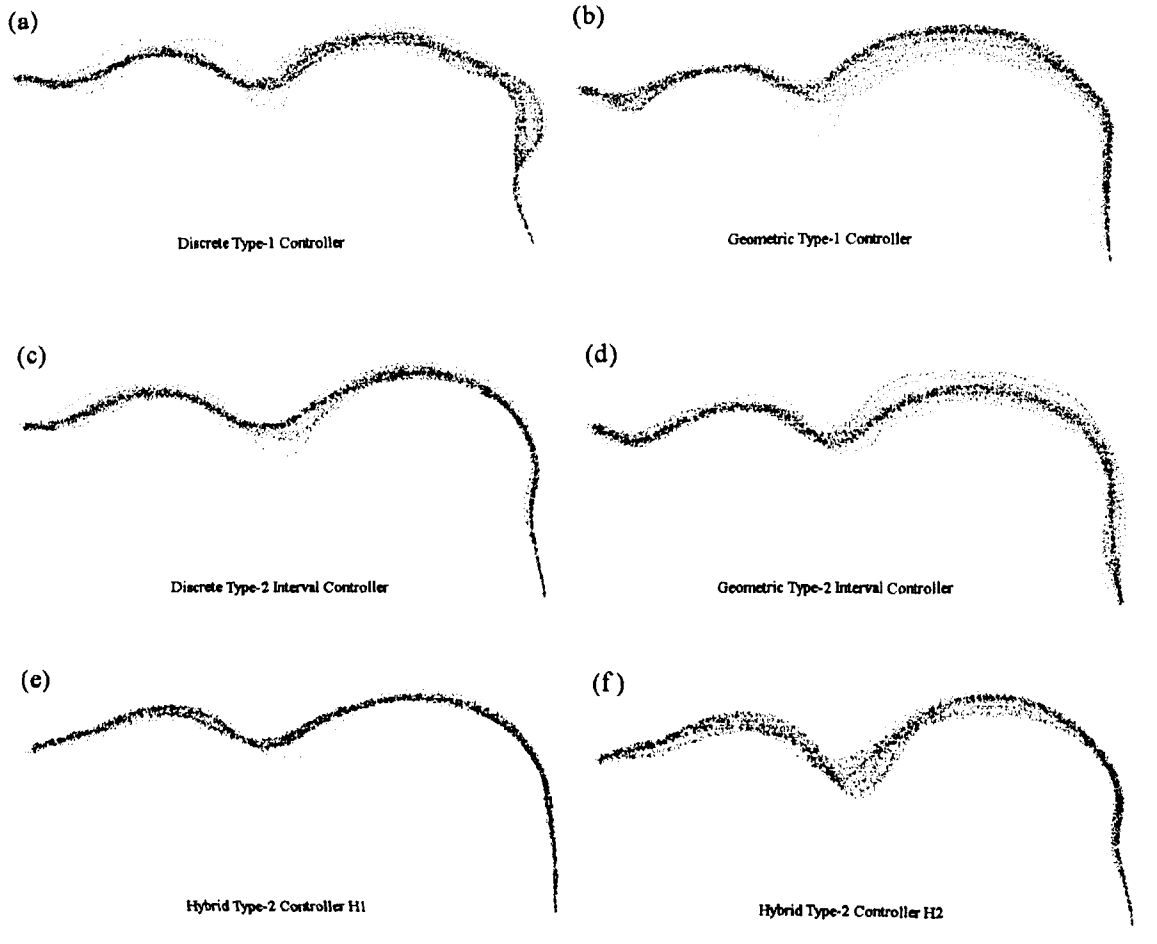


Fig. 5.13. The Raw Tracked Paths of the Robot FLCs.

5.5.2 Statistical Analysis

The first step in the analysis of the data is to test for the null hypothesis whether or not there are significant differences in the ability of the controllers to follow the ideal path. To do this a parametric statistic such as the F statistic can be used to confirm or reject the null hypothesis. Parametric statistics assume that the data being analysed displays certain characteristics. In the case of the F statistic it is assumed that all the populations have equal standard deviations and that each population has a normal distribution of values. Before performing such a test it is always wise to test these assumptions. To test normality a histogram of the results from each FLC was plotted, these are depicted in Figures 5.15 to 5.17 along with curves representing normal distributions. It is clear from these Figures that the results do not show a normal distribution. The distributions are not all skewed in a particular direction which would allow a data transformation to give normally distributed results. The other assumption to be tested is the equality of the variances amongst the six FLC. From the results given in table 5.4 it is clear that the variances are not equal. As a

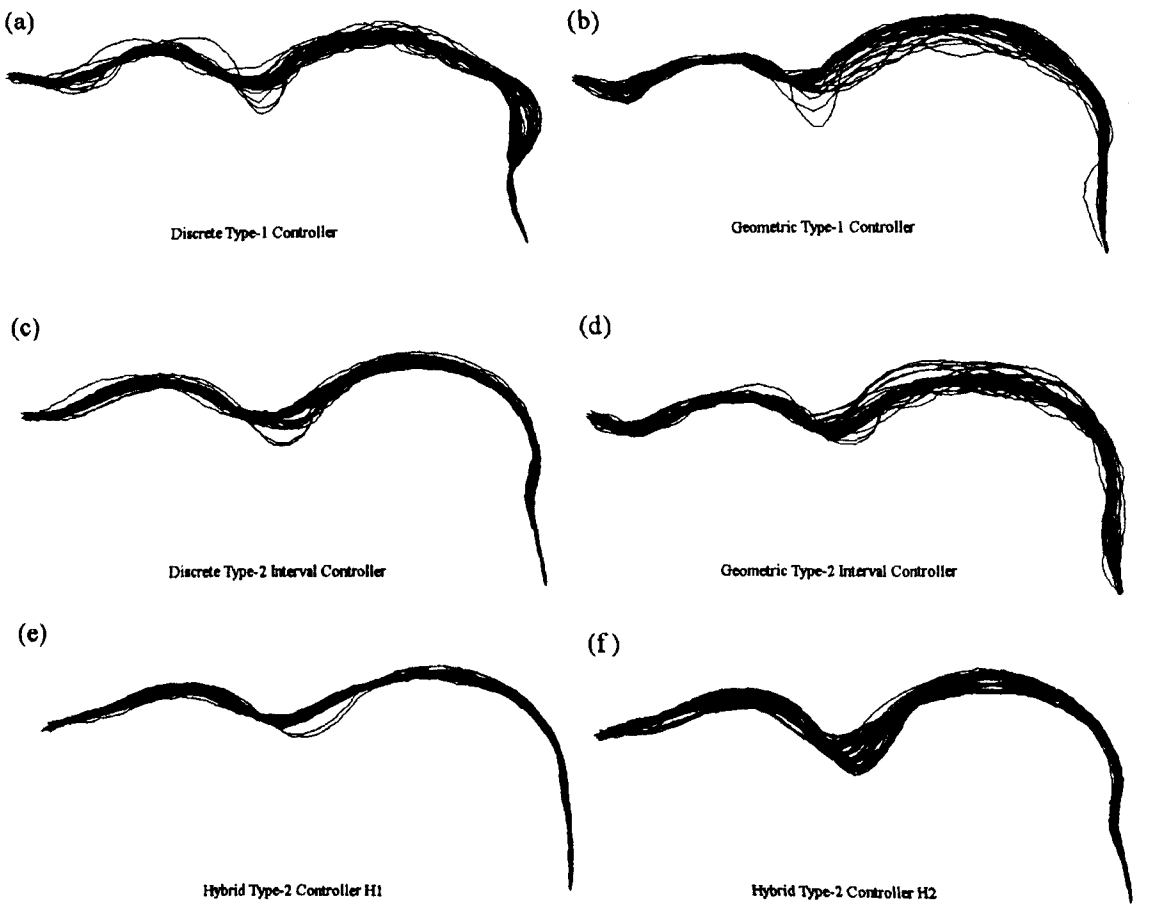


Fig. 5.14. The Tracked Paths of the Robot FLCs.

further measure the hypothesis that all variances are equal was tested. Using Levene's test with 95% confidence intervals a p-value of < 0.0005 was calculated showing compelling evidence to reject the hypothesis that all variances are equal. The values of the standard deviation for the six FLC's along with the confidence intervals are depicted in Figure 5.18.

Since the assumptions of normality and equality of variance have both failed it would not have been sensible to proceed with an analysis of variance. Instead a non-parametric statistic, the Kruskal-Wallis test, was used to test the null hypothesis. There is some disagreement about the assumptions that are required for the Krukal-Wallis test. Some (Ott & Longnecker. 2001) feel that although normal distributions are not necessary, the distributions should all have the same form. Others (Ostle & Malone 1988) are satisfied that the Kruskal-Wallis test holds for any data set where the results are independent. The Kruskal-Wallis procedure assess whether there is evidence to confirm or reject the null hypothesis h_0 in favour of the alternative hypothesis h_a:

- h_0: $\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6$

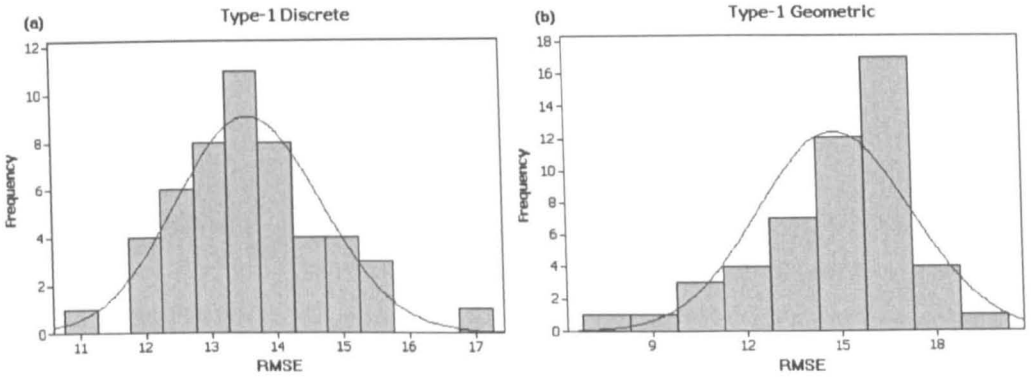


Fig. 5.15. Histogram of (a) The Results from the Discrete Type-1 FLC and (b) The Results from the Geometric Type-1 FLC both with Normal Curves.

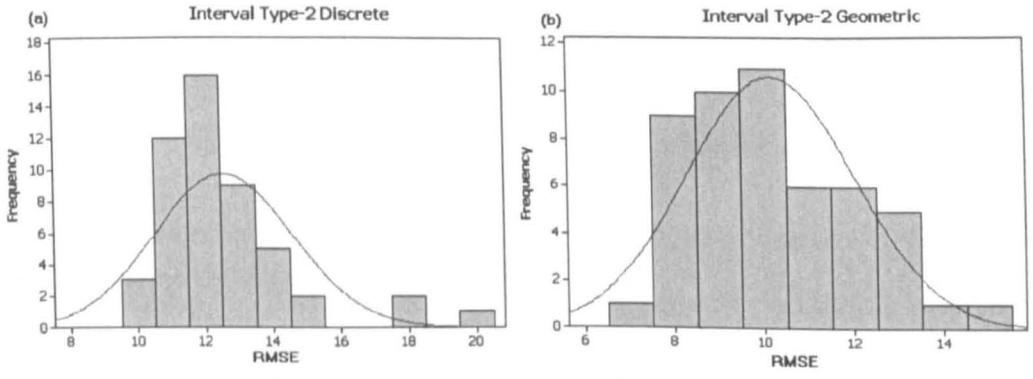


Fig. 5.16. Histogram of (a) The Results from the Discrete Type-2 Interval FLC and (b) The Results from the Geometric Type-2 Interval FLC both with Normal Curves.

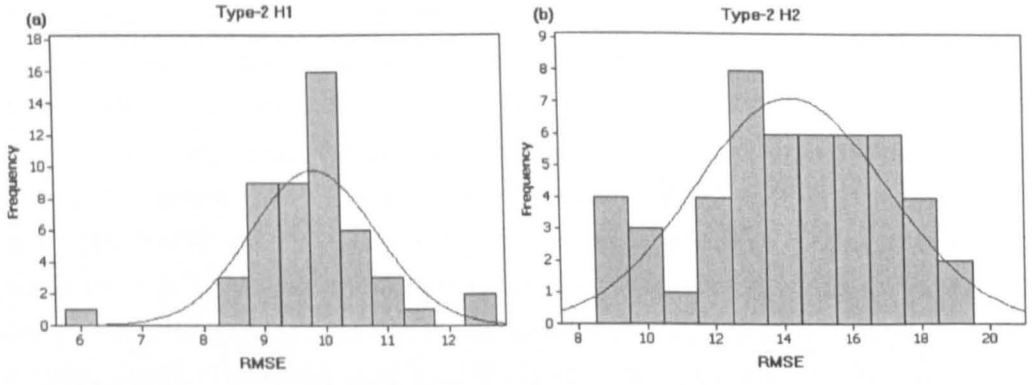


Fig. 5.17. Histogram of (a) The Results from the Type-2 FLC H1 and (b) The Results from the Type-2 FLC H2 both with Normal Curves.

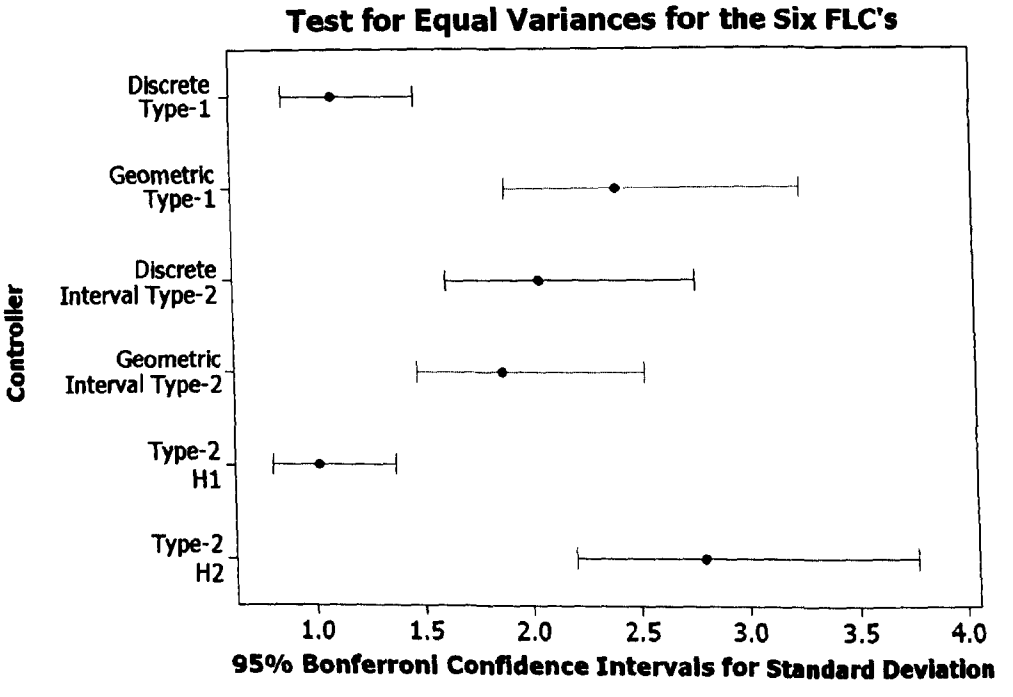


Fig. 5.18. Variances of the Six FLC with 95% Confidence Intervals.

There are no differences between the medians of the *RMSE* from the six FLC, i.e. all six FLC performed equally.

- h_a: $\neg(\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6)$
  There is a difference between the medians of the *RMSE* from the six FLC, i.e. at least one of the FLC gave a differing performance.

Where, in this experiment $\mu_{1...6}$ are the mean error values of the six FLCs.

The Kruskal-Wallis test ranks each experimental run by its RMSE value. The test statistic H is calculated based on the sum of these ranks. The value of H for the result from the six FLC was 185.95 to two decimal places, compared to a critical value of 11.07. This gave a p value which is < 0.0005. The null hypothesis can therefore be rejected and it may be stated that there is some difference in the performance of the six controllers. The Kruskal-Wallis test ranked each run of the robot by the RMSE value of that run. All the ranking positions of each controller are plotted in Figure 5.19. The mean rank of each controller is given in Table 5.5. Each line across Figure 5.19 depicts the rankings of the runs from a different controller. The highest rank, 1, is placed on the left of the Figure. The lowest rank, 300, is placed on the right of the Figure. The distribution of the ranking places of each controller can be seen.

The Kruskal-Wallis test has led to the null hypothesis being rejected. The test showed that there

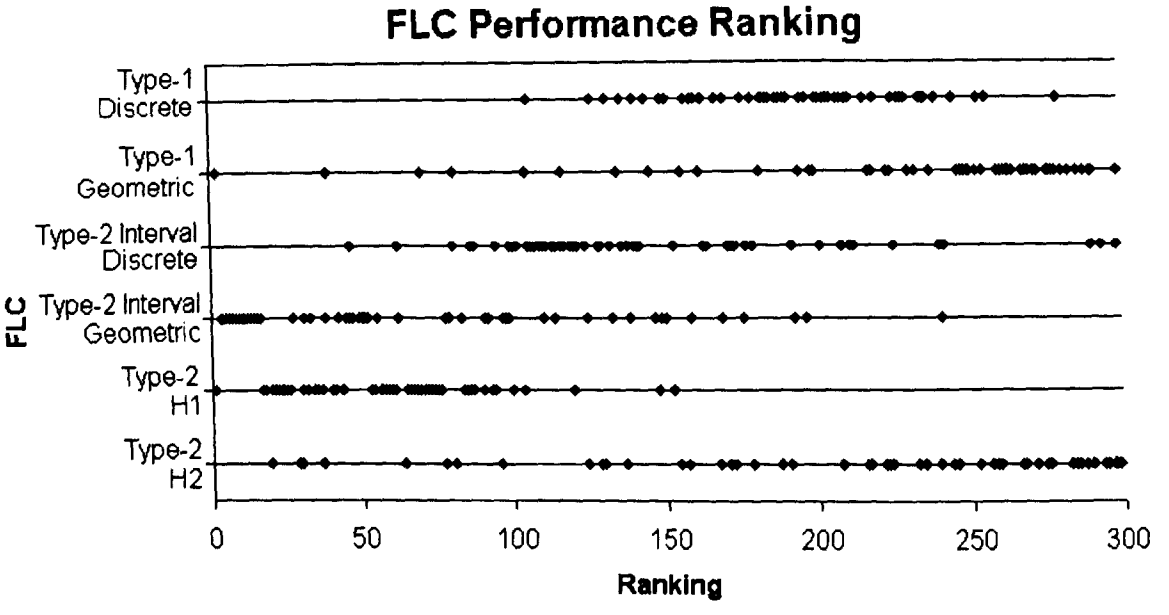


Fig. 5.19. The Rank Position of Each Run of Each FLC.

Robot FLC	Mean Rank
Type-1 Discrete	193.54
Type-1 Geometric	223.70
Interval Discrete	149.94
Interval Geometric	73.62
Type-2 H1	60.22
Type-2 H2	201.98

Table 5.5. The Mean Rank of each of the Robot FLC Over Fifty Runs. All numbers quoted to 2 d.p.

