Context dependent fuzzy modelling and its applications

Duc Thang Ho, BSc

Thesis submitted to The University of Nottingham

for the degree of Doctor of Philosophy

August 2013

Abstract

Fuzzy rule-based systems (FRBS) use the principle of fuzzy sets and fuzzy logic to describe vague and imprecise statements and provide a facility to express the behaviours of the system with a human-understandable language. Fuzzy information, once defined by a fuzzy system, is fixed regardless of the circumstances and therefore makes it very difficult to capture the effect of context on the meaning of the fuzzy terms. While efforts have been made to integrate contextual information into the representation of fuzzy sets, it remains the case that often the context model is very restrictive and/or problem specific. The work reported in this thesis is our attempt to create a practical frame work to integrate contextual information into the representation of fuzzy sets so as to improve the interpretability as well as the accuracy of the fuzzy system. Throughout this thesis, we have looked at the capability of the proposed context dependent fuzzy sets as a stand alone as well as in combination with other methods in various application scenarios ranging from time series forecasting to complicated car racing control systems. In all of the applications, the highly competitive performance nature of our approach has proven its effectiveness and efficiency compared with existing techniques in the literature.

Acknowledgments

I would like to thank many amazing people who helped me over the course of my PhD. Without their help and support, it would have been much more difficult, if not impossible, for me to complete this thesis.

First of all, I would like to thank my supervisor Dr. Jon M. Garibaldi for his guidance, patience and advice throughout the whole course of my research.

I would like to thank my father, my mother and my sister for their support and encouragement through the many years since I was a child.

Finally this thesis is for my wife, Kim Nga and my young sister, Huong Ho who were always there to support me unconditionally.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivations

Fuzzy rule based systems (FRBSs) have been extensively and successfully applied to both expert systems and various engineering fields such as control, classification, and regression in a significant number of applications [102, 104]. The key for their success is the ability to describe human knowledge using vague and imprecise statements, and at the same time to express the system behaviours in a descriptive language easily interpretable by human beings. A knowledge base – the main component of an FRBS that stores knowledge about the problem being solved – is composed of:

- a rule base (RB), constituted of a collection of linguistic IF-THEN rules, such as *IF the temperature is hot, THEN switch on the fan*, and
- a data base (DB), which associates semantics, represented by a fuzzy set and its membership functions, with each of the linguistic terms used in the RB, e.g., *hot, cold, fast*.

Traditionally, a knowledge base is built by experts with a thorough understanding of the system's nature and behaviour. Such a system is called a fuzzy expert system and the modelling process is called white box modelling. However, in practice, it is not always possible to achieve complete understanding of the underlying mechanisms of the system due to a number of reasons including; poor understanding of the under-

lying phenomena, inaccurate measurements, or the unpredictable nature of the model itself. In some applications, prior knowledge of the system may be completely obscure, for example, in time series predictions or function approximations. In such instance, deriving IF-THEN rules manually is not only a major effort but also results in poor performance in comparison with other methods. To overcome this limitation, black box modelling methods are usually employed to automatically derive a knowledge base either by means of a learning process, which generates a rule base and sometimes a data base as well, or by a tuning process, which only defines a data base. The process of learning FRBS is often referred to as fuzzy modelling. It must be noted, however, that the combination of fuzzy logic and black box methods usually results in an FRBS with low interpretability. This is often due to the the learning or tuning procedures not taking the semantic properties of the underlying fuzzy system into account. Thus, fuzzy modelling is usually useful in applications that favour accuracy such as function approximation, classification, time series prediction and control systems.

Notwithstanding their many advantages, FRBSs, whether created manually by experts or automatically by learning methods, can suffer from a major drawback. It is the lack of flexibility in the representation of fuzzy sets due to the implicit assumption that the context of the system is fixed while in fact, in many circumstances, especially in real-world applications, the context might change. This could lead FRBSs to sometimes lose robustness or even fail to operate when switching to a new context. While it is true that the fuzzy rules used to model universal knowledge can be fixed, fuzzy sets which represent different concepts are context-dependent and should be adjustable. For example, a simple car driving rule such as "*if the car speed is too fast, then reduce the speed*" is valid in all contexts. However, the understanding of the linguistic term "too fast" might change depending on the driving situations, for example, 60 km/h might be too fast when the car is taking a tight corner but is rather slow in a high speed motorway. Standard fuzzy sets cannot model this kind of context dependence directly. However, this could be achieved by introducing new linguistic terms to model the context as well as the corresponding representation of the term "*too fast*" in each context, e.g., $(context_1, too\ fast_1)$, $(context_2, too\ fast_2)$, etc. The same rule has to expand into a

series of rules of the form "*IF the car speed is too* $fast_i$ *and the context is* $context_i$*, THEN reduce the speed*" where i is the index of the context. Naturally, the introduction of new linguistic terms causes the number of rules to increase significantly, which reduces both the interpretability and the efficiency of the system. When the input variables are inter-dependent or closely related in their meanings, it is not always possible to fully model the inter-relationships no matter how many additional rules and terms are used. For example, the linguistic variable "too fast" might depend on the value of another variable "radius of the turn", which could take any real value from zero to infinity. To fully represent this relationship would require an unlimited number of fuzzy rules, which is impossible due to computer memory restriction. As a result, the loss of accuracy using standard fuzzy sets is unavoidable in this case. Thus, the standard fuzzy approach, despite having many advantages due to its simplicity, is not sufficiently flexible to model the influence of context on the meanings of linguistic terms in an accurate and concise manner.