Controller	1	2	3	4	5	6
1	-	-	-	-	-	-
2	0.0005	-	-	-	-	-
3	0.8120	0.0005	-	-	-	-
4	0.0005	0.0005	0.0005	-	-	-
5	0.0005	0.0005	0.0005	0.6172	-	-
6	0.0166	0.0030	0.0557	0.0005	0.0005	-

Table 5.6. Pairwise Mann-Whitney Tests for The Six Controllers. All numbers quoted to 4 d.p.

are significant differences between the controllers and the rankings suggest where some of these difference(s) may lay. To further illuminate the differences between the six controller pairwise comparisons were made using the Mann-Whitney test on the RMSE values with a confidence level of 95%. Results of these comparisons are given in Table 5.6. The comparisons mostly show differences between the controllers. Four comparisons show no differences:

- Controller 1 showed no significant difference to controller 3;
- controller 2 showed no significant difference to controller 4;
- controller 3 showed no significant difference to controller 6, and
- controller 4 showed no significant difference to controller 5.

Caution must be taken when commenting on these pairwise results. Each test has a confidence level of 95%, however, if several test results are put together the error will accumulate, dropping below the 5% of the orginal tests.

5.5.3 Commentary on the Results

When discussing the findings of the experiment described in this Chapter there are two sources of information that are assessed and the findings compared:

- Statistical analysis of the data, and
- The visual information about the FLC paths.

This allows for the comparison of the outcomes of the statistical analysis with the visual assessment of the paths. Where the two agree the findings are reinforced. If little or no agreement is found the doubts must be cast on the analytical method.

The Kruskal-Wallis test showed that there are significant differences between the performances of the six controllers. This is also shown by the paths the robot followed. Since the results did not meet the requirements for a parametric analysis of variance the statistical analysis cannot say definitively where these differences are. However, the mean, median and variance statistics for each FLC given in Table 5.4, the ranking distributions depicted in Figure 5.19 and the pairwise Mann-Whitney test do give information as to where the differences are amongst the controllers:

- The type-2 H1 and type-2 interval geometric FLC outperformed the other controllers.
- The type-2 interval discrete appeared to outperform the remaining three FLC but did not match the performance of its geometric counterpart.

- The type-1 discrete controller outperformed both the type-2 H2 and type-1 geometric FLC which had little performance difference to differentiate between them.

These observed performance differences cannot be said to be statistically significant but are still important if not measurable. In terms of variation of performance over the fifty runs, the type-2 H1 FLC had the lowest standard deviation closely followed by the discrete type-1 FLC. The other four FLC showed similar amount of variation. Figure 5.18 shows these standard deviation results with 95% confidence intervals. This shows that the type-2 H1 FLC has significantly lower standard deviation than all the other FLC apart from the discrete type-1 FLC.

A visual assessment of the paths depicted in Figures D.7 to D.12 yields the following conclusions:

- The type-2 H1 and discrete type-2 interval FLC followed the smoothest and most consistent paths.
- The type-2 H2 FLC also followed smooth paths but with less consistency.
- The discrete type-1 FLC paths showed some jagged paths but with a core of smooth consistent paths.
- The geometric type-1 and geometric type-2 interval FLC paths show similar jagged paths but with a greater dispersal of smooth paths.

Both methods of assessment show that the type-2 H2 and the geometric type-2 FLCs gave the best performance. The paths of the type-2 H2 FLC were very smooth. This is not reflected in the error measurements of the paths. The discrete type-1 FLC showed a good performance in both assessment methods.

The conclusion that is drawn from the experiment presented in this Chapter is that type-2 H1 and the geometric type-2 interval FLC gave the best performance. The type-2 was expected to perform well, however it is not clear why only the H1 FLC gave the best performance and not the H2 configuration. The only real difference between the two hybrids is how the final geometric type-2 fuzzy set is constructed from the combined rule consequents. This seems to give this unexpected difference in performance. In a similar way the geometric type-2 interval FLC outperformed the discrete type-2 interval FLC. The discrete type-2 FLC used type reduction where as the geometric system used the geometric defuzzifier. This accounts for the difference in performance. However, the conclusion that the geometric defuzzifier always outperforms type reduction cannot be drawn. It may be that the geometric defuzzifier gave a better performance in this particular experiment as it is based on the same principle as the type-1 discrete defuzzifier where type-reduction is found with the use of the extension principle. Since the rule base is extended from a discrete type-1

system, it may be that the geometric interval system, with characteristics that are more closely matched to the original system gave a better performance. Recall that the original type-1 rules were tuned by hand. It may be possible to tune the interval rules for use with type reduction and reverse this performance difference. This is also true for the consistency figures of the controllers. The type-2 H1 and the discrete type-1 FLC gave the most consistent performance. However the interval system may give a more consistent performance if the rules were tuned in a different way.

From these discussions it is fair to state that the type-2 H1 FLC gave the best overall performance showing low error and consistency of the paths taken. This is demonstrated both statistically and by a visual inspection of the FLC paths. This is a significant finding since it shows that a type-2 fuzzy system can outperform a type-2 interval system. The discussion of uncertainty and type-2 fuzzy logic presented in Chapter 2 suggested that this should be the case. The results presented here confirm this.

5.6 Discussion

This Chapter has described six fuzzy logic controllers for a mobile robot, two of which used geometric type-2 fuzzy logic. To the author's knowledge these are the first FLC to utilise general type-2 fuzzy logic. It is the geometric approach presented in the previous Chapter that has made this possible. Prior to this thesis, the amount of computation required to type-reduce a type-2 set would have meant that this application, which has to execute every quarter of second, was not possible. This is demonstrated by the timings given in Section 5.3.

An experiment has been presented which was designed to compare the type-2 FLC against comparable type-1 and type-2 interval geometric and discrete controllers. In this experiment one of the hybrid type-2 FLC performed the task more consistently and with less error that the other controllers.

The Hagras study showed similar results to the experiment presented in this Chapter. Hagras focused on the impact of hierarchical controllers and the difference between robot performance in an indoor laboratory compared to being outside on a paved area. The Hagras study did look at the performance of a type-1 FLC compared to a type-2 interval FLC in following the edge of an irregular wall at a distance of 0.35 metres under indoor conditions. The performance of each controller was measured over 8 experimental runs. The type-1 FLC had a mean deviation from the ideal path of 0.020 metres, with a standard deviation of 0.9. The type-2 interval had a mean deviation from the ideal path of 0.016 metres, with a standard deviation of 1.2. These figures for standard deviation broadly fit with the results from this study. The path deviation cannot be

compared as this study did not use absolute distance measurement when calculating error from an ideal path. However the fact that the standard deviation figures are similar suggests that the studies are finding the same results, showing a degree of repeatability.

The experiments were designed to allow an analysis of variance (ANOVA) with multiple comparisons to be performed on the results to show where any significant differences are in the controller performances are. The results did not allow an ANOVA to be performed so a non-parametric substitute was used, the Kruskal-Wallis test. The ANOVA could not be performed as the results showed differences in standard deviation and the distributions were not normal and could not be transformed. Some difference in variance of results was expected, but to lesser degree than actually happened. These non-normal distributions were surprising. With fifty experimental runs being performed it was expected that the results would follow a normal distribution. It may be that the nature of the task makes normal distributions of results less likely. For example, here is only a small area between the obstacle and ideal path for errors to occur. There is a much larger area outside the ideal path where errors could occur. It may also be the case that the rule base is focused on controlling the robot around the ideal path area and does not provide tight enough control once the robot moves away from this area. Both of these factors suggest the results may be skewed to one side or the other. However, the results did not show a uniform skew to either side and so the form of the distributions remains unexplained.

This Chapter has presented an experiment designed to assess the performance of a number of mobile robot controllers. The experiment demonstrated that a type-2 FLC can outperform other fuzzy controllers. The next Chapter draws together the key points from this work and analyses the importance of this work to the field of fuzzy logic.

Chapter 6

Conclusions and Discussion

This Chapter concludes the thesis by summarising the key points and outcomes of the research. A discussion of directions for future work on geometric fuzzy systems is discussed.

The main outcome of this thesis is the geometric approach to type-2 fuzzy systems. The geometric approach presented in thesis has allowed the first type-2 fuzzy logic controller to be built. This is a significant step for type-2 fuzzy logic. Furthermore this first type-2 fuzzy logic controller has been compared to type-1 and type-2 interval controllers. The geometric approach makes this comparison possible. The research is also assessed against the research hypothesis that was stated in Chapter 1 as:

"Type-2 fuzzy logic systems offer a great deal in terms of modelling uncertain concepts and inferencing under uncertain conditions. However, the computational complexity of type-2 fuzzy logic is arresting the research and development of such systems. Geometric methods can resolve these computational problems making it possible for type-2 fuzzy logic to be applied in a time critical domain such as control."

This Chapter summarises the arguments presented throughout this thesis that directly relate to this hypothesis. The central arguments supporting the research hypothesis, which will be expanded upon, are:

(i) *Geometric Fuzzy Sets Improve Accuracy.* It is argued in this thesis that the use of discrete fuzzy sets results in a loss of information about the fuzzy set resulting in inaccuracies throughout the inferencing process. Geometric fuzzy sets can model fuzzy sets over a continuous domain, removing the need for discretisation and therefore increasing accuracy. Examples and discussion of these examples argue this point.

(ii) *Geometric Type-2 Fuzzy Logic Systems are Computational Efficient.* It is argued in this thesis that the complexity of type-2 fuzzy sets results from defining operations with use of

the extension principle. The extension principle is not applicable to geometric type-2 fuzzy systems. Operations are instead defined by modelling the third dimension of the type-2 sets as a third geometric dimension. This results in a significant reduction in the computational complexity of type-2 fuzzy systems.

(iii) *Hybridised Type-2 Fuzzy Logic Systems Require Minimal Computation.* Experiments reported in this thesis suggest that the most efficient form of type-2 fuzzy systems are hybridised type-2 fuzzy systems. Prior to this thesis, no examples of generalised type-2 fuzzy logic control appear in the literature.

(iv) *Hybrid Type-2 Fuzzy Logic Control is Possible.* In this thesis the design of two hybrid type-2 fuzzy logic controllers is described. The performance of these controllers were subsequently evaluated.

(v) *Hybrid Type-2 Fuzzy Logic Can Outperform Other Fuzzy Technologies.* Experimental data, both visual and statistical, reported in this thesis suggests that a hybrid type-2 fuzzy logic controller gave a smoother and more consistent performance in the task of edge following.

Each of these points will now be explored in greater detail.

6.1 Geometric Fuzzy Sets Improve Accuracy

Prior to this work, fuzzy logic systems, deployed on either hardware or software, have had to use discrete fuzzy sets. It may be the case that the concepts being modelled by the sets have a naturally discrete domain. Often, however, such sets are based over a continuous domain. Discretising a fuzzy set over a continuous domain results in a loss of information about the form of the set. Inaccuracies are introduced to the set model before any inferencing has been performed. During inferencing these errors are compounded. A worked example, presented in Chapter 3, demonstrates this loss of information and inaccuracies that result from discretisation. This loss of information can be significant. The worked example showed an inaccuracy equivalent to approximately 4.7% of the support of the fuzzy set. This represents a significant error.

Geometric fuzzy sets do not require discretisation, therefore offer more accurate representations of fuzzy membership functions. For membership functions that do not contain any curved elements, i.e. piecewise linear membership functions, a type-1 geometric fuzzy set offers a completely accurate representation. Geometric type-2 interval fuzzy sets also offer a completely accurate representation when both the upper and lower membership functions are piecewise linear. Geometric type-2 interval and type-1 fuzzy sets can also improve the accuracy of the representation of membership functions that are not piecewise linear. This is illustrated by a worked example in Chapter

3. Errors that occur when a set is discretised are compounded during the inferencing process. Using the geometric model helps reduce such errors. For example, when discrete fuzzy sets are combined, the areas between the discrete points can be incorrectly calculated. Geometric fuzzy sets have no discrete points, therefore, such inaccuracies cannot occur. This point is illustrated by Figure 3.3 of this thesis.

The accuracy issues identified for discrete type-1 fuzzy sets are magnified in discrete type-2 fuzzy sets. Increasing the number of points in a discrete type-2 fuzzy set has a significant impact on computational complexity of a type-2 fuzzy system. In particular, the computational cost of type-reduction increases significantly as the number of discrete points in a type-2 fuzzy set increases. This relationship is explored in detail in Chapter 2 of this thesis. There is an significant computational gain to be made from using less points in a discrete type-2 fuzzy set. Reducing the number of points in a discrete set has a serious impact on the accuracy of the representation of that set. In the example in Chapter 2 two discrete type-2 fuzzy sets are given which are both modelling an identical continuous fuzzy set. One set contains three discrete points in the primary and secondary domains. The other set contains six discrete points in the primary and secondary domains. The computational cost of type-reducing the two sets varies by a factor of over 1700. The difference between the defuzzified values of the two sets is approximately equal to 9% of the support of the fuzzy sets.

The concept of accuracy is more complicated for geometric type-2 fuzzy sets. Analogous to the 2D concept of a piecewise linear function is the 3D concept of a planar surface. A planar surface is a 3-dimensional surface that is constructed from a series of smaller surfaces, each of which lie entirely on a single plane. If the membership function of a type-2 fuzzy set is a planar surface then a geometric type-2 fuzzy set can represent that membership function with complete accuracy. For non-planar surfaces geometric type-2 fuzzy sets offer an approximation of that surface. The accuracy of the geometric model cannot be directly compared with that of the discrete model for type-2 fuzzy sets. This is because the aim of the type-reduction operation is to achieve a different goal to the geometric defuzzifier.

6.2 Geometric Type-2 Fuzzy Logic Systems are Computational Efficient

There are two aspects of type-2 fuzzy logic that are responsible for the computational problems of type-2 fuzzy logic, inferencing (join and meet) and type-reduction.

- *Inferencing.* The join and meet operations are the building blocks of three aspects of the type-2 fuzzy inferencing process, combination of antecedents, implication and the combi-

nation of consequents. The efficiency of these two operations determines the efficiency of the inferencing process. The optimised join and meet operations given in Section 4.1 give computationally efficient definitions for these operations. The join and meet operations for geometric type-2 fuzzy sets are geometric interpretations of these efficient operations. The geometric type-2 implication and combination of consequent operations provide 3-dimensional geometric interpretations of these efficient operations.

- *Type-Reduction.* As the number of points in a discrete type-2 fuzzy set increase the amount of computation required to type-reduce that set increases significantly. The growth in computational complexity for the geometric defuzzifier is much less of an issue than with type-reduction. The computational cost of the geometric defuzzifier grows in a linear fashion with the number of triangular facets used to model the surface. It is the output processing stage of type-2 fuzzy logic where the geometric approach gives huge reductions in computational complexity. This point was emphasised in Section 5.3 where type-reduction was shown to take 99.9% of the overall inferencing time of an example discrete type-2 fuzzy logic system.

The argument presented in this thesis is that geometric type-2 fuzzy system are computationally efficient. Most of the efficiency gain over discrete systems stems from the type-2 geometric defuzzifier.

6.3 Hybridised Type-2 Fuzzy Logic Systems Require Minimal Computation

The research hypothesis states that geometric methods can be used to overcome the computational problems of type-2 fuzzy logic. This does not necessarily mean that using only geometric methods will achieve the best possible reduction in computational complexity. Hybridised type-2 fuzzy logic presented in Chapter 4 allows the most efficient aspects of discrete and geometric type-2 fuzzy logic to be combined in one system. By defining operations to transform between the discrete and geometric representations it is possible to select the component configuration, given the current definitions, with the minimum computational requirements for a particular application.

An example of the design an optimal hybridised type-2 fuzzy system is presented in Chapter 5 of this thesis. Each of the possible discrete and geometric system components is tested under simulated conditions. The execution speed of each component was measured under controlled conditions. The system configuration with the lowest possible execution speed was then selected. This process resulted in a computationally optimal type-2 fuzzy logic system. It is argued in this

thesis that hybrid type-2 fuzzy logic results in a system requiring the minimum computation since the most efficient system components can be selected and combined.

6.4 Hybrid Type-2 Fuzzy Logic Control is Possible

Prior to this thesis there were no reported examples of generalised type-2 fuzzy logic being applied to a control problem. The argument presented throughout this thesis is that the computational complexity of discrete type-2 fuzzy systems renders them unsuitable for such applications. Having demonstrated that hybrid type-2 fuzzy systems require minimal computation, the application of such systems to a control problem was then investigated.

Type-2 fuzzy sets model uncertain concepts. If type-2 systems are to be applied in the control domain it is only sensible that the application chosen has inherent uncertainties. The problem of mobile robot control was identified as an application containing the necessary uncertainties. The robotic platform chosen for the controller was the Pioneer 2 robot. The real-time constraint for this platform requires a control action to be calculated every 250 milliseconds. This means that the sensor readings have to be taken, passed to the FLC, reasoned with, a decision taken and the decision actioned by the robot once every quarter of a second. Achieving such execution speeds with a discrete type-2 fuzzy system would require hardware resources far beyond the embedded PC on the Pioneer 2 robots. The type-reduction operation of the sets used in the controllers would require 8.88×10^{34} embedded sets to be enumerated. The hybrid approach offers minimal computation, however, the ability to perform the required computation in under a quarter of a second on the available hardware was not known until the systems were implemented. Once implemented it became clear that these time constraints present no problems to the hybrid controllers. Experiments (reported in Chapter 5) were conducted using two hybrid fuzzy logic robot controllers supporting the argument that hybrid type-2 fuzzy logic can indeed be applied to a control problem.

6.5 Hybrid Type-2 Fuzzy Logic Can Outperform Other Fuzzy Technologies

Type-2 fuzzy logic models and reasons with uncertain concepts. Such a system should therefore improve the decision making process under uncertain conditions. The application area of mobile robotics meets this criteria. In Chapter 5 an experiment was designed to compare the results of type-2, type-2 interval and type-1 fuzzy logic controllers of various configurations. The result of the this experiment show the hybrid type-2 fuzzy logic controller H1 gave the best control performance of those that were tested. This result was borne out by both a visual inspection of

the paths the robot controllers took and a statistical analysis of those paths. The investigation that forms Chapter 5 of this thesis supports the conclusion that a hybrid type-2 fuzzy system can outperform its fuzzy counterparts.

6.6 Further Work

This thesis has presented the novel geometric approach to fuzzy logic systems. This study has given rise to a number of areas where further work in this field is warranted.

- The geometric approach is currently limited to modelling fuzzy operations based on the minimum and maximum t-norm and t-conorm. The approach may be more widely accepted by practitioners if operations based on other operators had a geometric interpretation. In particular a geometric model of the product t-norm would be useful as this t-norm is often applied in the control domain.

- More studies should be conducted into the performance of geometric fuzzy systems. The study presented in this thesis shows the type-2 fuzzy systems can outperform type-1 and type-2 interval systems. This study demonstrated an improved control performance of a mobile robot controller over the other systems. The conditions and factors that facilitate this performance improvement should be studied.

- The geometric defuzzification and type-reduction offer two completely different approaches to output processing in a type-2 fuzzy logic system. The characteristics of these two techniques should be studied. This may inform the design and use of these techniques.

- This thesis reports the software implementation of a type-2 fuzzy logic controller. This is a major development in the field of type-2 fuzzy logic. The possibility of a hardware implementation of a type-2 fuzzy system is now apparent. Such a development would enable type-2 fuzzy logic systems to be a applied to a wider range of applications, particularly in the field of control.

- Currently there are no methods for finding the optimal levels of discretisation for type-2 fuzzy sets. There is a particular problem with type-2 fuzzy sets as they have a primary and a secondary domain. Methods for determining the optimal levels in both domains for a particular application would be very useful.

6.7 Summary

This thesis is concerned with the definition and application of geometric fuzzy logic systems. Throughout the thesis it has been argued that the process of discretising fuzzy sets leads to inaccuracies in the set model and the inferencing process. Geometric fuzzy sets, which do not require discretisation have been proposed. The set model and inferencing process of such system is completely accurate for non-curved membership functions. The major contribution of this work has been the use of geometry in reducing the computational complexity of type-2 fuzzy logic. The novel geometric type-2 fuzzy logic methods have allowed, for the first time, generalised type-2 fuzzy logic to be applied to a control problem. The results show that type-2 fuzzy logic can outperform both type-1 and type-2 interval fuzzy logic. This result is a promising outcome for the field of type-2 fuzzy logic.

Bibliography

Bashein, G. & Detmer, P. R. (1994), Centroid of a Polygon, *in* P. S. Heckbert, ed., 'Graphics Gems IV', Academic Press, Massachusetts, pp. 3 – 6.

Bentley, J. L. & Ottmann, T. A. (1979), 'Algorithms for reporting and counting geometric intersections', *IEEE Transactions on Computing* **28**(9), 643–647.

Bourke, P. (1988), Calculating the area and centroid of a polygon). Available at http:// astronomy.swin.edu.ac.uk/~pbourke/geometry/lineline/polyarea/ or on CDROM from http:// astronomy.swin.edu.ac.uk/~pbourke/.

Bourke, P. (1989), Intersection point of two lines (2 dimensions). Available at http:// astronomy.swin.edu.ac.uk/ ~ pbourke/geometry/lineline/2d/ or on CDROM from http:// astronomy.swin.edu.ac.uk/~pbourke/.

Coupland, S., Gongora, M., John, R. & Wills, K. (2006), A Comparative Study of Fuzzy Logic Controllers for Autonomous Robots, *in* 'Proc. IPMU 2006', Paris, France. Accepted for publication.

Coupland, S. & John, R. (2003), An Approach to Type-2 Fuzzy Arithmetic, *in* 'Proc. UK Workshop on Computational Intelligence', pp. 107 – 114.

Coupland, S. & John, R. (2004*a*), A New and Efficient Method for the Type-2 Meet Operation, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 959 – 964.

Coupland, S. & John, R. (2004*b*), Fuzzy Logic and Computational Geometry, *in* 'Proc. RASC 2004', Nottingham, England, pp. 3 – 8.

Coupland, S. & John, R. (2005*a*), Geometric Interval Type-2 Fuzzy Systems, *in* 'Proc. EUSFLAT 2005', Barcelona, Spain, pp. 449 – 454.

Coupland, S. & John, R. (2005*b*), Towards More Efficient Type-2 Fuzzy Logic Systems, *in* 'Proc. FUZZ-IEEE 2005', Reno, NV, USA, pp. 236 – 241.

Coupland, S. & John, R. (2006*a*), An Investigation into Alternative Methods for the Defuzzification of an Interval Type-2 Fuzzy Set, *in* 'Proc. FUZZ-IEEE 2006', Vancouver, Canada. Accepted for publication.

Coupland, S. & John, R. (2006*b*), 'Geometric Type-1 and Type-2 Fuzzy Logic Systems', *IEEE Transactions on Fuzzy Systems* . Accepted for publication.

Cyrus, M. & Beck, J. (1978), 'Generalized Two and Three Dimensional Clipping', *Computers and Graphics* **3**(1), 23 – 28.

Dubois, D. & Prade, H. (1980), *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York.

Dubois, D. & Prade, H. (1982), 'A Class of Fuzzy Measures Based on Triangular-Norms', *Int. Journal of General Systems* **8**, 43–61.

Figueroa, J., Posada, J., Soriano, J., Melgarejo, M. & Rojas, S. (2005), A Type-2 Fuzzy Controller for Tracking Mobile Objects in the Context of Robotic Soccer Games, *in* 'Proc. FUZZ-IEEE 2005', Reno, AZ, USA, pp. 359 – 364.

Foley, J. & Van Dam, A. (1982), *Fundementals of Interactive Computer Graphics*, Addison-Wesley, Reading, Massachusetts, pp. 146 – 149.

Garibaldi, J. M., Musikasuwan, S. & Ozen, T. (2005), The Association between Non-Stationary and Interval Type-2 Fuzzy Sets: A Case Study, *in* 'Proc. FUZZ-IEEE 2005', Reno, NV, USA, pp. 224–229.

Garibaldi, J. M., Musikasuwan, S., Ozen, T. & John, R. I. (2004), A Case Study to Illustrate the Use of Non-Convex Membership Functions for Linguistic Terms, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 1403 – 1408.

Garibaldi, J., Westgate, J., Ifeachor, E. & Greene, K. (1997), 'The Development and Implementation of an Expert System for the Analysis of Umbilical Cord Blood', *Artificial Intelligence in Medicine* **10**(2), 129 – 144.

Greenfield, S., John, R. & Coupland, S. (2005), A Novel Sampling Method for Type-2 Defuzzification, *in* 'Proc. UKCI 2005', pp. 120 – 127.

Guigue, P. & Devillers, O. (2003), 'Fast and Robust Triangle-Triangle Overlap Test Using Orientation Predicates', *Journal of Graphics Tools* **8**(1), 25 – 32.

Hagras, H. (2004), 'A Hierarchical Type-2 Fuzzy Logic Control Architecture for Autonomous Mobile Robots', *IEEE Transactions on Fuzzy Systems* **12**, 524–539.

Innocent, P. & John, R. I. (2004), 'Computer Aided Fuzzy Medical Diagnosis', *Information Sciences* **162**, 81 – 104.

John, R. (1996), Type-2 inferencing and community transport scheduling, *in* 'Proc. Fourth European Congress on Intelligent Techniques and Soft Computing, EUFIT 1996', Aachen, Germany, pp. 1369 – 1372.

John, R. (1998*a*), Type–2 Fuzzy Sets for Knowledge Representation and Inferencing, *in* 'Proc. 7th Intl. Conf. on Fuzzy Systems FUZZ-IEEE 1998', pp. 1003–1008.

John, R. (1998*b*), 'Type 2 Fuzzy Sets: An Appraisal of Theory and Applications', *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems* **6**(6), 563–576.

John, R. (1999*a*), 'Fuzzy sets of type-2', *Journal of Advanced Computational Intelligence* **3**(6), 499 – 508.

John, R. (1999*b*), 'Type-2 fuzzy sets', *Expert Update* **2**(2). ISSN 1465-4091.

John, R. I. (2000), Perception Modelling Using Type-2 Fuzzy Sets, PhD thesis, De Montfort University.

John, R. I., Innocent, P. R. & Barnes, M. R. (2000), 'Neuro-fuzzy clustering of radiographic tibia image data using type-2 fuzzy sets', *Information Sciences* **125**, 203–220.

John, R., Innocent, P. & Barnes, M. (1998), Type–2 Fuzzy Sets and Neuro-Fuzzy Clustering of Radiographic Tibia Images, *in* 'Proc. FUZZ-IEEE 1998', pp. 1373–1376.

John, R. & Lake, S. (2001*a*), Modelling nursing perceptions using type-2 fuzzy sets, *in* 'EUROFUSE 2001 Workshop on Preference Modelling and Applications', pp. 241 – 246.

John, R. & Lake, S. (2001*b*), Type-2 fuzzy sets for modelling nursing intuition, *in* 'Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference', pp. 1920 – 1925.

Karnik, N. N. & Mendel, J. M. (1998*a*), An Introduction to Type-2 Fuzzy Logic Systems, Technical report, University of Southern California.

Karnik, N. N. & Mendel, J. M. (1998*b*), Introduction to Type-2 Fuzzy Logic Systems, *in* 'Proc. IEEE World Congress on Computational Intelligence', Anchorage, Alaska, USA, pp. 915–920.

Karnik, N. N. & Mendel, J. M. (1998*c*), Type-2 Fuzzy Logic Systems: Type-Reduction, *in* 'Proc. IEEE Systems, Man and Cybernetics', pp. 2046–2051.

Karnik, N. N. & Mendel, J. M. (1999), 'Application of Type-2 Fuzzy Logic System to Forecasting of Time-Series', *Information Sciences* **120**, 89 – 111.

Karnik, N. N. & Mendel, J. M. (2001*a*), 'Centroid of a type-2 fuzzy Set', *Information Sciences* **132**, 195–220.

Karnik, N. N. & Mendel, J. M. (2001*b*), 'Operations on Type-2 Fuzzy Sets', *Fuzzy Sets and Systems* **122**, 327–348.

Karnik, N. N., Mendel, J. M. & Liang, Q. (1999), 'Type-2 Fuzzy Logic Systems', *IEEE Transactions on Fuzzy Systems* **7**, 643–658.

Klir, G. J. & Folger, T. A. (1988), *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall.

Kosko, B. (1997), *Fuzzy Engineering*, Prentice Hall, New Jersey.

Liang, Q. & Mendel, J. M. (2000*a*), 'Equalization of Nonlinear Time-Varying Channels Using Type-2 Fuzzy Adaptive Filters', *IEEE Transactions on Fuzzy Systems* **8**, 551–563.

Liang, Q. & Mendel, J. M. (2000*b*), 'Interval Type–2 Fuzzy Logic Systems: Theory and Design', *IEEE Transactions on Fuzzy Systems* **8**, 535–549.

Liang, Q. & Mendel, J. M. (2001), 'Mpeg vbr video traffic modeling and classification using fuzzy technique', *IEEE Transactions on Fuzzy Systems* **9**, 183–193.

Luigi Di Lascio, A. G. & Nappi, A. (2005), Medical differential diagnosis through Type-2 Fuzzy Sets, *in* 'Proc. FUZZ-IEEE 2005', Reno, NV, USA, pp. 371 – 376.

Lynch, C., Hagras, H. & Callaghan, V. (2005), Embedded Type-2 FLC for Real-Time Speed Control of Marine and Traction Diesel Engines, *in* 'Proc. FUZZ-IEEE 2005', Reno, AZ, USA, pp. 347 – 352.

Mamdani, E. H. (1974), 'Application of Fuzzy Algorithms for Simple Dynamic Plant', *Proc. IEE* **121**, 1585 – 1588.

Mamdani, E. H. (1977), 'Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis', *IEEE Transactions on Computers* **26**(12), 1182 – 1191.

Melgarejo, M., Garcio, A. & Pena-Reyes, C. (2004), Pro-Two: A Hardware Based Platform For Real Time Type-2 Fuzzy Inference, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 977 – 982.

Melgarejo, M. & Pena-Reyes, C. (2004), Hardware Architecture and FPGA Implementation of a Type-2 Fuzzy System, *in* 'Proc. GLSVSLI 2004', Boston, Massachusetts, USA, pp. 458 – 461.

Melin, P. & Castillo, O. (2003), Fuzzy Logic for Plant Monitoring and Diagnostics, *in* 'Proc. NAFIPS 2003', pp. 20 – 25.

Melin, P. & Castillo, O. (2004), Intelligent Control of Non-Linear Dynamic Plants Using Type-2 Fuzzy Logic and Neural Networks, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary.

Mendel, J. M. (1999), Computing With Words, When Words Mean Different Things to Different People, *in* 'Proc. of Third International ICSC Symposium on Fuzzy Logic and Applications', Rochester Univ., Rochester, NY.

Mendel, J. M. (2001*a*), The Perceptual Computer: an Architecture for Computing With Words, *in* 'Proc. FUZZ-IEEE 2001', Melbourne, Australia.

Mendel, J. M. (2001*b*), *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, Upper Saddle River, NJ.

Mendel, J. M. (2003), Fuzzy sets for words: a new beginning, *in* 'Proc. FUZZ-IEEE 2003', St. Louis, MO, USA, pp. 37 – 42.

Mendel, J. M. & John, R. I. (2002), 'Type-2 Fuzzy Sets Made Simple', *IEEE Transaction on Fuzzy Systems* **10**(2), 117–127.

Mitchell, H. B. (2005), 'Pattern Recognition Using Type-II Fuzzy Sets', *Information Sciences* **170**, 409–418.

Mizumoto, M. & Tanaka, K. (1976), 'Some properties of fuzzy set of type-2', *Information and control* **31**, 312–340.

Mizumoto, M. & Tanaka, K. (1981), 'Fuzzy Sets of Type 2 Under Algebraic Product and Algebraic Sum', *Fuzzy Sets and Systems* **5**, 277–290.

Mizumoto, M. & Tanaka, K. (1982), 'Comparison of Fuzzy Reasoning Methods', *Fuzzy Sets and Systems* **8**, 253–283.

Möller, T. (1997), 'A Fast Triangle-Triangle Intersection Test', *Journal of Graphics Tools* **2**(2), 25 – 30.

Musikasuwan, S., Ozen, T. & Garibaldi, J. (2004), An investigation into the effect of number of model parameters on performance in type-1 and type-2 fuzzy logic systems, *in* 'Proc.

10th Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU 2004)', Perugia, Italy, pp. 1593 – 1600.

Ostle, B. & Malone, L. C. (1988), *Statistics in Research.*, fourth edn, Science Press.

Ott, R. L. & Longnecker., M. (2001), *An Introduction to Statistical Methods and Data Analysis.*, fifth edn, Duxbury.

Ozen, T. & Garibaldi, J. M. (2003), Investigating Adaptation in Type-2 Fuzzy Logic Systems Applied to Umbilical Acid-Base Assessment, *in* 'Proc. of the 2003 European Symposium on Intelligent Technologies', Oulu, Finland, pp. 289–294.

Ozen, T. & Garibaldi, J. M. (2004), Effect of Type-2 Fuzzy Membership Function Shape on Modelling Variation in Human Decision Making, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 971 – 976.

Reznik, L. (1997), *Fuzzy Controllers*, Reed Elsevier.

Schwartz, D. G. (1985), 'The case for an interval-based representation of linguistic truth', *Fuzzy Sets and Systems* **17**, 153 – 165.

Seising, R. (2005), 1965 "'Fuzzy Sets"' appear A Contribution to the 40th Anniversary, *in* 'Proc. FUZZ-IEEE 2005', Reno, AZ, USA, pp. 5 – 10.

Sugeno, M. & Kang, G. T. (1988), 'Structure Identification of Fuzzy Model', *Fuzzy Sets and Systems* **28**(1), 15–33.

Sugeno, M. & Nishida, M. (1985), 'Fuzzy Control of Model Car', *Fuzzy Sets and Systems* **16**, 103–113.

Sutherland, I. & Hodgman, G. (1974), 'Reentrant Polygon Clipping', *Communications of the ACM* **17**(1), 32 – 42.

Takagi, H. (1994), Application of neural networks and fuzzy logic to consumer products, *in* 'Fuzzy Logic Technology and Applications', The Institute of Electrical and Electronics Engineers, New York.

Takagi, T. & Sugeno, M. (1985), 'Fuzzy Identification of Systems and Its Application to Modeling and Control', *IEEE Transactions on Systems, Man and Cybernetics* **15**(1), 116–132.

Thompson, S. (1999), *Haskell: The Craft of Functional Programming*, second edn, Addison-Wesley, Harlow, UK.

Türkşen, I. B. (1993*a*), 'Interval-valued fuzzy sets and fuzzy connectives', *Interval Computations* **4**, 35 – 38.

Türkşen, I. B. (1993*b*), Interval-valued fuzzy uncertainty, *in* 'Proc. Fifth IFSA World Congress', Seoul, Korea, pp. 35 – 38.

Türkşen, I. B. (1995), Knowledge representation and approximate reasoning with type ii fuzzy sets., *in* 'Proc. FUZZ-IEEE 1995', Vol. 2, Yokohama, Japan, pp. 1911 – 1917.

Türkşen, I. B. (2002), 'Type 2 Representation and Reasoning for CWW', *Fuzzy Sets and Systems* **127**, 17–36.

Van den Broek, P. (1997), Fuzzy Reasoning with Continuous Piecewise Linear Membership Functions, *in* 'Proc. NAFIPS 1997', pp. 371 – 376.

Van den Broek, P. (1999), Efficient Algorithms for Approximate Reasoning, *in* 'Proc. ICONIP 1999', pp. 292 – 297.

Weiler, K. & Atherton, P. (1977), Hidden surface removal using polygon area sorting, *in* 'Proc. Siggraph 1977', San Jose, California, USA, pp. 214 – 222.

Wu, D. & Tan, W. W. (2004), A Type-2 Fuzzy Logic Controller for the Liquid-level Process, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 953 – 958.

Wu, D. & Tan, W. W. (2005), Computationally Efficient Type-Reduction Strategies for a Type-2 Fuzzy Logic Controller, *in* 'Proc. FUZZ-IEEE 2005', Reno, AZ, USA, pp. 353 – 358.

Wu, H. & Mendel, J. M. (2001), Introduction to Uncertainty Bounds and Their Use in the Design of Interval Type-2 Fuzzy Logic Systems, *in* 'Proc. FUZZ-IEEE 2001', Melbourne, Australia.

Wu, H. & Mendel, J. M. (2002), 'Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems', *IEEE Transactions on Fuzzy Systems* pp. 622–639.

Zadeh, L. A. (1975*a*), 'The Concept of a Linguistic Variable and its Application to Approximate Reasoning', *Information Sciences* **8**, 199–249.

Zadeh, L. A. (1975*b*), 'The Concept of a Linguistic Variable and its Application to Approximate Reasoning – II', *Information Sciences* **8**, 301–357.

Zadeh, L. A. (1975*c*), 'The Concept of a Linguistic Variable and its Application to Approximate Reasoning – III', *Information Sciences* **9**, 43–80.

Zadeh, L. A. (1996), 'Fuzzy Logic = Computing with Words', *IEEE Transactions on Fuzzy Systems* **4**, 103–111.

Zadeh, L. A. (1999), 'From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions', *IEEE Transactions on Circuits and Systems – I:Fundamental Theory and Applications* **45**, 105–119.