To the best of our knowledge, several attempts have been made to address the issue of context dependence in the literature. In the field of fuzzy expert systems, the most common approach is to separate the creation of unchanged knowledge; i.e., the default rule base and data base; and dynamic knowledge, i.e., knowledge which changes under certain circumstances, into two different processes. The unchanged knowledge is always defined as the first step. Subsequently, any number of fuzzy context can be defined (or reused) by experts and will replace the default contexts in the final inference process. This separation allows decision-makers to concentrate solely on the inferential reasoning process, knowing that the experts maintaining the system can model the data at a later stage. The approach has been successfully applied in fuzzy control systems [132] and fuzzy medical support systems [120]. The main advantage of this approach is that it allows a linguistic term to have different interpretations depending on other entities. For example, many medical entities, especially quantitative medical data, can have different meanings depending on age, sex and health conditions. A drawback of this approach is that the creation of contexts and their corresponding fuzzy sets requires expert involvement, which hinders the automation process.

In the field of fuzzy modelling, major literature has pointed to the approach proposed by Gudwin [53]. The key point of this approach is that the fuzzy rules are expected to be universal to a high extent, and therefore are fixed, while the concepts represented by the fuzzy sets might change. As the first step, a normalised FRBS, which contains a fixed set of fuzzy rules as well as linguistic terms associated with fuzzy sets uniformly distributed on a normalised universe of discourse, is created by experts or by using identification methods. Subsequently, the normalised fuzzy sets are adapted to the specific context by using linear or polynomial-based transformation functions. The parameters of the transformation functions are tuned and optimised in a process called context adaptation in order to maximise or minimise the objective function. At the end of the context adaptation process, the best performing set of membership functions (MFs) are obtained and will be fixed in the entire fuzzy inference process. This context adaptation method can be thought of as an alternative fuzzy optimisation method and has been applied in several function approximation and classification applications [6, 12, 34, 54, 74, 85, 86]. In terms of dealing with modelling context dependence of fuzzy sets, the context adaptation approach suffers from the following major limitations:

- A scaling function applied to a base MF of a linguistic term is only dependent on that linguistic variable. Therefore, similar to a standard fuzzy approach, the dependence of one linguistic term on others cannot be fully captured using this approach.

- Cognitively, finding a function that transforms the base MF to the expected shape and distribution is not always intuitive especially for non-linear functions, e.g., mental visualisation of transforming a triangular MF to a trapezoidal MF is not trivial. Therefore, the parameters in the transformation functions are better found using an automated optimisation or learning method rather than manual expert tuning.

- The number of contexts is dependent on the partitioning of the linguistic variables. Each partition will have different scaling function. Accordingly, only a limited number of contexts can be realistically modelled using this approach.

- The use of non-linear transformation functions almost certainly garbage the interpretability of the fuzzy sets. Even with linear functions, care must be taken to preserve the semantic ordering of linguistic terms, e.g., the term "high" should always follow "low", "young" should always precede "old".

In summary, most existing context modelling approaches are designed for a specific field of applications and usually only partially solve the problem of context dependence. Approaches designed for expert systems are usually not easily generated automatically by learning methods, and approaches designed to be tuned by learning methods are not suitable to be used by experts. Regardless of the approaches, the definitions of context models are largely ignored, e.g., [6, 12, 34, 54, 74, 85, 86], or left entirely for the experts. In other words, previous methods are only concerned with modelling the effects that are caused by the contexts rather than what the contexts are actually built from. While it is commonly acknowledged that successful applications of fuzzy sets depend on, to a large extent, the transparent and meaningful representations of knowledge, this is quite surprising given the lack of a semantically well-defined context model. In addition, none of the existing methods that claim to model context in the literature can fully capture the dependence of the meanings of linguistic terms on other terms or entities with infinite universe of discourse. This is a major disadvantage because it has been shown in many applications that sometimes by capturing the hidden relationships amongst inputs, even more useful information than the original inputs can be found and utilised to improve the transparency and interpretation of the system. For example, consider a fuzzy medical system calculating the likelihood of heart attack of a patient with weight and height of the patient are two of the inputs. The weight and height can be used to calculate the body mass index of the patient, which is obviously more useful than both weight and height in measuring the risk of obesity, which directly increase the likelihood of a patient having heart attack. The body mass index can be stored in an intermediate variable and subsequently used as an additional input of the fuzzy system. The introduction of new inputs which are calculated from the system's original inputs is not allowed in standard fuzzy approaches as well as existing context dependent fuzzy approaches. This method is called hierarchical fuzzy systems

(HFS) [111], which was designed to solve the "curse of dimensionality" problems in standard fuzzy systems. Although not specifically designed for context modelling, this approach certainly can be used for the task. However, there are also some drawbacks of this approach:

- Intermediate variables also introduce additional rules. The number of fuzzy rules in general hierarchical fuzzy systems increases polynomially with the growth of input variables. In the case of standard fuzzy systems, the number of rules in the fuzzy rule base increases exponentially with the number of inputs. While it is required a lot fewer rules than standard fuzzy systems, the number of rules is still significant which hinders the transparency and interpretation (important advantages of fuzzy systems) of the HFS.
- In the field of fuzzy modelling, expert knowledge about the problems is often lacking or very limited, it is not clear whether HFS is better than standard fuzzy systems both in terms of transparency and accuracy.
- Intermediate variables might or might not have any physical meanings. If these variables do not have any physical meanings but only introduced for calculation purposes, the interpretability of the system is reduced. It is therefore required to handle these variables efficiently to ensure both transparency and interpretation

This thesis aims at a solution which addresses the unsolved issues in previous context dependent fuzzy approaches as well as HFS. In particular, the new method should address the absence of a semantically well-defined context model in previous approaches and at the same time provide a solution with equal or better capability than HFS to handle the inherent "curse of dimensionality" problems when modelling the dependence of the meanings of linguistic terms on other terms or entities with infinite universe of discourse.

1.2 Objectives

There are two primary goals of thesis:

1. **To solve the issue of generality of linguistic terms by incorporating contextual information into the representation of fuzzy sets** so as to improve accuracy while still preserving interpretability of FRBSs. The method should address the following points:

    - Address the lack of transparency pertaining to previous approaches by using a semantically well-defined context model;
    - Improve the modelling capability both in terms of interpretability and accuracy of standard fuzzy sets so that the dependence of one linguistic term on others can be fully captured; and
    - Create a new fuzzy approach that is amenable to both expert design and automatic generation from learning methods.

2. **To investigate the effectiveness and role of the proposed fuzzy model** compared to other approaches, especially conventional fuzzy approaches in several research areas. In particular, the following issues will be addressed:

    - The expressive capability of our approach as a stand alone method compared to conventional fuzzy approaches both in terms of interpretability as well as accuracy;
    - The role of the proposed context fuzzy model in fuzzy modelling tasks; and
    - The advantages of combining the proposed approach with other techniques in a multi-agent framework.

1.3 List of papers

This thesis is based on theoretical and experimental research which has been previously written up in six scientific papers, all of which have passed the peer-review process and have been accepted for publication in various international conferences and journals with high academic standards. This section lists the papers, along with in what section(s) of chapter(s) of this thesis, the work in the paper is presented and discussed. The papers are:

Conference papers:

- Duc Thang Ho and J. M. Garibaldi. A novel fuzzy inferencing methodology for simulated car racing. Proceedings of the IEEE International Conference on Fuzzy Systems, pages 1909-1916, 2008. Extensively discussed in section 4.2
- Duc Thang Ho and J. M. Garibaldi. A fuzzy approach for the 2007 CIG simulated car racing competition. Proceedings of the 2008 IEEE Symposium on Computational Intelligence and Games, pages 128-134, 2008. Extensively discussed in section 4.3
- Duc Thang Ho and J. M. Garibaldi. Context modelling in fuzzy systems. Proceedings of the IEEE International Conference on Fuzzy Systems, pages 1-7, 2012. Extensively discussed in chapter 3.
- Duc Thang Ho and J. M. Garibaldi. An improved optimisation framework for fuzzy time-series prediction. Proceedings of the IEEE International Conference on Fuzzy System. 2013

Journal papers:

- Julian Togelius, Simon Lucas, Duc Thang Ho, Jonathan M. Garibaldi, Tomoharu Nakashima, Chin Hiong Tan, Itamar Elhanany, Shay Berant, Philip Hingston, Robert M. Maccallum, Thomas Haferlach, Aravind Gowrisankar, and Pete Burrow. The 2007 IEEE CEC simulated car racing competition. Genetic Programming and Evolvable Machines, 9(4):295-329, 2008. Briefly discussed in section 4.3.2
- Duc Thang Ho and J. M. Garibaldi. Context-Dependent Fuzzy Systems With Application to Time-Series Prediction. Accepted for publication in IEEE Transactions on Fuzzy Systems. 2013. doi: 10.1109/TFUZZ.2013.2272645. Extensively discussed in chapter 5.

1.4 Organisation of thesis

The first two chapters of this thesis, including this chapter, provide the background for the discussion of existing fuzzy techniques, their applications and some of the previous attempts to model the influence of context on the meanings of linguistic terms. The contributions of this thesis are presented from chapters 3 to 6. In chapter 3, the context-dependent fuzzy sets and the structure of a context-dependent fuzzy model are described in detail. Chapter 4 presents the award-winning fuzzy controllers developed by the proposed fuzzy method for the FUZZ-IEEE 2007 and the 2007 IEEE CEC Simulated Car Racing Competitions. In chapter 5, the roles and benefits of integrating a context model into fuzzy modelling methods are discussed and evaluated in the well-known Mackey-Glass time-series prediction benchmark. Chapter 6 describes the TORCS-based Simulated Car Racing Championship and the use of our fuzzy approach in the development of a hybrid fuzzy controller for the racing problem. The final chapter puts the results in context and discusses future directions for the research.

CHAPTER 2

Literature review

This chapter provides the background knowledge of existing fuzzy approaches and their applications, as well as some of the attempts to solve the issue of generality of linguistic terms by incorporating contextual information into the representation of fuzzy sets that are necessary to understand the rest of thesis.