Appendix A

Proofs

A.1 Demonstration That Equation 4.6 ≡ Equation 4.7

This Section demonstrates that

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\theta)) \vee (\mu_{\tilde{A}}(x,\theta) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{A.1}$$

is equivalent to

$$\mu_{\tilde{A}\sqcup\tilde{B}}(x,\theta) = (\mu_{\tilde{A}}(x,\theta) \vee \mu_{\tilde{B}}(x,\theta)) \wedge (\mu_{\tilde{A}}(x,\upsilon_1) \wedge \mu_{\tilde{B}}(x,\upsilon_2)) \tag{A.2}$$

To demonstrate this fact let the four values $\mu_{\tilde{A}}(x,\upsilon_1)$, $\mu_{\tilde{A}}(x,\theta)$, $\mu_{\tilde{B}}(x,\upsilon_2)$ and $\mu_{\tilde{B}}(x,\theta)$ each take a distinct value from the set $\{0.4, 0.6, 0.8, 1\}$. This will give all possible permutations of the terms in equations concerned. These permutations are given in Table A.1 where $\mu_{\tilde{A}}(x,\upsilon_1) = i$, $\mu_{\tilde{B}}(x,\theta) = j$, $\mu_{\tilde{A}}(x,\theta) = k$ and $\mu_{\tilde{B}}(x,\upsilon_2) = l$. Because $\mu_{\tilde{A}}(x,\upsilon_1) \geq \mu_{\tilde{A}}(x,\theta)$ and $\mu_{\tilde{B}}(x,\upsilon_2) \geq \mu_{\tilde{B}}(x,\theta)$ only case where these two facts are true need to be taken in to account. Table A.1 shows that for every tuple of values for which these facts hold Equations 4.6 and 4.7 are equivalent.

				A	B	C	D	
i	j	k	l	$(i \wedge j) \vee (k \wedge l)$	$(j \vee k) \wedge (i \wedge l)$	$i > k \& l > j$	A = B	C ⇒ D
1	0.8	0.6	0.4	0.8	0.4	0	0	1
1	0.8	0.4	0.6	0.8	0.6	0	0	1
1	0.6	0.8	0.4	0.6	0.4	0	0	1
1	0.6	0.4	0.8	0.6	0.6	1	1	1
1	0.4	0.8	0.6	0.6	0.6	1	1	1
1	0.4	0.6	0.8	0.6	0.6	1	1	1
0.8	1	0.6	0.4	0.8	0.4	0	0	1
0.8	1	0.4	0.6	0.8	0.6	0	0	1
0.8	0.6	1	0.4	0.6	0.4	0	0	1
0.8	0.6	0.4	1	0.6	0.6	1	1	1
0.8	0.4	1	0.6	0.6	0.6	0	1	1
0.8	0.4	0.6	1	0.6	0.6	1	1	1
0.6	1	0.8	0.4	0.6	0.4	0	0	1
0.6	1	0.4	0.8	0.6	0.6	0	1	1
0.6	0.8	1	0.4	0.6	0.4	0	0	1
0.6	0.8	0.4	1	0.6	0.6	1	1	1
0.6	0.4	1	0.8	0.8	0.6	0	0	1
0.6	0.4	0.8	1	0.8	0.6	0	0	1
0.4	1	0.8	0.6	0.6	0.4	0	0	1
0.4	1	0.6	0.8	0.6	0.4	0	0	1
0.4	0.8	1	0.6	0.6	0.4	0	0	1
0.4	0.8	0.6	1	0.6	0.4	0	0	1
0.4	0.6	1	0.8	0.8	0.4	0	0	1
0.4	0.6	0.8	1	0.8	0.4	0	0	1

Table A.1. All Possible Permutations of the Term in Equations 4.6 and 4.7.

Appendix B

The Fuzzy Sets Employed in the Fuzzy logic Controllers

B.1 Type-1 Discrete FLC

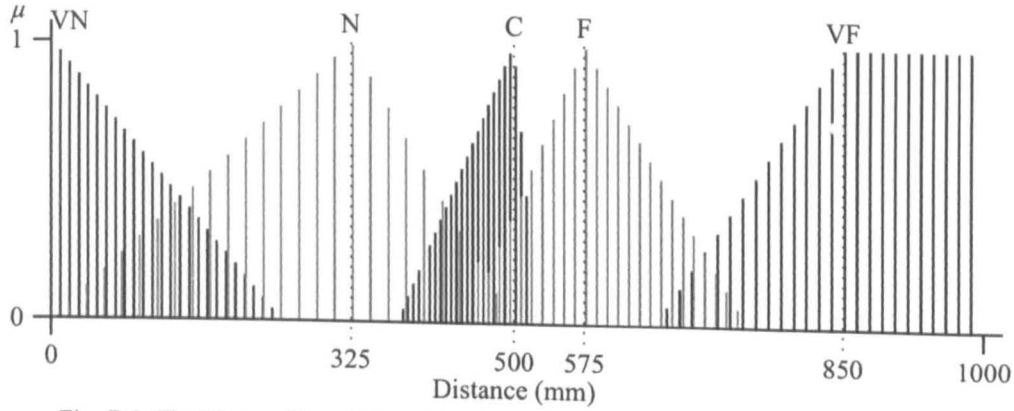


Fig. B.1. The Discrete Type-1 Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

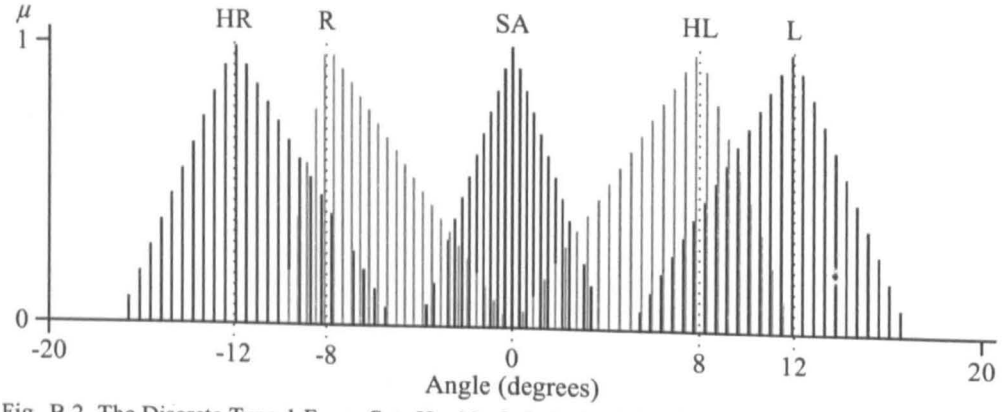


Fig. B.2. The Discrete Type-1 Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

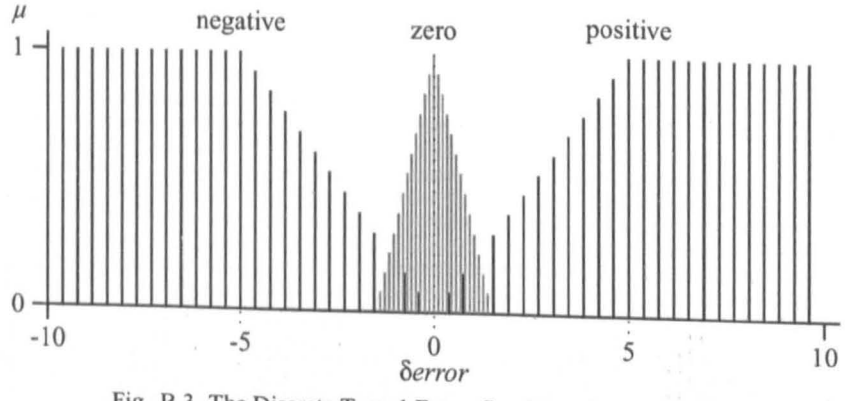


Fig. B.3. The Discrete Type-1 Fuzzy Sets Negative, Zero, Positive

B.2 Type-1 Geometric FLC

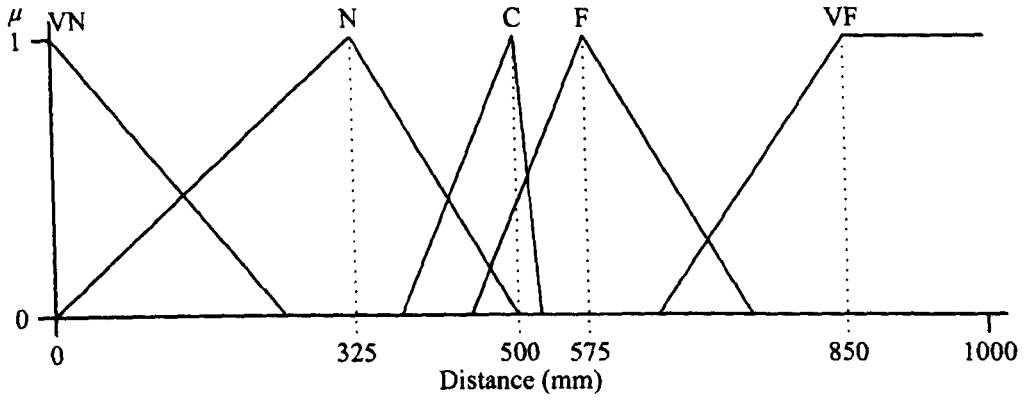


Fig. B.4. The Geometric Type-1 Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

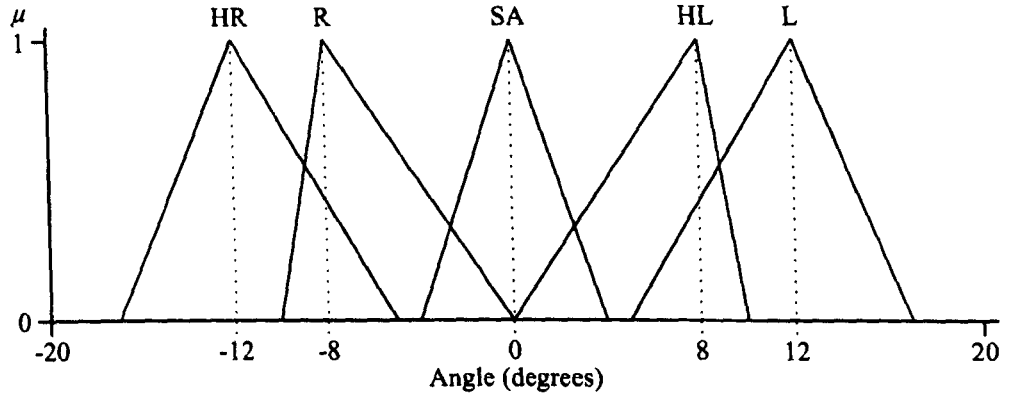


Fig. B.5. The Geometric Type-1 Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

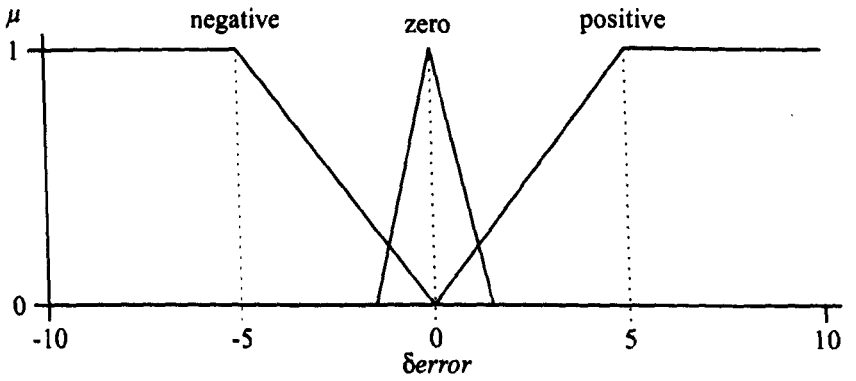


Fig. B.6. The Geometric Type-1 Fuzzy Sets Negative, Zero, Positive

B.3 Type-2 Interval Discrete FLC

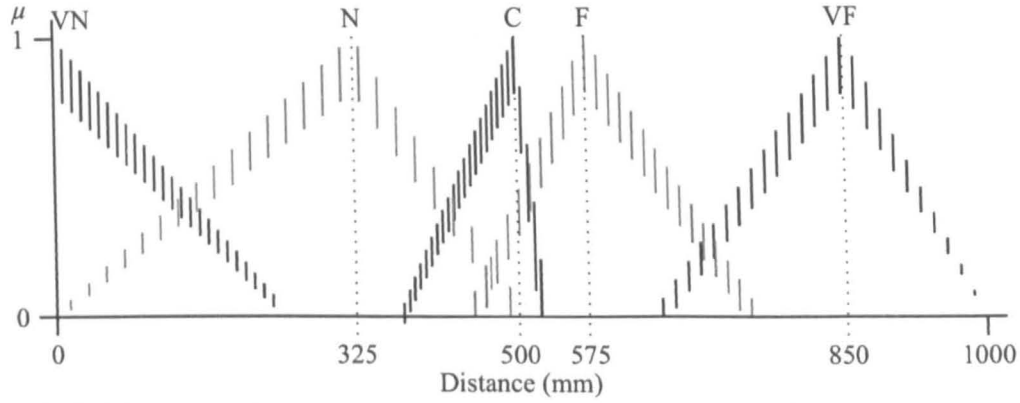


Fig. B.7. The Discrete Type-2 Interval Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

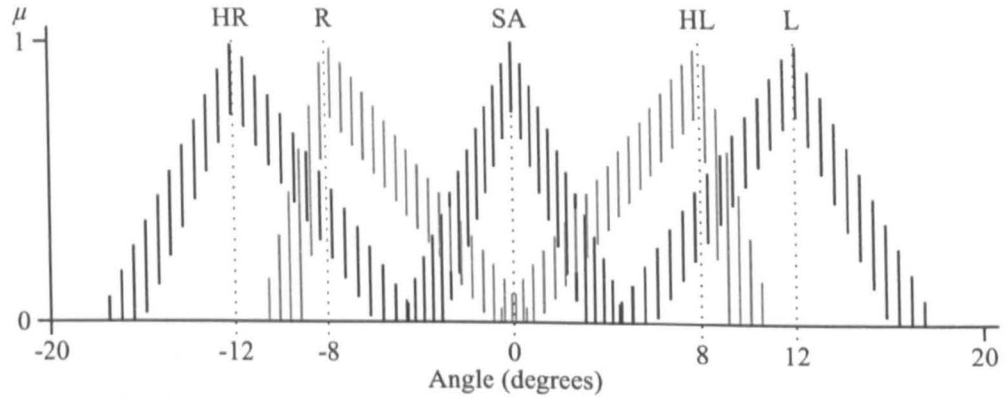


Fig. B.8. The Discrete Type-2 Interval Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

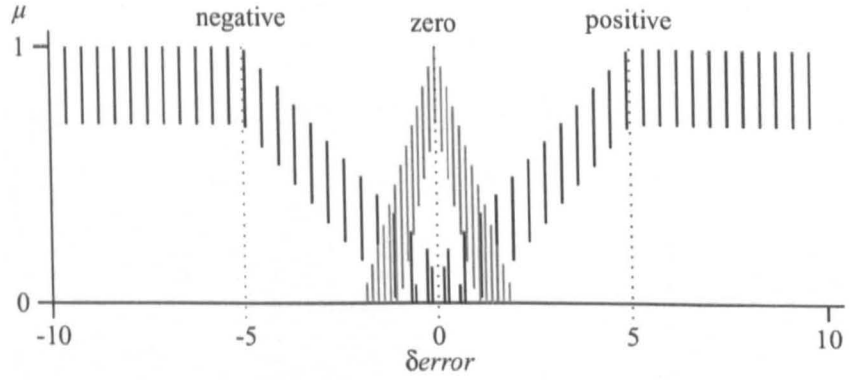


Fig. B.9. The Discrete Type-2 Interval Fuzzy Sets Negative, Zero, Positive

B.4 Type-2 Interval Geometric FLC

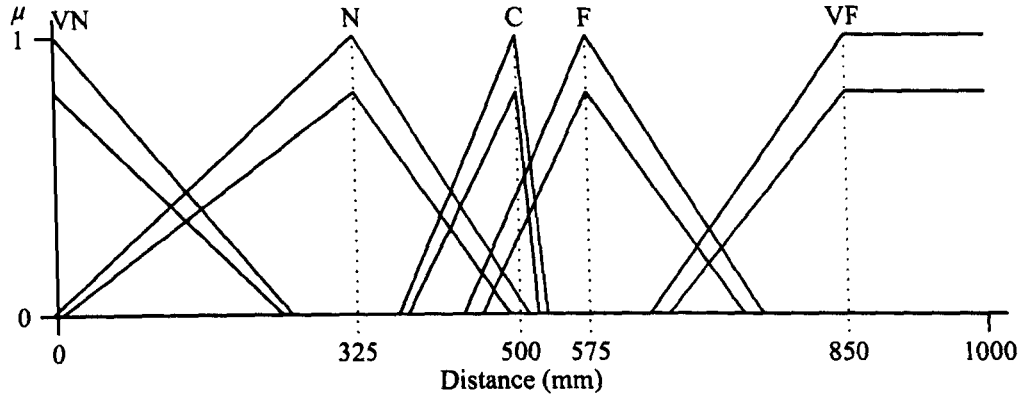


Fig. B.10. The Geometric Type-2 Interval Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

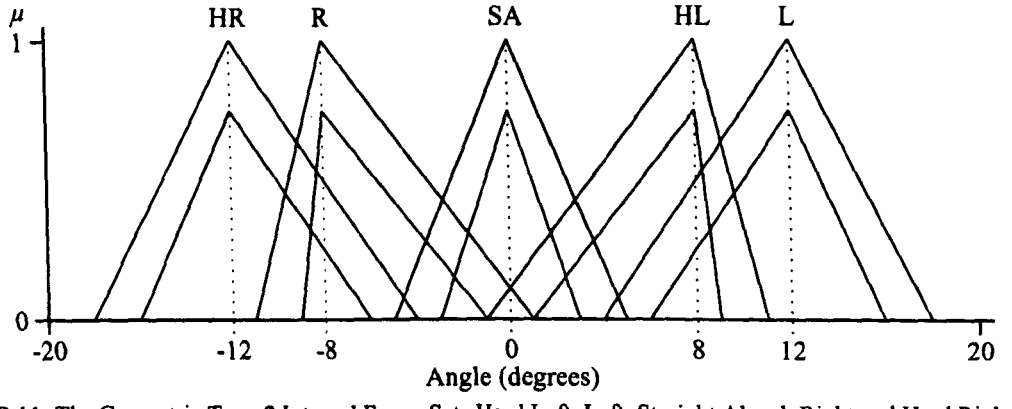


Fig. B.11. The Geometric Type-2 Interval Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

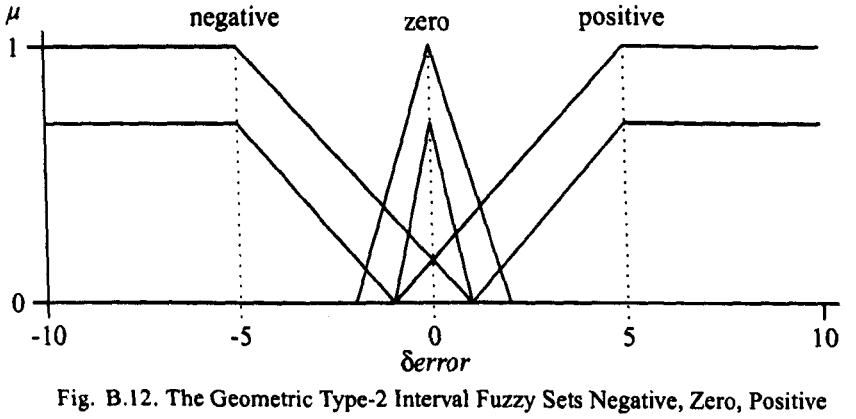


Fig. B.12. The Geometric Type-2 Interval Fuzzy Sets Negative, Zero, Positive

B.5 Type-2 H1 FLC

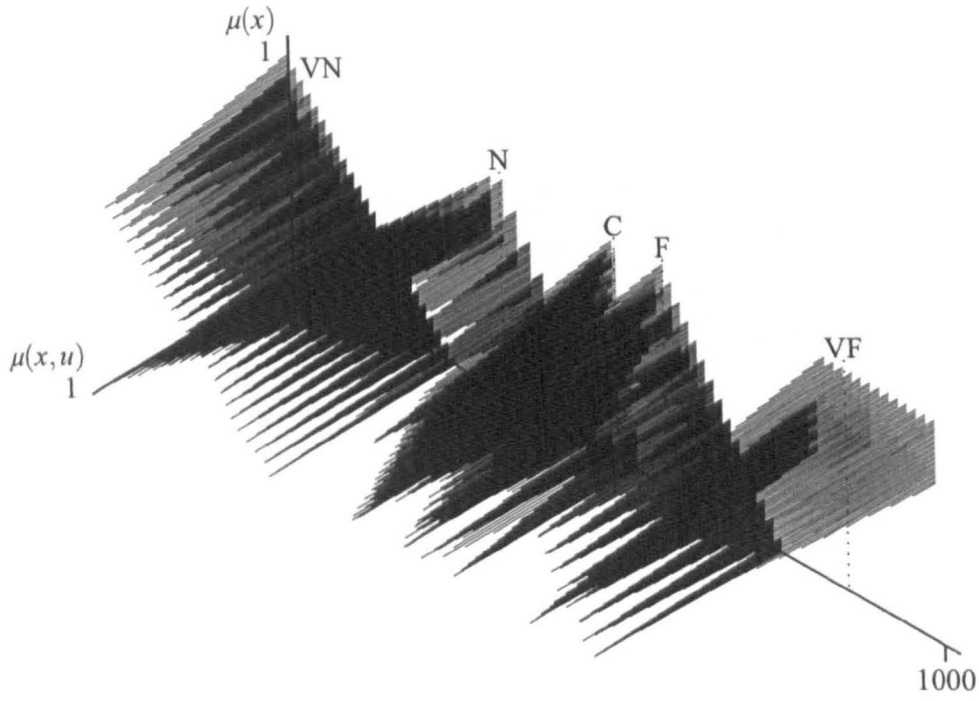


Fig. B.13. The Discrete Type-2 Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

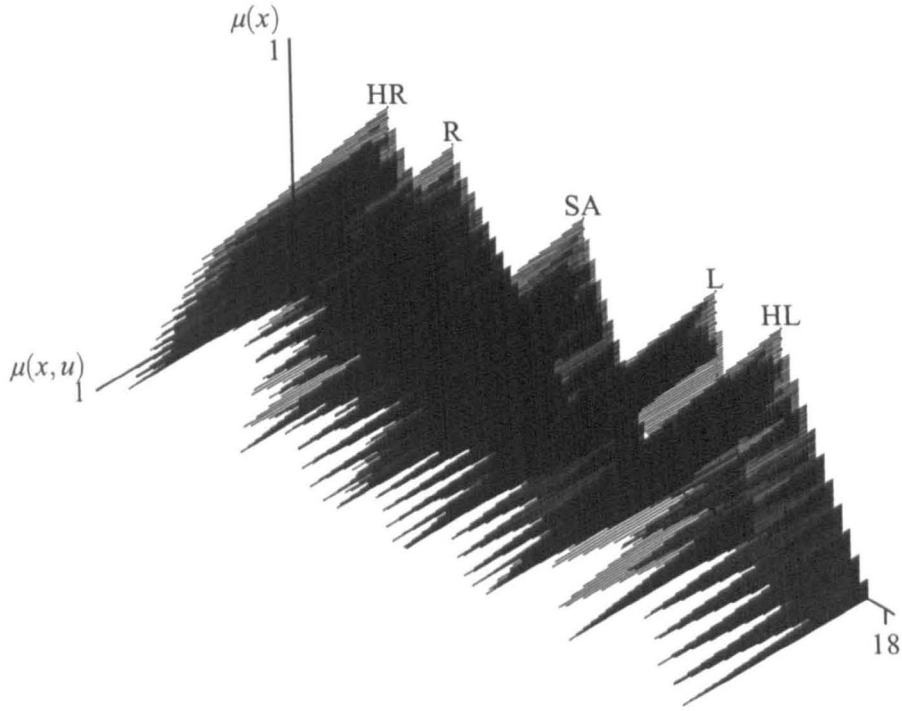


Fig. B.14. The Discrete Type-2 Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

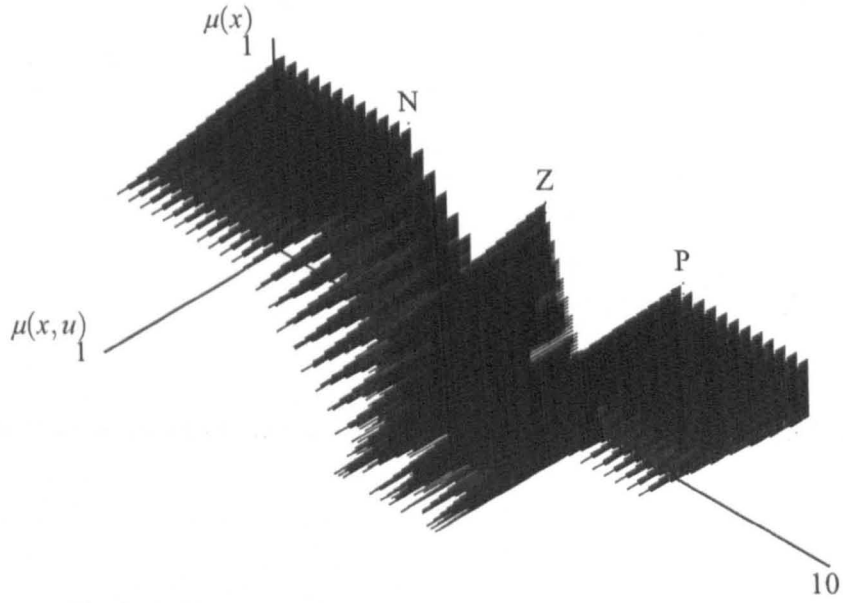


Fig. B.15. The Discrete Type-2 Fuzzy Sets Negative, Zero, Positive

B.6 Type-2 H2 FLC

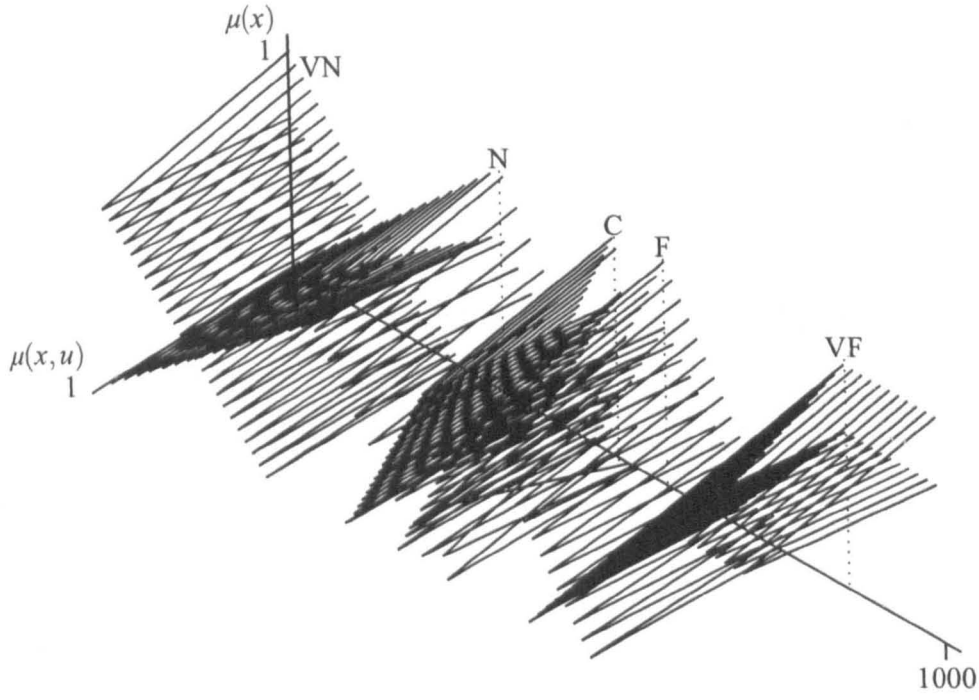


Fig. B.16. The Hybrid Type-2 Fuzzy Sets Very Near, Near, Correct, Far and Very Far.

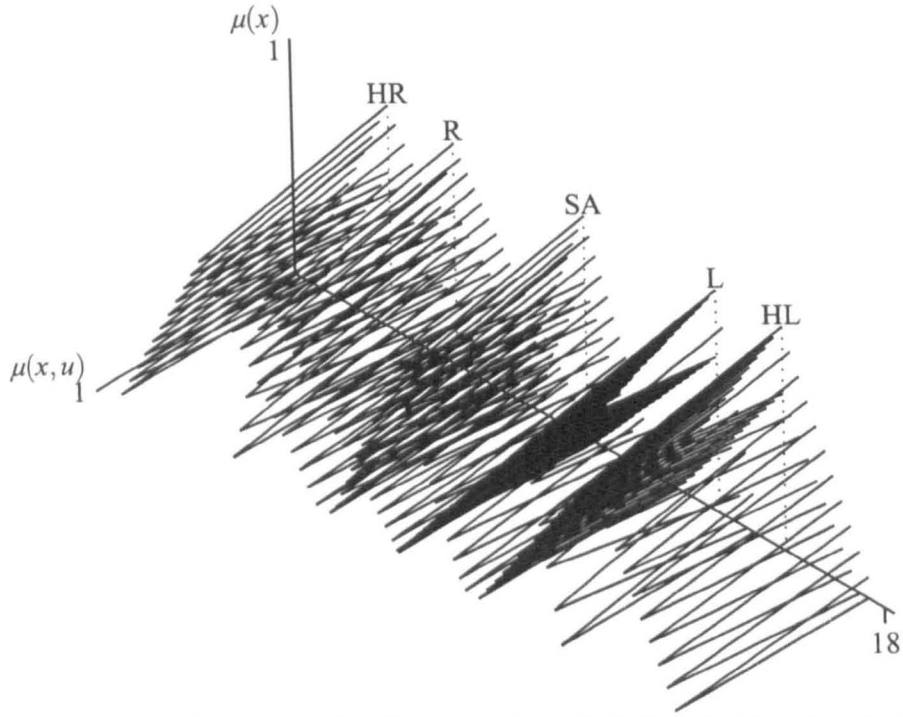


Fig. B.17. The Hybrid Type-2 Fuzzy Sets Hard Left, Left, Straight Ahead, Right and Hard Right.

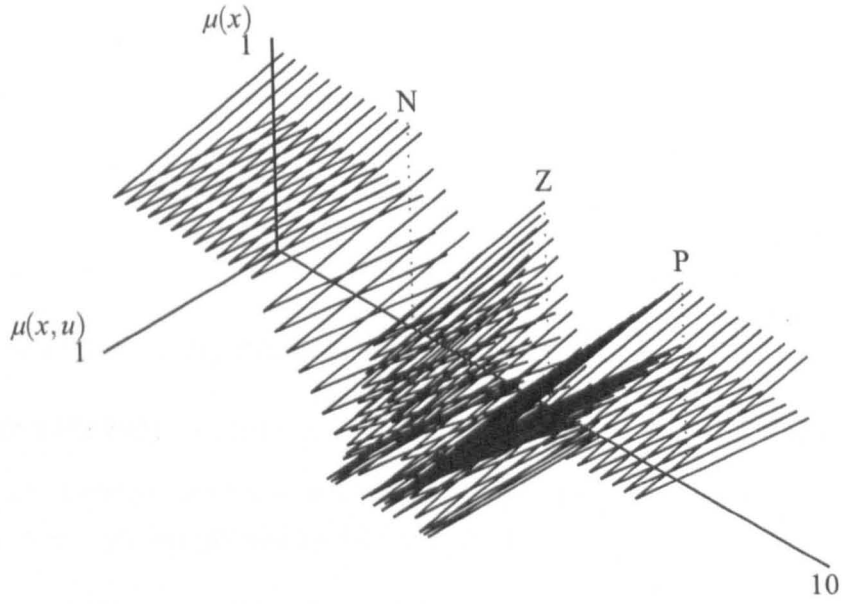


Fig. B.18. The Hybrid Type-2 Fuzzy Sets Negative, Zero, Positive

Appendix C

Software Testing

C.1 Testing Strategy

The black box (Myers79) testing methodology was be applied to each of the FLC used in chapter 5. This was chosen over white box testing as performing white box testing would require specifications of every function in all the code and resources don't allow this.
Each part of the inferencing process in each system will be tested using a set of equivalence classes. For testing purposes each inferencing system will be broken down into the following parts:

1. Building the sets
2. Taking a membership grade
3. AND/OR of that membership grade
4. Implication
5. Combination of implied sets with the OR
6. Defuzzification

C.2 Type-1 Discrete System Testing

C.2.1 Building the Sets

Figure C.1 depicts a comparison between a discrete type-1 fuzzy sets produced by the software and the discrete type-1 set that should have been produced.

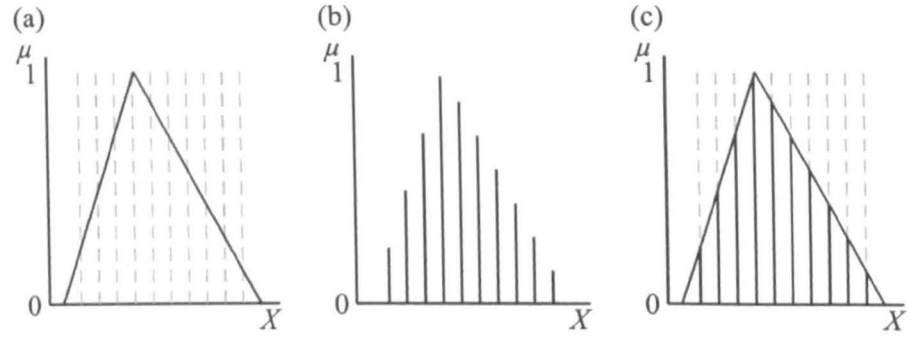


Fig. C.1. Building a Discrete Type-1 Fuzzy Set (a) Theoretical Set (b) Actual Set (c) Comparison

C.2.2 Taking a membership grade

Case	Expected Result	Actual Result
x < s	0	0.000000
x = 0.2(x mod δ X != 0)	0.4375	0.437500
x = 0.2163636(x mod δ X == 0)	0.4886363	0.488636
x > e	0	0.000000

C.2.3 AND/OR of that membership grade

Case	Expected AND	Expected OR	Actual AND	Actual OR
0 - 0.1	0	0.1	0.000000	0.100000
0.1 - 0	0	0.1	0.000000	0.100000
1 - 0.1	0	1	0.000000	1.000000
0.1 - 1	0	1	0.000000	1.000000
0.9 - 0.1	0.1	0.9	0.100000	0.900000
0.1 - 0.9	0.1	0.9	0.100000	0.900000

C.2.4 Implication

Figure C.2 depicts a comparison between an implied discrete type-1 fuzzy sets produced by the software and the implied discrete type-1 set that should have been produced.

C.2.5 Combination of Implied Sets with the OR

Figure C.3 depicts a comparison between a discrete type-1 fuzzy set which results from the combination of two discrete type-1 fuzzy sets produced by the software and the discrete type-1 set that should have been produced.

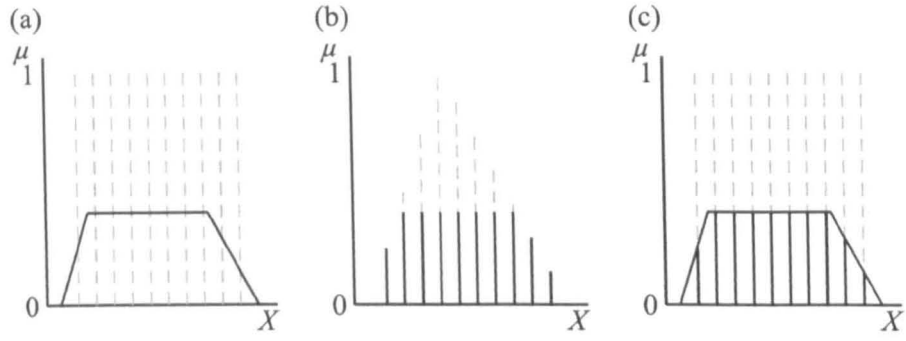


Fig. C.2. Implication on a Discrete Type-1 Fuzzy Set (a) Theoretical Set (b) Actual Set (c) Comparison

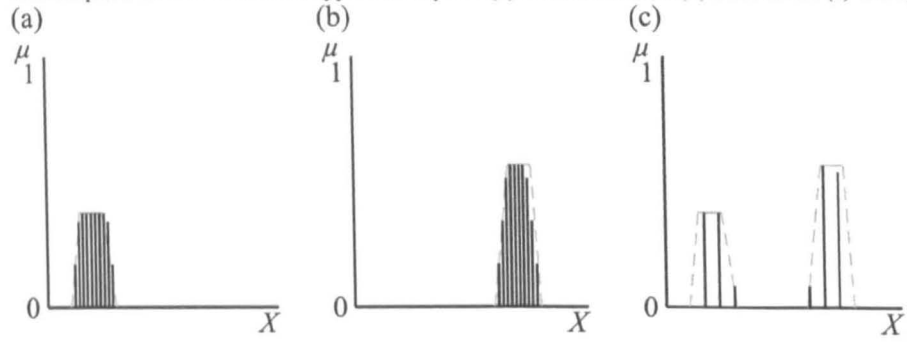


Fig. C.3. Combination of Implied Discrete Type-1 Fuzzy Sets with the OR (a) Set A (b) Set B (c) A or B

C.2.6 Defuzzification

COA of A or B	
Expected answer:	0.570760887857714
Actual answer	0.570761

C.2.7 Result

The type-1 discrete system has been tested and is correct.

C.3 Type-1 Geometric System Testing

C.3.1 Building the sets

Figure C.5 depicts a comparison between a geometric type-1 fuzzy set produced by the software and the geometric type-1 set that should have been produced.