2.1 Fuzzy systems

2.1.1 Fuzzy sets

Fuzzy sets and the related theory was introduced by Lotfi A. Zadeh [143] in 1965 as an extension of the classical set theory. In the classical set theory [71], an element x either belongs or does not belong to a set A. The membership function $\mu_A(x)$ which represents the membership of x into A is given by:

$$\mu_A(x) = \begin{cases} 1 \; if \; x \in A \\ 0 \; if \; x \notin A \end{cases} \tag{2.1.1.1}$$

By contrast, fuzzy set theory permits partial set membership which allows the gradual assessment of the membership of elements in a set. In other word, an object x can belong to the set A to a certain degree. Let U be the set of all possible values of the element x. U is referred to as the universe of discourse of x. The membership function

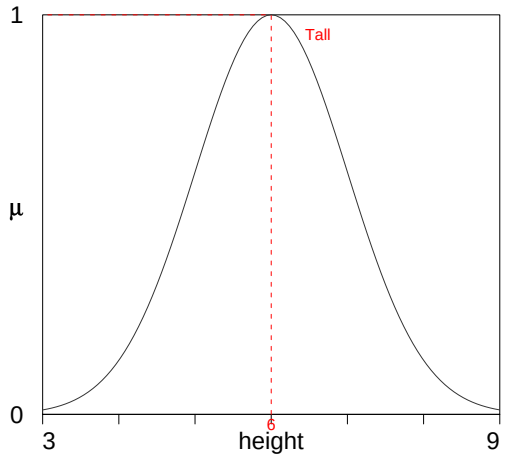


Figure 2.1: Fuzzy representations of the set of tall people

$\mu_A(x)$ which associates the membership of x into the fuzzy set A is given by:

$$\mu_A \ : \ X \rightarrow [0,1] \tag{2.1.1.2}$$

The membership value of 0.0 represents absolute FALSENESS while the value of 1.0 represents absolute TRUTH. Fuzzy sets are usually more flexible and more accurate than crisp sets when representing real world concepts due to the gradual transition from absolute membership to non-membership. A typical example would be the problem of where, given the height of a person, we are required to determine if he/she is tall or not. In other words, we need to define the set of tall people. Most people would agree that any person who has the height of *around* 6 feet or higher can be considered tall. With classical sets, people with height equal to or higher than 6 feet belong to the group of tall people and those whose height is less than 6 feet are considered not tall. Fuzzy sets allow a much more reasonable and at the same time more accurate way of defining the set as shown in Figure 2.1. The smoothly varying curve around the 6 feet value represents the meaning of the vague term "around 6 feet", which is commonly used in human conversations. Similarly, people who are much taller than 6 feet belong more to the set of *very tall.* People who are much shorter than 6 feet belong more to the set of *very short.*

2.1.1.1 Operations on fuzzy sets

Similar to operations on classical crisp sets, intersection, unification and negation operators can be defined for fuzzy sets. It has been suggested by L. A. Zadeh [143] that the minimum and maximum functions can be used as intersection and unification operators respectively. Tables 2.1 and 2.2 show that these operators coincide with the crisp unification, and intersection if membership degrees of 0 and 1 are considered. Figure 2.2 visually demonstrates the standard fuzzy operation. The operation and formulae described below are taken from the paper [143] of L. A. Zadeh. The general form of intersection and union are represented by triangular norms (T-norms) and triangular conorms (T-conorms or S-norms). Both T-norms and T-conorms are two-place functions from $[0,1] \times [0,1] \to [0,1]$. T-norms must satisfy the following criteria:

$$\begin{aligned} T(a,1) &= a && \textit{One identity} \\ T(a,b) &\leq T(c,d), \textit{ whenever } a \leq c,\ b \leq d && \textit{Monotonicity} \\ T(a,b) &= T(b,a) && \textit{Commutiativity} \\ T(T(a,b),c) &= T(a,T(b,c)) && \textit{Associativity} \end{aligned} \tag{2.1.1.3}$$

T-conorms or S-norms must satisfy the following criteria:

$$\begin{aligned} S(a,0) &= 0 && \textit{Zero identity} \\ S(a,b) &\leq S(c,d), \textit{ whenever } a \leq c,\ b \leq d && \textit{Monotonicity} \\ S(a,b) &= S(b,a) && \textit{Commutiativity} \\ S(S(a,b),c) &= S(a,S(b,c)) && \textit{Associativity} \end{aligned} \tag{2.1.1.4}$$

The most common T-norms are minimum and product functions while the most common T-conorms are maximum and probabilistic sum functions.

Fuzzy operations are defined as follows:

- Union:

$$\mu_{A \cup B} = S(\mu_A, \mu_B) \tag{2.1.1.5}$$

A	B	A and B	A or B	not A
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	1

Table 2.1: Standard Boolean Logic Operations

A	B	min(A,B)	max(A,B)	1-A
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	1

Table 2.2: Fuzzy Logic Operations

- Intersection:

$$\mu_{A\cap B} = T(\mu_A, \mu_B) \tag{2.1.1.6}$$

- Complement:

$$\mu_{\overline{A}} = 1 - \mu_A \tag{2.1.1.7}$$

Theoretical operations on fuzzy sets (union, intersection and complement) are the basis for fuzzy logical operations (*or*, *and* and *not*).