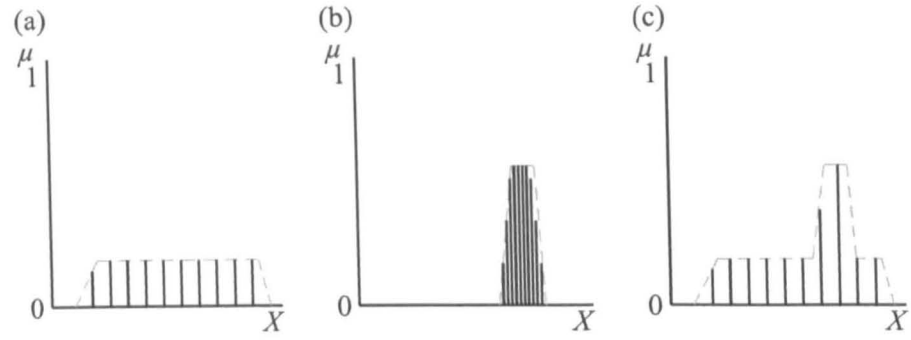


Fig. C.4. Combination of Implied Discrete Type-1 Fuzzy sets with the OR (a) Set A (b) Set B (c) A or B

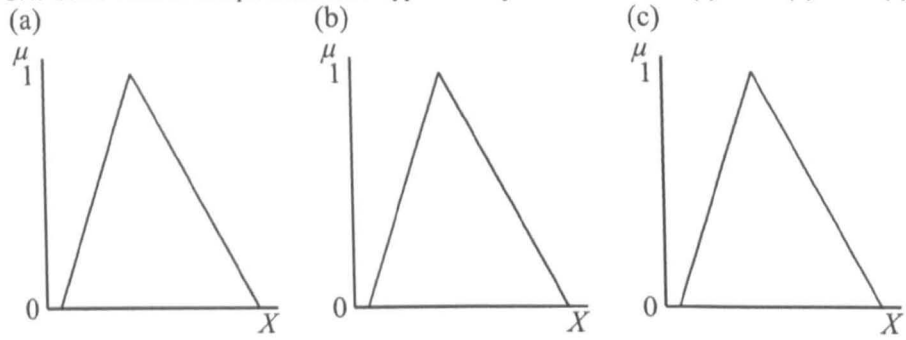


Fig. C.5. Building a Geometric Type-1 Fuzzy set (a) Expected Set (b) Actual Set (c) Comparison

C.3.2 Taking a membership grade

Case	Expected Result	Actual Result
$x < s$	0	0.000000
$x = 0.2$ ($s < x < e$)	0.4375	0.437500
$x > e$	0	0.000000

C.3.3 AND/OR of that membership grade

Case	Expected AND	Expected OR	Actual AND	Actual OR
0 - 0.1	0	0.1	0.000000	0.100000
0.1 - 0	0	0.1	0.000000	0.100000
1 - 0.1	0	1	0.000000	1.000000
0.1 - 1	0	1	0.000000	1.000000
0.9 - 0.1	0.1	0.9	0.100000	0.900000
0.1 - 0.9	0.1	0.9	0.100000	0.900000

C.3.4 Implication

Figure C.6 depicts a comparison between an implied geometric type-1 fuzzy set produced by the software and the implied geometric type-1 set that should have been produced.

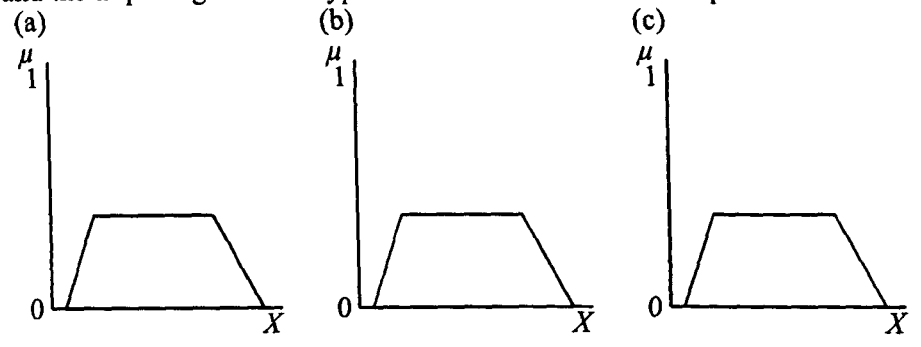


Fig. C.6. Performing Implication on a Geometric Type-1 Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

C.3.5 Combination of implied Sets with the OR

Figures C.7 and C.8 depict the results of two combinations of implied geometric type-1 fuzzy sets.

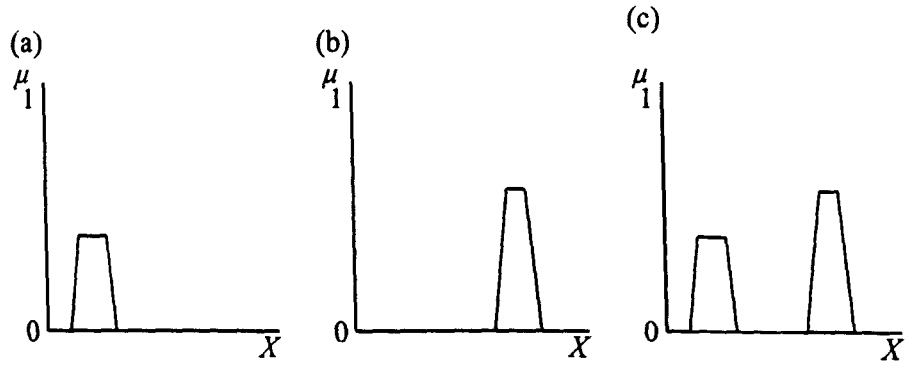


Fig. C.7. Combination of Implied Geometric Type-1 Fuzzy Sets with the OR - Disjoint Sets (a) A (b) B (c) A or B

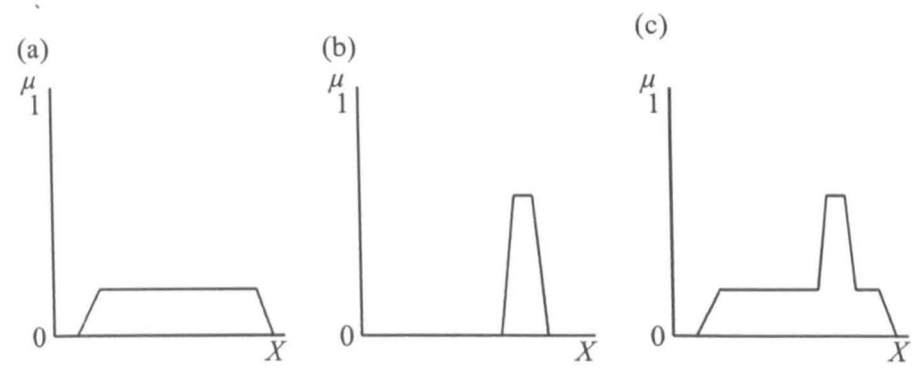


Fig. C.8. Combination of Implied Geometric Type-1 Fuzzy Sets with the OR - Non-Disjoint Sets (a) A (b) B (c) A or B

C.3.6 Defuzzification

COA of A or B	
Expected answer:	0.57228855721393
Actual answer	0.572289

C.3.7 Result

The type-1 geometric fuzzy system has been tested and shown to be correct.

C.4 Type-2 Interval Discrete System Testing

C.4.1 Building the Sets

Figure C.9 depicts a comparison between the discrete type-2 interval fuzzy set produced by the software and the discrete type-2 interval set that should have been produced.

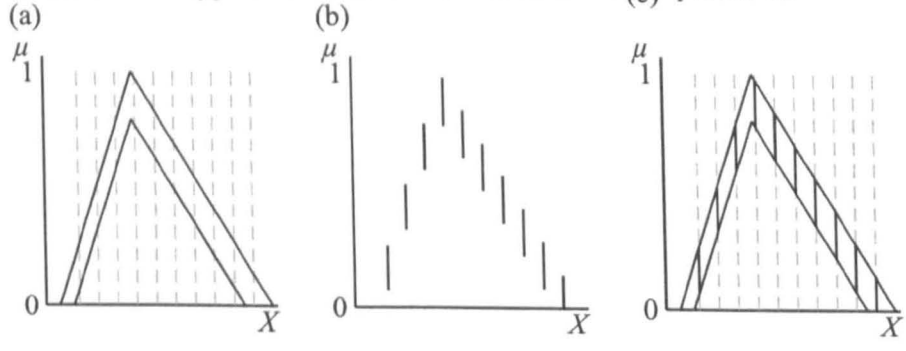


Fig. C.9. Building a Discrete Type-2 Interval Fuzzy Set (a) Theoretical Set (b) Actual Set (c) Comparison

C.4.2 Taking a membership grade

Case	Expected Result	Actual Result
x = 0.0 (x < upper s)	[0,0]	[0.000000,0.000000]
x = 0.1 (upper s < x < lower s)	[0.784091,0.587412]	[0.784091,0.587412]
x = 0.310909 (lower s < x < lower e AND x mod δ X == 0)	[0.$\ddot{9}$66,0.$\ddot{7}$66]	[0.966667, 0.766667]
x = 0.4 (lower s < x < lower e AND x mod δ X != 0)	[0.$\ddot{9}$66,0.$\ddot{7}$66]	[0.966667, 0.766667]
x = 0.9 (lower e < x < upper e)	[1.$\ddot{3}$3,0]	[0.133333,0.000000]
x = 1.0 (x > upper e)	[0,0]	[0.000000,0.000000]

C.4.3 AND/OR of that membership grade

Case	Expected AND	Expected OR	Actual AND	Actual OR
[0.2,0.3] - [0.4,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.2,0.4] - [0.3,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.2,0.5] - [0.3,0.4]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.3,0.4] - [0.2,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.3,0.5] - [0.2,0.4]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]

C.4.4 Implication

Figure C.10 depicts a comparison between the implied discrete type-2 interval fuzzy set produced by the software and the implied discrete type-2 interval set that should have been produced.

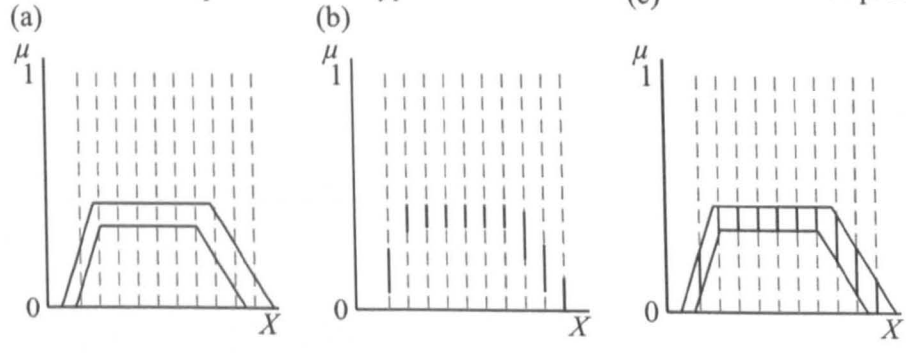


Fig. C.10. Performing Implication on a Discrete Type-2 Interval Fuzzy Set (a) Theoretical Set (b) Actual Set (c) Comparison

C.4.5 Combination of implied Sets with the OR

Figures C.11 and C.12 depict the results of two combinations of implied discrete type-2 interval fuzzy sets.

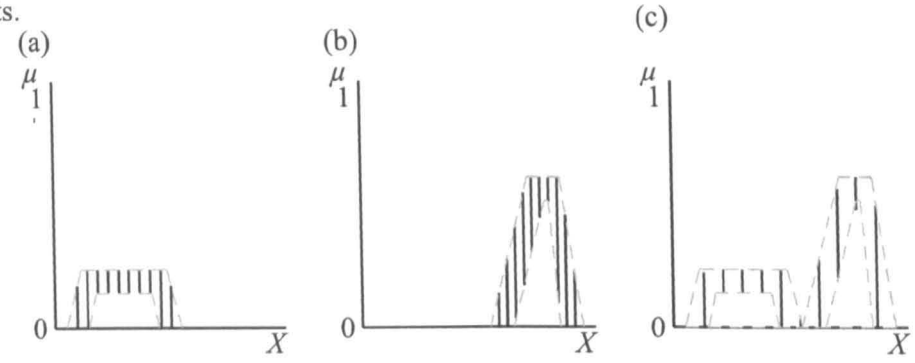


Fig. C.11. Combination of Implied Discrete Type-2 Interval Fuzzy Sets with the OR - Disjoint Sets (a) Theoretical Set (b) Actual Set (c) Comparison

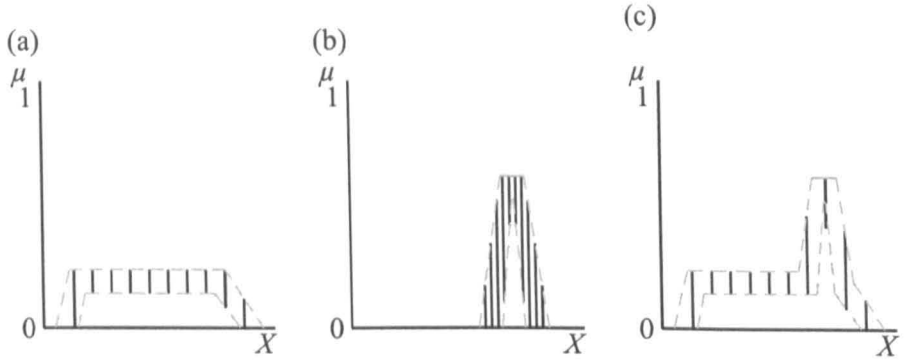


Fig. C.12. Combination of Implied Discrete Type-1 Fuzzy Sets with the OR - Non-Disjoint Sets (a) Theoretical Set (b) Actual Set (c) Comparison

C.4.6 Defuzzification

COA of A or B	
Expected Type-Reduced Set	[0.442761532606134,0.610163833080078]
Actual Type-Reduced Set	[0.442762,0.610164]
Expected Answer	0.526462682843106
Actual Answer	0.526463

C.4.7 Result

The discrete type-2 interval system has been tested and is correct.

C.5 Type-2 Interval Geometric System Testing

C.5.1 Building the Sets

Figure C.13 depicts a comparison between the geometric type-2 interval fuzzy set produced by the software and the geometric type-2 interval set that should have been produced.

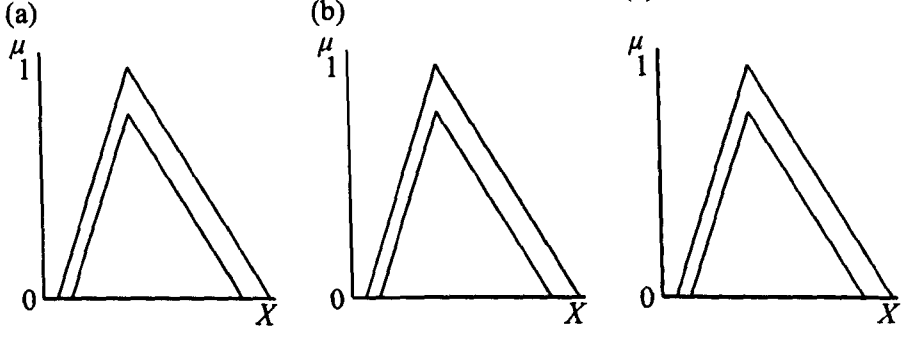


Fig. C.13. Building a Geometric Type-2 Interval Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

C.5.2 Taking a membership grade

Case	Expected Result	Actual Result
x = 0.0 (x < upper s)	[0,0]	[0.000000,0.000000]
x = 0.1 (upper s < x < lower s)	[0.125,0]	[0.125000,0.000000]
x = 0.4 (lower s < x < lower e)	[0.$\ddot{9}$66,0.$\ddot{7}$66]	[0.966667, 0.766667]
x = 0.9 (lower e < x < upper e)	[1.$\ddot{3}$3,0]	[0.133333,0.000000]
x = 1.0 (x > upper e)	[0,0]	[0.000000,0.000000]

C.5.3 AND/OR of that membership grade

Case	Expected AND	Expected OR	Actual AND	Actual OR
[0.2,0.3] - [0.4,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.2,0.4] - [0.3,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.2,0.5] - [0.3,0.4]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.3,0.4] - [0.2,0.5]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]
[0.3,0.5] - [0.2,0.4]	[0.2,0.3]	[0.4,0.5]	[0.200000,0.300000]	[0.400000,0.500000]

C.5.4 Implication

Figure C.14 depicts a comparison between the implied geometric type-2 interval fuzzy set produced by the software and the implied geometric type-2 interval set that should have been produced.

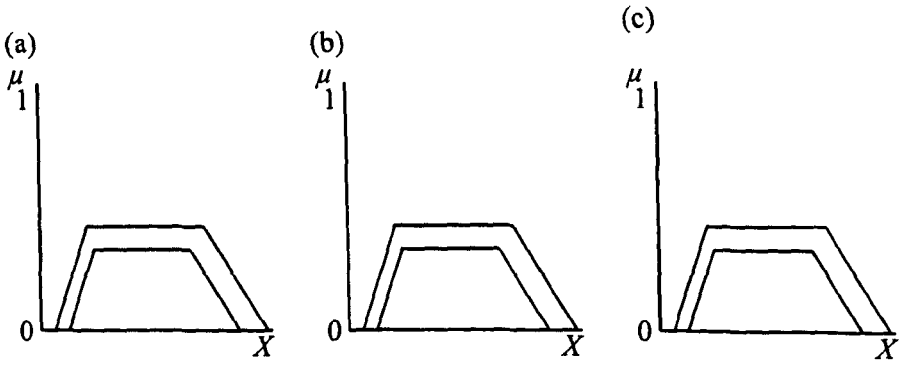


Fig. C.14. Performing Implication on a Geometric Type-2 Interval Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

C.5.5 Combination of Implied Sets with the OR

Figures C.15 and C.16 depict a comparison between a discrete type-1 fuzzy set which results from the combination of two discrete type-1 fuzzy sets produced by the software and the discrete type-1 set that should have been produced.

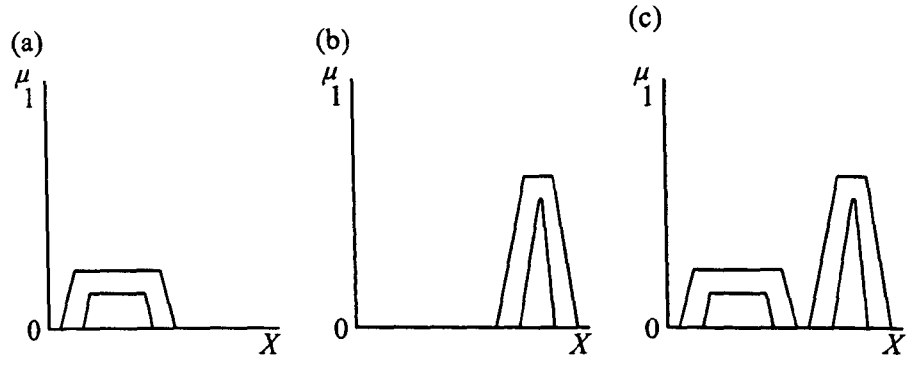


Fig. C.15. Combination of Implied Geometric Type-2 Interval Fuzzy Sets with the OR - Disjoint Sets (a) Expected Set (b) Actual Set (c) Comparison

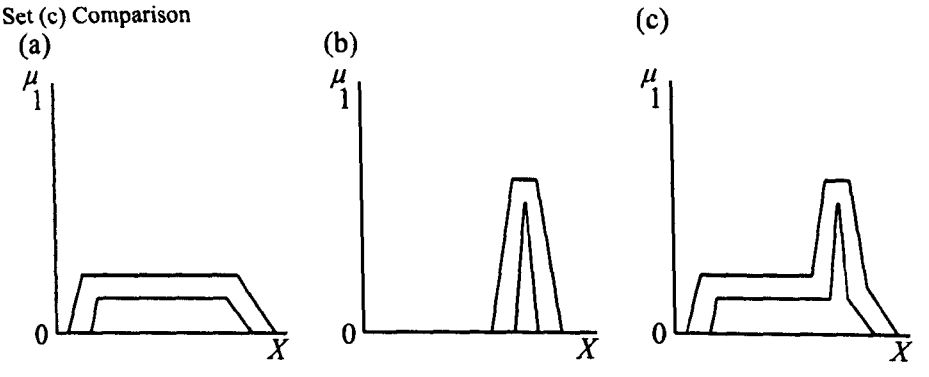


Fig. C.16. Combination of Implied Geometric Type-2 Interval Fuzzy Sets with the OR - Non-Disjoint Sets (a) Expected Set (b) Actual Set (c) Comparison

C.5.6 Defuzzification

COA of A or B	
Expected Answer	0.547204461448106
Actual Answer	0.547205

C.5.7 Result

The geometric type-2 interval system has been test and is correct.

C.6 Type-2 Discrete System Testing

C.6.1 Building the Sets

Figure C.17 depicts a comparison between the discrete type-2 fuzzy set produced by the software and the discrete type-2 set that should have been produced.

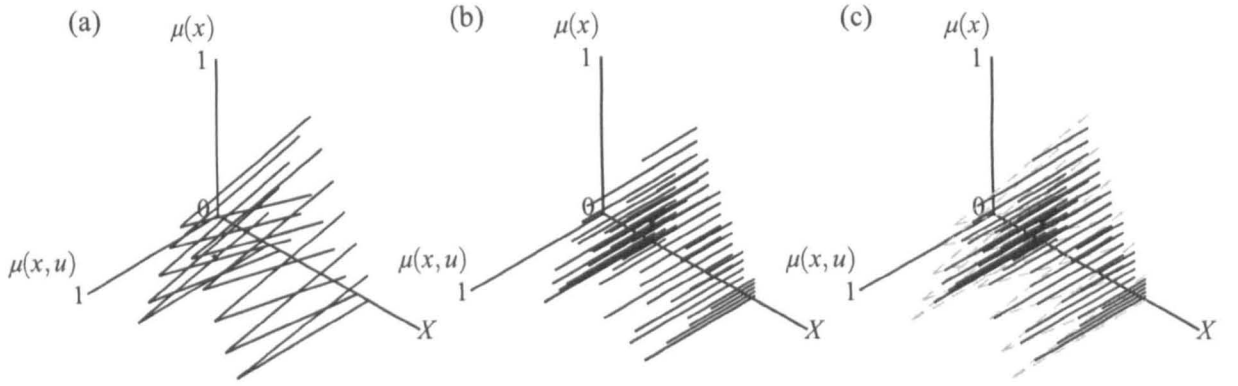


Fig. C.17. Building a Discrete Type-2 Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

C.6.2 Taking a membership grade

Case	Expected Result	Actual Result
x < s	Null	Null
x = 0.5(x mod δ X != 0)	Figure C.19 (a)	Figure C.19 (b)
x = 0.545455(x mod δ X == 0)	Figure C.18 (a)	Figure C.18 (b)
x > e	Null	Null

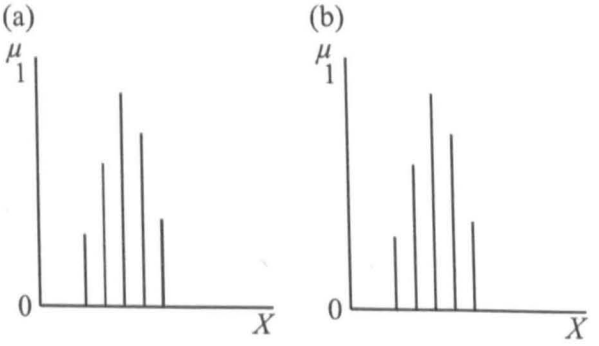


Fig. C.18. Taking a Membership Grade of A Discrete Type-2 Geometric Fuzzy Set I (a) x = 0.545455 (x mod *delta* X == 0) Expect result (b) x = 0.545455 (x mod *delta* X == 0) Actual result

C.6.3 AND/OR of that membership grade

Figures C.20 and C.21 depict a comparison between the secondary membership functions that should have been produced by the system and the actual functions given by the system.

C.6.4 Implication

Figure C.22 depicts a comparison between the implied discrete type-2 fuzzy set produced by the software and the implied discrete type-2 set that should have been produced.

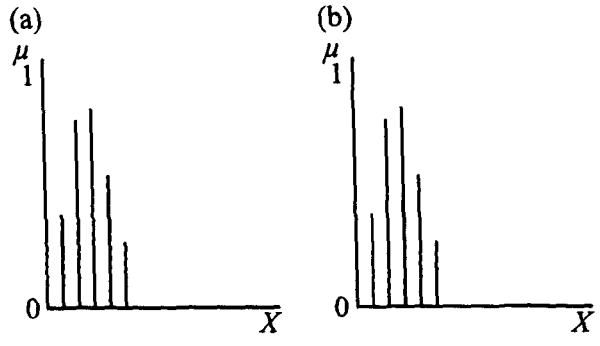


Fig. C.19. Taking a Membership Grade of A Discrete Type-2 Geometric Fuzzy Set II (a) x = 0.5 (x mod *delta* X != 0) Expected result (b) x = 0.5 (x mod *delta* X != 0) Actual result

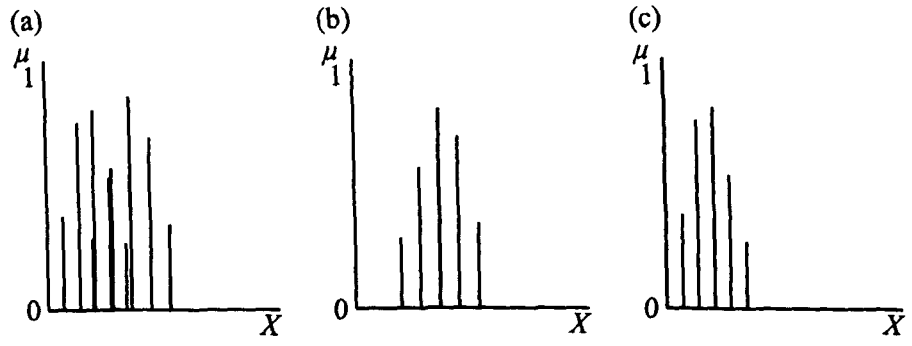


Fig. C.20. The Join and Meet Operations of a Discrete Type-2 Fuzzy Set I (a) Sets (b) Join (c) Meet

C.6.5 Combination of implied Sets with the OR

Figures C.23 and C.24 depict a comparison between a discrete type-2 fuzzy set which results from the combination of two discrete type-2 fuzzy sets produced by the software and the discrete type-2 set that should have been produced.

C.6.6 Defuzzification

Figure C.24 depict a comparison between the first 100 embedded sets that should have been enumerated from a discrete type-2 set and the actual embedded sets given by the system.

C.6.7 Result

The discrete type-2 system has been tested and is correct.

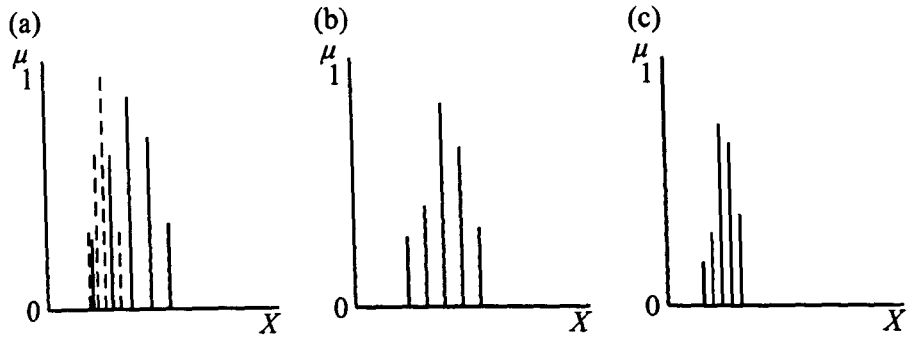


Fig. C.21. The Join and Meet Operations of a Discrete Type-2 Fuzzy Set II (a) Sets (b) Join (c) Meet

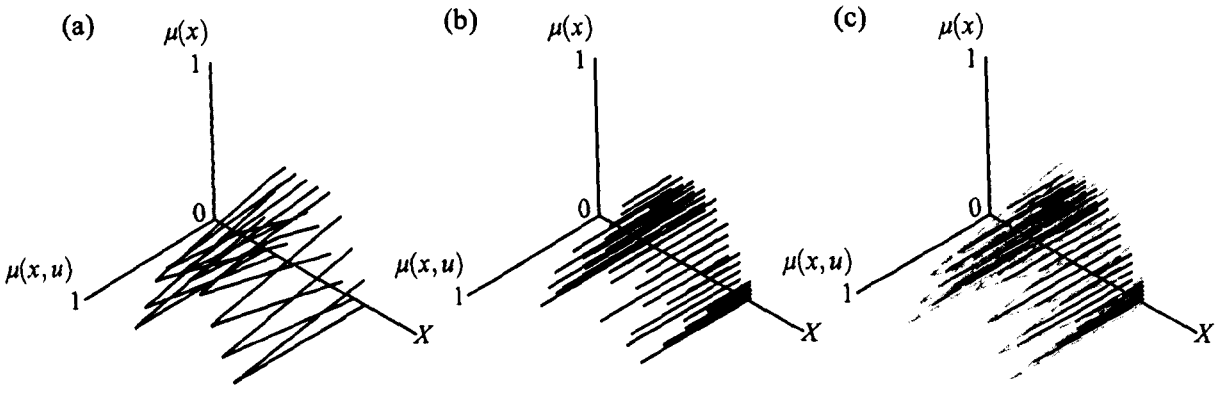


Fig. C.22. Performing Implication on a Discrete Type-2 Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

C.7 Type-2 Hybrid 1 System Testing

Figure C.25 depicts a comparison between the discrete type-2 fuzzy set produced by the software and the discrete type-2 set that should have been produced.

C.7.1 Building the Sets

C.7.2 Taking a membership grade

Case	Expected Result	Actual Result
x < s	Null	Null
x = 0.5(x mod δ X != 0)	Figure C.27 (a)	Figure C.27 (b)
x = 0.545455(x mod δ X == 0)	Figure C.26 (a)	Figure C.26 (b)
x > e	Null	Null

C.7.3 AND/OR of that membership grade

Figures C.28 and C.29 depict a comparison between the secondary membership functions that should have been produced by the system and the actual functions given by the system.

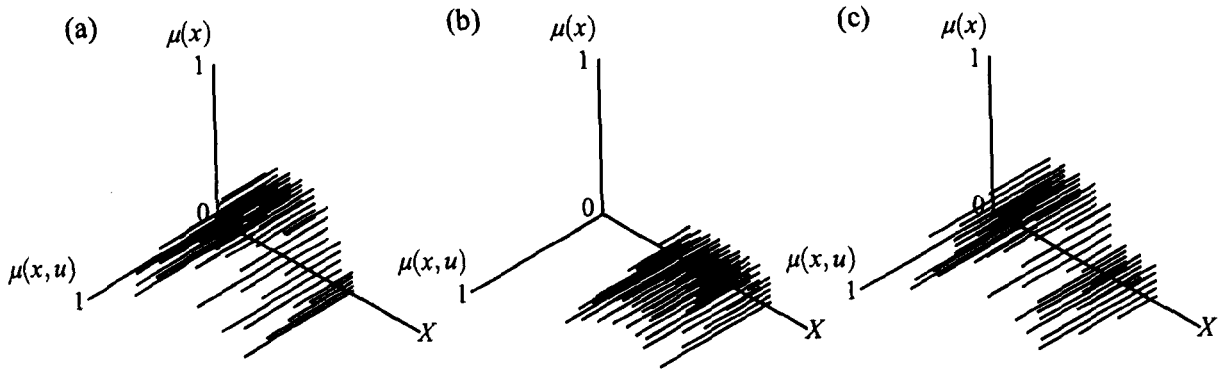


Fig. C.23. Combination of Implied Discrete Type-2 Interval Fuzzy Sets with the OR (a) Set A (b) Set B (c) A or B

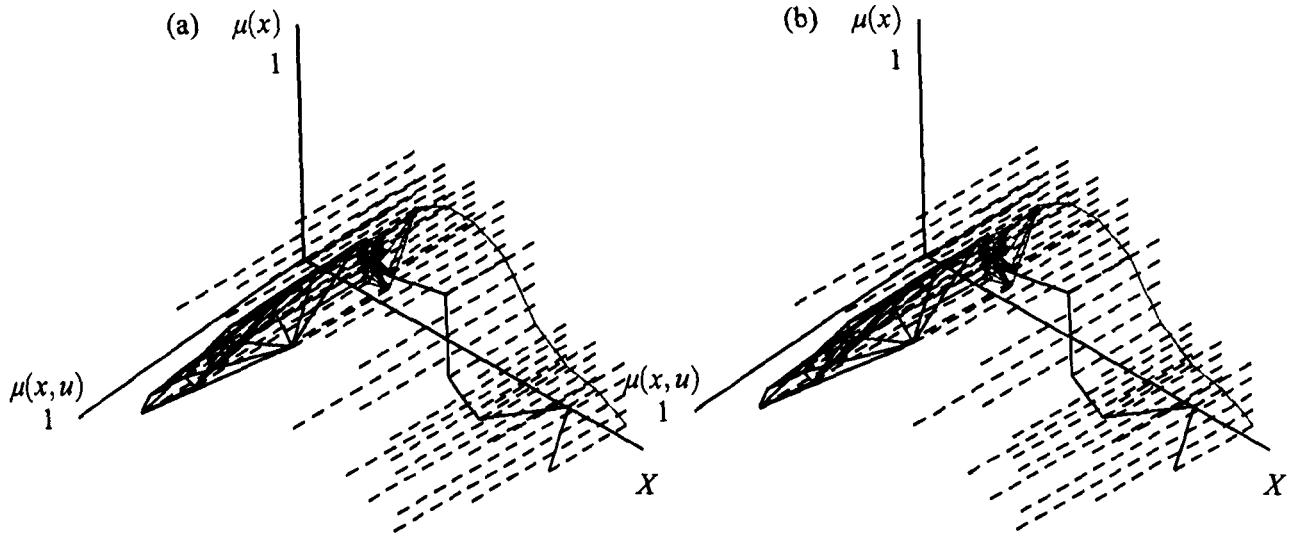


Fig. C.24. Enumeration of the First One Hundred Embedded Sets(a) Expected Embedded Sets (b) Actual Embedded Sets

C.7.4 Implication

Figure C.30 depicts a comparison between the implied discrete type-2 fuzzy set produced by the software and the implied discrete type-2 set that should have been produced.

C.7.5 Combination of implied Sets with the OR

Figures C.31 and C.32 depict a comparison between a discrete type-2 fuzzy set which results from the combination of two discrete type-2 fuzzy sets produced by the software and the discrete type-2 set that should have been produced.

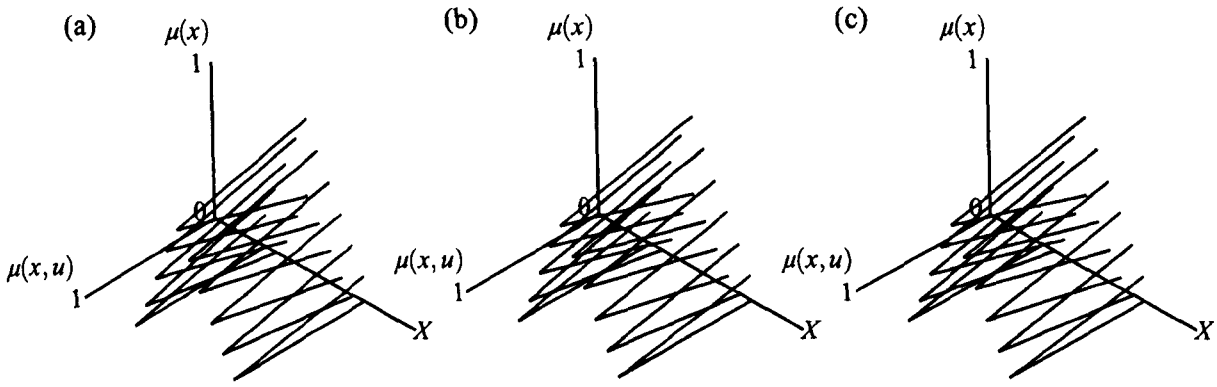


Fig. C.25. Building a Hybrid Type-2 Fuzzy Set (a) Expected Set (b) Actual Set (c) Comparison

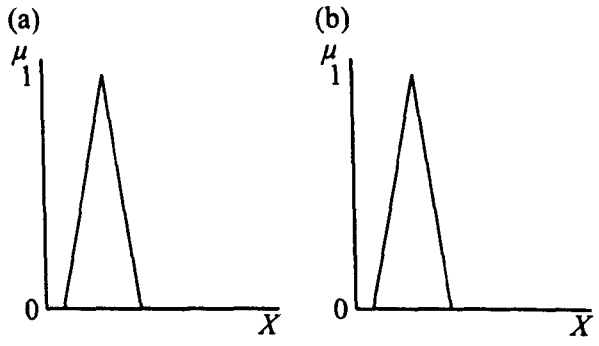


Fig. C.26. Taking a Membership Grade of A Hybrid Type-2 Fuzzy Set I (a) x = 0.545455 (x mod *delta* X == 0) Expect result (b) x = 0.545455 (x mod *delta* X == 0) Actual result

C.7.6 Defuzzification

Figure C.32 depicts a comparison between the hybrid to geometric transform that should have been produced by the system the the transform that actually was produced.

C.7.7 Result

The hybrid type-2 system has been tested and is correct.