2.1.1.2 Linguistic variable

Fuzzy sets can be conveniently used to interpret linguistic terms such as "old", "young", "cold", "hot", etc. Based on the idea of fuzzy sets, Zadeh [145] also introduced the definition of linguistic or fuzzy variables whose values are words or sentences in natural or artificial language. Formally, a linguistic variable is characterised by a quintuple $(\chi, T(\chi), U, R, M)$, where

- χ is the name of the variable such as Age, Temperature, etc.
- $T(\chi)$ is the term set of χ. For example,

$$T(Age) = \{young,\ very\ young,\ not\ young,\ old,\ very\ old, ...\} \tag{2.1.1.8}$$

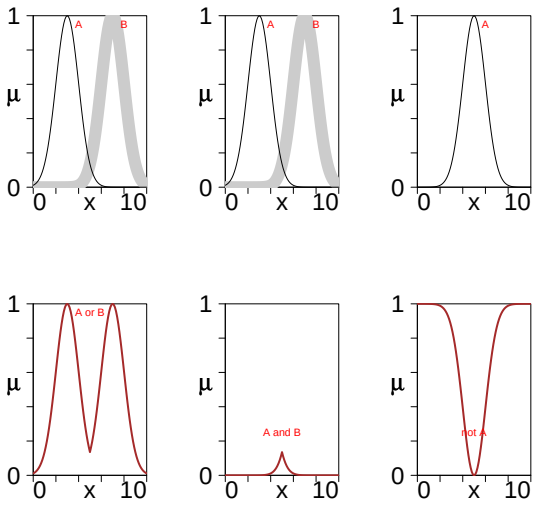


Figure 2.2: Standard Fuzzy logic operations

- U is the universe of discourse of the base variable. For example, the universe of discourse for Age may be taken to be the interval $U = [0, 100]$
- R is syntactic rule for generating linguistic terms of $T(\chi)$
- M is a semantic rule which associates each linguistic value X with its meaning (fuzzy sets).

The main purpose of linguistic variables is to replace large domain sets of real numbers with a finite set of partitions conceivable to human beings. As an illustration, consider the linguistic variable Age, i.e., χ= Age. The term set $T(\chi)$ is represented as

$$T(Age) = \{Young,\ MiddleAge,\ Old\} \tag{2.1.1.9}$$

The universe of discourse for Age is $U = [0, 100]$. A linguistic value of Age, for example *Young*, is a name of a fuzzy set which defines the meaning of the term *Young*. For any given numerical value of Age, the compatibilities of the value with each of the linguistic term can easily be achieved using the membership functions of each fuzzy set. For example, the compatibilities of the numerical ages 18, 23 and 34 with *Young* may be 1, 0.85 and 0.3 respectively. See figure 2.3 for a graphical representation of the linguistic variable Age.

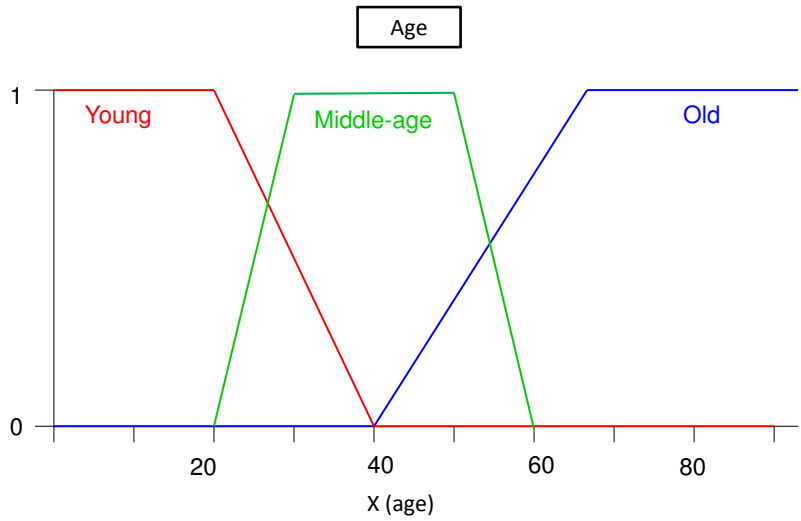


Figure 2.3: Linguistic variable

It is worth mentioning that one of the great utilities of linguistic variables is that they can be modified by applying linguistic hedges to the terms, such as *more or less, quite, extremely*, and so on. Moreover, linguistic values can also be combined using connectives to create a composite linguistic value from the primary terms. With all these features, linguistic variables have proven to be particularly useful in expressing rules and facts, fundamental parts of any fuzzy system, in a human-friendly manner.

2.1.1.3 Fuzzy systems

Fuzzy sets and fuzzy logic form the basis of fuzzy rule-based systems [144]. The main component of a typical fuzzy system consists of a knowledge base (KB) and an inference layer as shown in Figure 2.4. The knowledge base - component of FRBS that stores expert knowledge about the problem being solved – is composed of:

• a rule base (RB), constituted of a collection of linguistic IF-THEN rules, such as *if the temperature is hot, then switch on the fan*, and

• a data base (DB), which associates semantics, represented by a fuzzy set and its membership functions, with each of the linguistic terms used in the RB, e.g., *hot*.