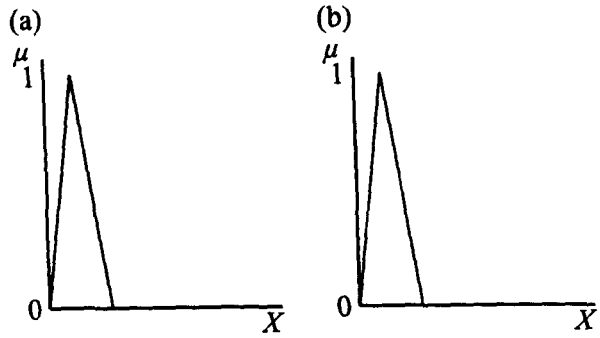


Fig. C.27. Taking a Membership Grade of A Hybrid Type-2 Fuzzy Set II (a) x = 0.5 (x mod *delta* X != 0) Expected result (b) x = 0.5 (x mod *delta* X != 0) Actual result

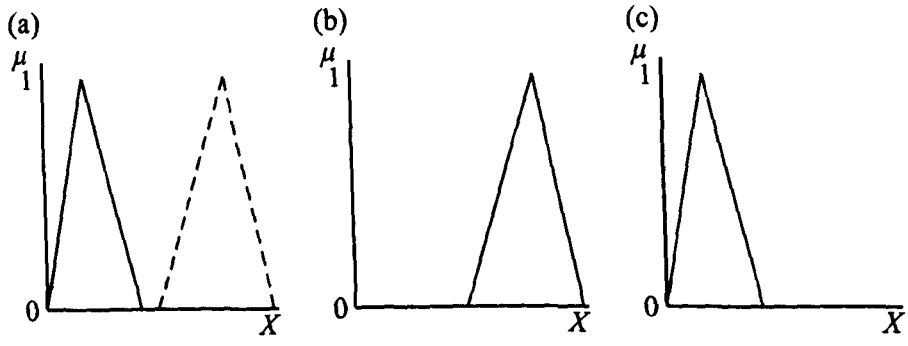


Fig. C.28. The Join and Meet Operations of a Hybrid Type-2 Fuzzy Set I (a) Sets (b) join (c) meet

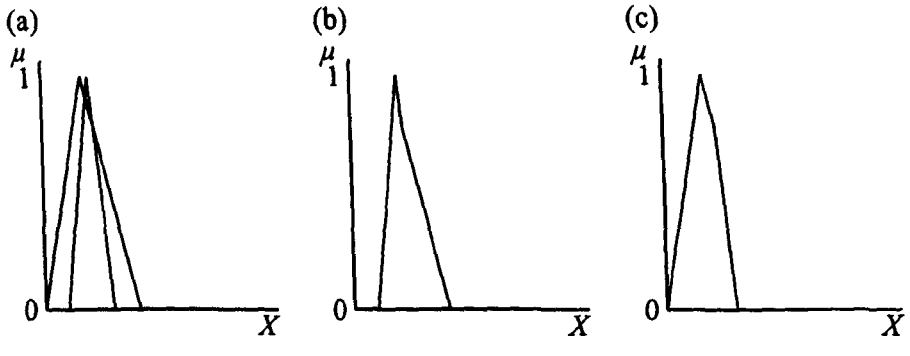


Fig. C.29. The Join and Meet Operations of a Hybrid Type-2 Fuzzy Set II (a) Sets (b) join (c) meet

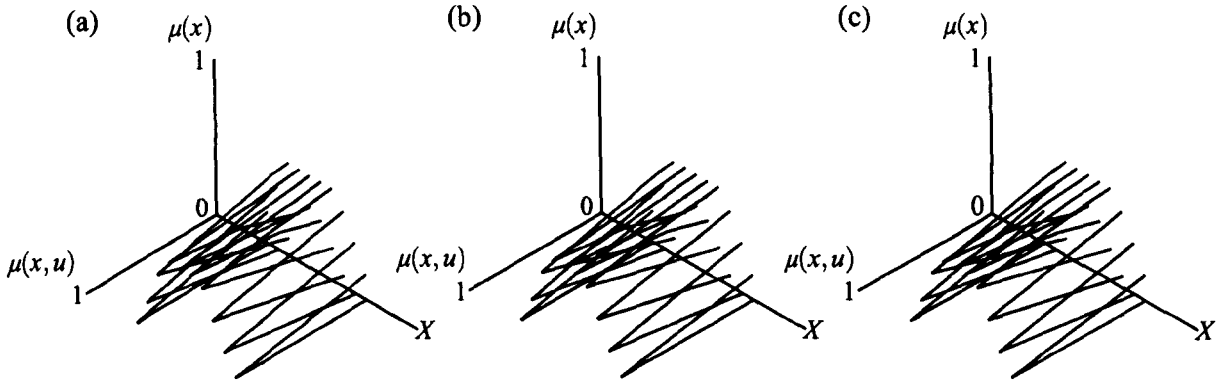


Fig. C.30. Performing Implication on a Hybrid Type-2 Fuzzy Set(a) Expected Set (b) Actual Set (c) Comparison

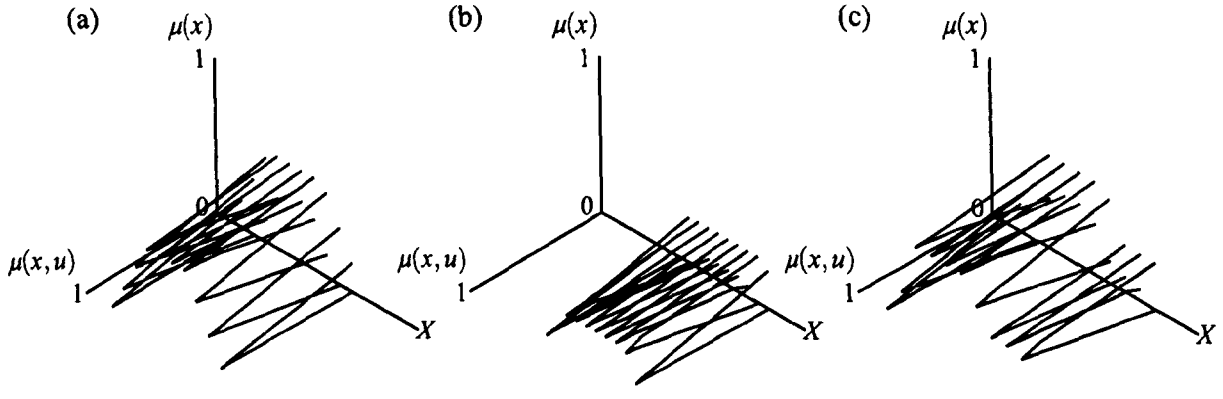


Fig. C.31. Combination of Implied Hybrid Type-2 Fuzzy Sets with the OR (a) A (b) B (c) A or B

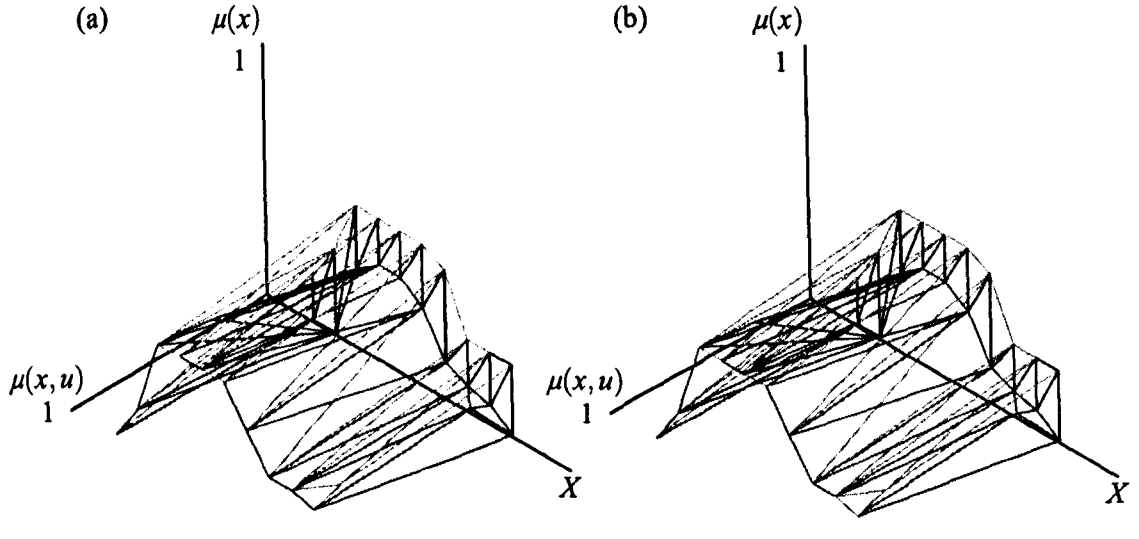


Fig. C.32. Hybrid to Geometric Transformation (a) Expected (b) Actual

Appendix D

Paths Taken by the Robot

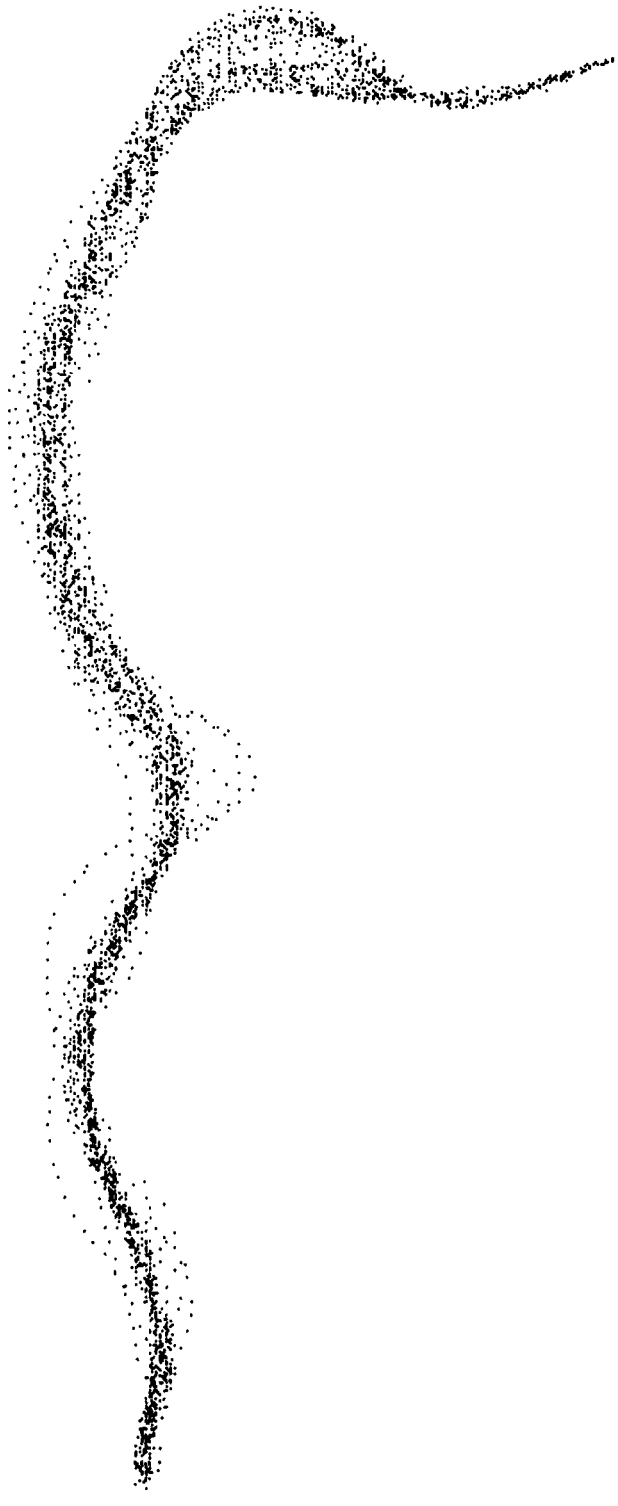


Fig. D.1. The Raw Tracked Paths of the Discrete Type-1 Robot FLC.

Fig. D.2. The Raw Tracked Paths of the Geometric Type-1 Robot FLC.

Fig. D.3. The Raw Tracked Paths of the Discrete Type-2 Interval Robot FLC.

Fig. D.4. The Raw Tracked Paths of the Geometric Type-2 Robot FLC.

Fig. D.5. The Raw Tracked Paths of the Type-2 H1 Robot FLC.

Fig. D.6. The Raw Tracked Paths of the Type-2 H2 Robot FLC.

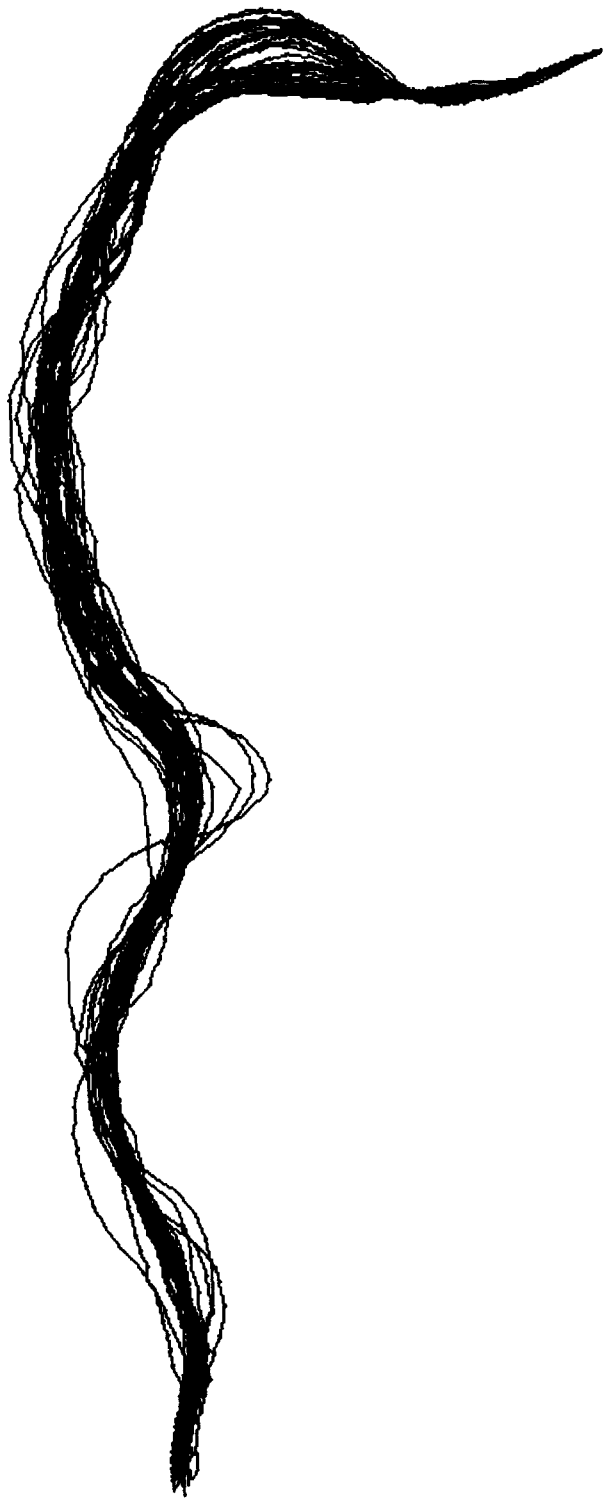


Fig. D.7. The Tracked Paths of the Discrete Type-1 Robot FLC.

Fig. D.8. The Tracked Paths of the Geometric Type-1 Robot FLC.

Fig. D.9. The Tracked Paths of the Discrete Type-2 Interval Robot FLC.

Fig. D.10. The Tracked Paths of the Geometric Type-2 Robot FLC.

Fig. D.11. The Tracked Paths of the Type-2 H1 Robot FLC.

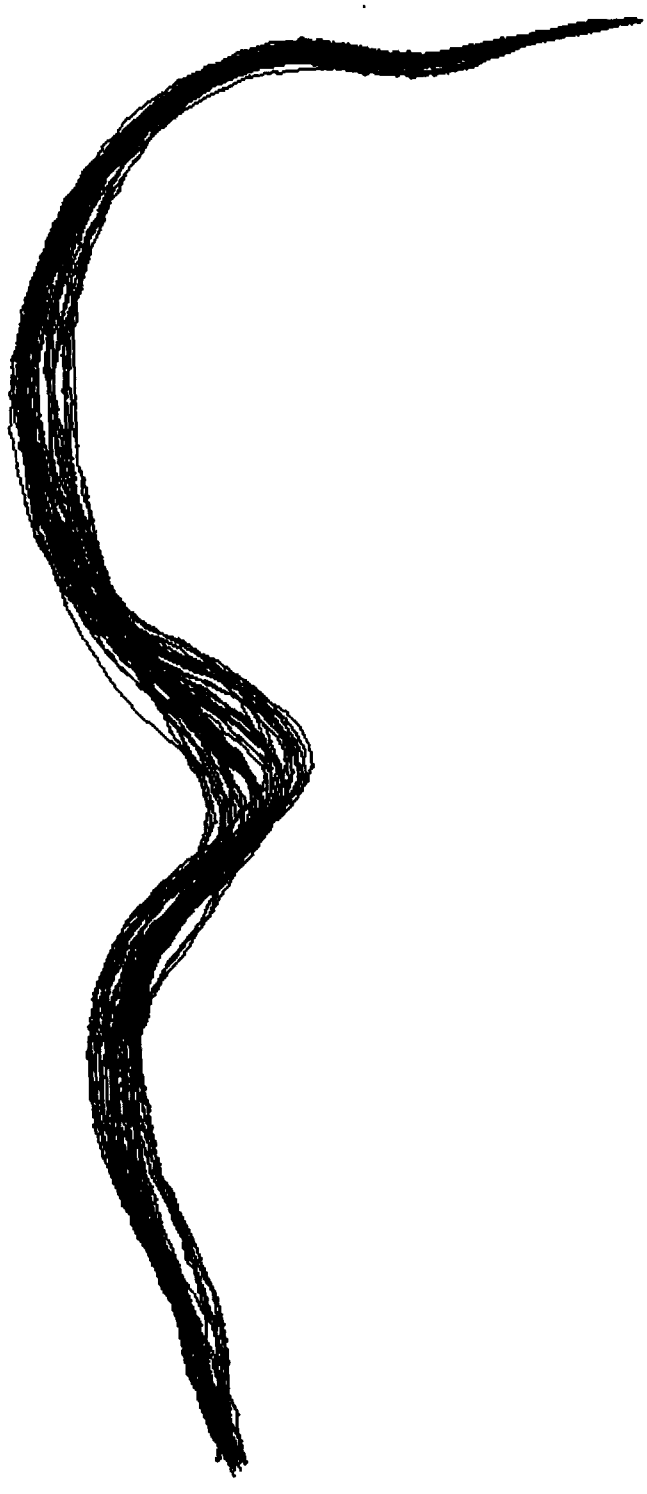


Fig. D.12. The Tracked Paths of the Type-2 H2 Robot FLC.

Appendix E

Copies of Publications by Simon C. Coupland that are Directly Related to this Thesis

Coupland, S. & John, R. (2003), An Approach to Type-2 Fuzzy Arithmetic, *in* 'Proc. UKCI 2003', pp. 107 – 114.

Coupland, S. & John, R. (2004), A New and Efficient Method for the Type-2 Meet Operation, *in* 'Proc. FUZZ-IEEE 2004', Budapest, Hungary, pp. 959 – 964.

Coupland, S. & John, R. (2004), Fuzzy Logic and Computational Geometry, *in* 'Proc. RASC 2004', Nottingham, England, pp. 3 – 8.

Coupland, S. & John, R. (2005), Towards More Efficient Type-2 Fuzzy Logic Systems, *in* 'Proc. FUZZ-IEEE 2005', Reno, NV, USA, pp. 236 – 241.

Coupland, S. & John, R. (2005), Geometric Interval Type-2 Fuzzy Systems, *in* 'Proc. EUSFLAT 2005', Barcelona, Spain, pp. 449 – 454.

Greenfield, S., John, R. & Coupland, S. (2005), A Novel Sampling Method for Type-2 Defuzzification, *in* 'Proc. UKCI 2005', pp. 120 – 127.

Coupland, S. & John, R. (2006), An Investigation into Alternative Methods for the Defuzzification of an Interval Type-2 Fuzzy Set, *in* 'Proc. FUZZ-IEEE 2006', Vancouver, Canada. Accepted for publication.

Coupland, S., Gongora, M., John, R. & Wills, K. (2006), A Comparative Study of Fuzzy Logic Controllers for Autonomous Robots, *in* 'Proc. IPMU 2006', Paris, France. Accepted for publication.

Coupland, S. & John, R. (2006), 'Geometric Type-1 and Type-2 Fuzzy Logic Systems', *IEEE Transactions on Fuzzy Systems* . Accepted for publication.