A fuzzy rule describing the relationship between the linguistic labels of input and output variables consists of a premise and a consequent part. An example of such a rule might be

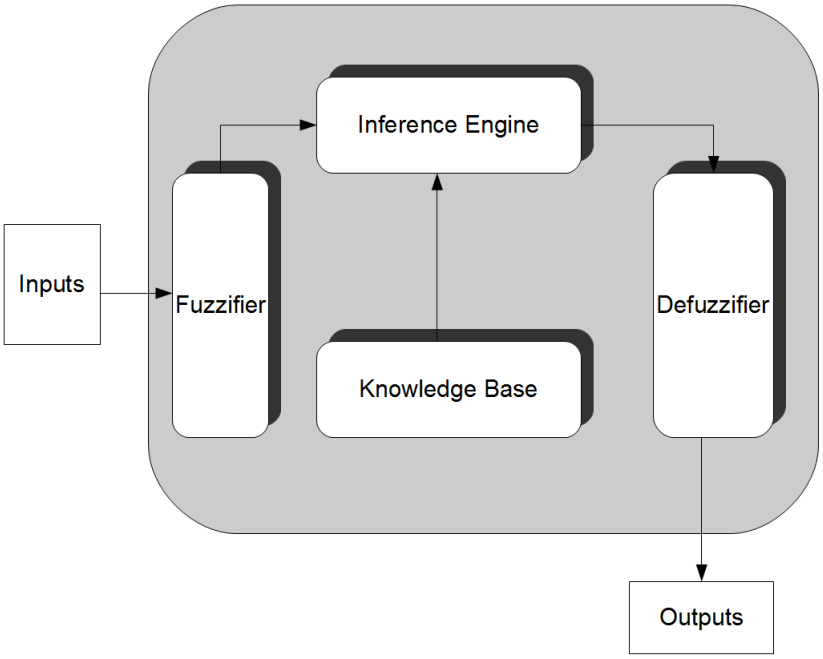


Figure 2.4: Structure of a standard fuzzy system

IF SERVICE IS GOOD THEN TIP IS AVERAGE

The IF-part of the rule "*SERVICE IS GOOD*" - is called the antecedent or premise part, while the THEN-part of the rule "*TIP IS AVERAGE*" is called the consequent part. The concept *GOOD* is represented as a fuzzy set that maps each value of *service* to a number between 0 and 1, and therefore the antecedent is an interpretation that returns a single number between 0 and 1. This value is called the *support* of the antecedent. The consequent part, *TIP IS AVERAGE*, then uses this *support* value to truncate the fuzzy set represented by *average* and return it as the output of the IF-THEN rule. The inference layer comprises of three parts: a fuzzifier, an inference engine and a defuzzifier .

- **Fuzzifier**

The fuzzifier takes the inputs and determines the extent to which they belong to each of the appropriate fuzzy sets via membership functions. Figure 2.5 shows to what extent the value of X respectively belongs to the set of *low* and *high*. In this case, the value of X is 0.2, which, given the graphical definition of X, corresponds to $\mu = 0.61$ in the *low* membership function and $\mu = 0$ in the *high* membership function.

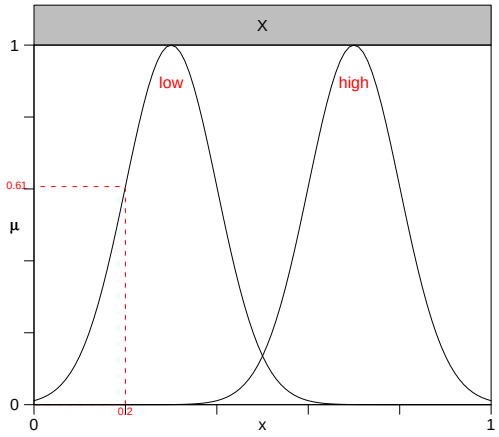


Figure 2.5: Fuzzification

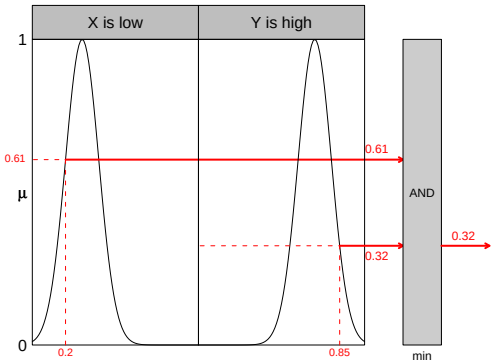


Figure 2.6: Rule evaluation: X is low and Y is high

- **Inference engine**

The inference engine applies fuzzy logic reasoning on the fuzzy inputs obtained from the fuzzifier to compute fuzzy outputs. If the antecedent has multiple parts, all parts of the antecedents are calculated simultaneously using fuzzy logical operators to obtain one number that represents the result of the antecedent for that rule. This number is called the *support* of the antecedent which is then applied to the output function. The aim is to truncate the output fuzzy set to the degree specified by the value of the *support* of the antecedent. Figure 2.6 shows a detailed example of a rule evaluation with two inputs $X = 0.2$ and $Y = 0.85$. The operator in use is the AND (min) operator. The *support* of the antecedent is 0.32 in this case

The *support* of the antecedent is then used to reshape the consequent in the process

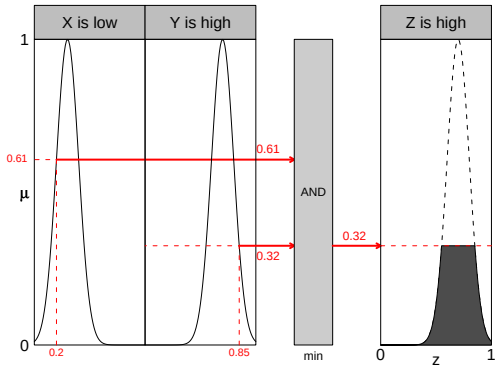


Figure 2.7: Rule inference: IF X is low and Y is high THEN Z is high

known as *implication*. In the case of two-valued or binary logic, the implication process is straight forward. If the antecedent is true, then the conclusion is true. In the case of fuzzy logic, if the antecedent is true to some degree, then the consequent is also true to that same extent. Thus:

- in binary logic: $p \rightarrow q$ (p and q are either both true or both false.)
- in fuzzy logic: $0.3p \rightarrow 0.3q$ (Partial antecedents provide partial implication.)

The *implication* function usually modifies the output fuzzy set by a truncation using *T-norms*. The two standard T-norms functions that are often used as *implication* operators are minimum, which truncates the output fuzzy set, and product, which scales the output fuzzy set. The final results are fuzzy outputs of the inference engine. An example is shown in Figure 2.7 to demonstrate the inference process of the rule "IF X is low and Y is high THEN Z is high" with two inputs $X = 0.2$ and $Y = 0.85$

- **Defuzzifier**

  The purpose of the defuzzifier is to convert the output fuzzy set into crisp values. The outputs of each rule are first unified into a single fuzzy set in a process called aggregation. Subsequently, the aggregated output fuzzy sets are converted into crisp numbers in a process called defuzzification. Amongst the most popular defuzzification methods are centroid calculation, which returns the centre of the

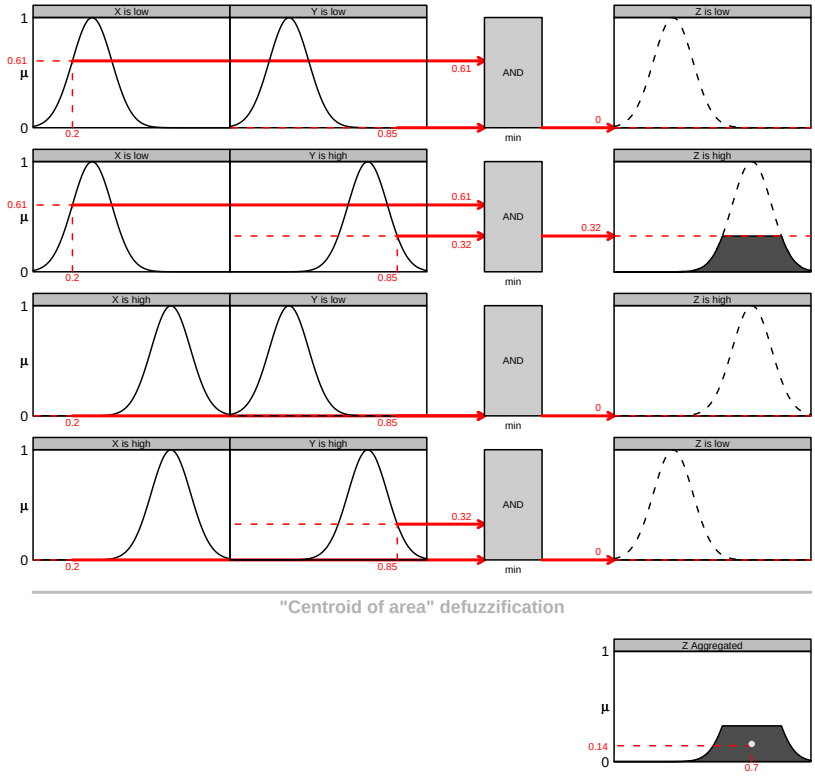


Figure 2.8: Aggregation & defuzzification

area under a curve as shown in Figure 2.8, and mean of maxima (MoM), which discriminates part of a fuzzy output where membership values are below a certain threshold level.

2.1.2 Takagi-Sugeno-Kang fuzzy systems

Fuzzy systems described above belong to the class of linguistic (Mamdani) FRBSs. The main feature of such type of FRBS is that both the antecedent and consequent parts of the rules are expressed using linguistic variables [146]. As a consequence, a Mamdani fuzzy system is highly verbally interpretable even though the defuzzification process might be time consuming. As such, Mamdani fuzzy systems are referred to as linguistic fuzzy systems and are usually suitable for application in situations where transparency is more important than accuracy, e.g., in expert systems, economics, and so on. Another type of FRBS, the Takagi-Sugeno-Kang fuzzy systems [121], is often applied in situations where accuracy is vital, e.g., in control systems, function approximations, time

series predictions, etc. This type of fuzzy systems focuses on increasing the accuracy as much as possible and pays little attention to the interpretability of the final model. As a result, the Takagi-Sugeno-Kang (TSK) fuzzy approach is also referred to as precise fuzzy approach. One of the main differences between TSK and Mamdani fuzzy structures is the use of linear or non-linear functions of the input in the consequent part of a fuzzy rule instead of linguistic variables. Typically, a TSK rule is of the form:

$$IF \ \ (x_1 \ is \ A_1) \ \ AND \ ... \ AND \ \ (x_n \ is \ A_n) \ \ THEN \ \ y \ = f(x_1, ..., x_n) \tag{2.1.2.1}$$

where $x_1, ..., x_n$ are input variables, $A_1, ..., A_n$ are fuzzy sets and y is the output.

In TSK fuzzy systems, the evaluation of the antecedents is identical to that of the standard fuzzy systems. However, there are some minor differences in the implication and aggregation process of the consequent parts. In particular, the *implication* function in TSK fuzzy systems is usually the product functions while in Mamdani systems, minimum function are often used. The aggregation operator in the TSK systems is usually the sum operator. However, aggregation and defuzzification are often combined into a single defuzzification operation in the TSK approach[67]. A review of the literature reveals the two most popular defuzzification methods. The first method is the weighted mean (WM) [114]:

$$y^* = \frac{\sum_{i=1}^{r} w_i y_i}{\sum_{i=1}^{r} w_i} = \sum_{i=1}^{r} \bar{w}_i f_i(x_1, ..., x_n) \tag{2.1.2.2}$$

where, r is the number of rules, w_i is the activation level according to the i^{th} rule. $\bar{w}_i$ denotes the normalised value of w_i

$$\bar{w}_i = \frac{w_i}{\sum_{i=1}^{r} w_i} \tag{2.1.2.3}$$

The second defuzzification method is the so-called weighted sum (WS) [76]

$$y^* = \sum_{i=1}^{r} w_i y_i \tag{2.1.2.4}$$

The computation of the final output using the defuzzification method in TSK fuzzy

systems are very efficient due to the simplicity of the formulae (WS, WM); while in the case of Mamdani fuzzy model, defuzzification is computationally expensive since the whole MF of the final output fuzzy set must be computed. Another advantage of the TSK fuzzy structure is that when the relationship between the input and output relationship can be modelled using mathematical analysis, direct use of the mathematical model in the computation of output is not only simpler and computationally cheaper, but also gives more accurate results than Mamdani's approach to defuzzification. These advantages of the TSK fuzzy approach make it highly useful in the fields of control and modelling where both accuracy and efficiency are extremely important factors.

2.1.3 Fuzzy modelling

System modelling is a technique to express, analyse and transform the behaviours of real systems using mathematical models. Models can be used in different ways as part of a process for: simulation, enhanced comprehension of the underlying mechanism, problem identification, decision-making support, optimisation and so on. Traditionally, modelling problems are solved by experts with a thorough understanding of the system's nature and behaviour. This approach is called white box modelling. However, in practice, it is not always possible to achieve complete understanding of the underlying mechanisms of the system due to a number of reasons, including, poor understanding of the underlying phenomena, inaccurate measurements, or unpredictable nature of the model itself. In some other applications, prior knowledge of the system may be completely absent, for example, in time series predictions or function approximations. In such instances, a different approach is to perform system identification using some sufficiently general black-box structure as a general function approximator. Black-box models usually are typically not related to the structure of the real system, and model parameters do not have any physical or verbal significance. Therefore, *black box* modelling approaches, despite producing more reliable models, commonly provide little insight into the underlying process [23].

There have been several attempts to combine the advantages of both black box and

white box approaches in such a way that certain known parts of the system are modelled using physical knowledge and the unknown or less certain parts are identified with black box procedures. These methods are often referred to as *grey-box modelling*, see, e.g., [10, 72, 77]. Today, *fuzzy modelling* (FM) is one of the most successful *grey-box* approaches, which usually considers model structures in the form of FRBSs and construct them using black box methods. With this approach, a better balance between reliability and comprehensibility is attained [22]. A major advantage of fuzzy modelling compared to other standard modelling techniques such as artificial neural networks, is that extra information, such as expert knowledge and experience, which is often imprecise and qualitative by nature, can be conveniently incorporated using descriptive language easily interpretable by human beings. The transparent representation of linguistic terms using fuzzy sets and the logical structure of fuzzy rules facilitate the understanding of models in a way largely similar to humans reasoning in the real world.

There are two methods to construct a fuzzy system using fuzzy modelling - identification of fuzzy systems from a collection of data (measurements) and expert knowledge. The latter is largely problem-dependent and usually involves a more heuristic method rather than an exact algorithm. The former is often carried out by learning methods which automate the identification of the system's parameters. However, it is usually the case that at least some parameters (that have a dramatic impact on modelling quality) such as the number of rules, the type of fuzzy systems, the number of variables and so on, are given by experts. Therefore, fuzzy modelling is always more or less a combination of both white box and black box approaches.

2.1.3.1 Fuzzy models

The main objective of fuzzy modelling are to increase (i) interpretability, which refers to the capacity of the fuzzy model to express the behaviour of a system in an understandable manner, and (ii) accuracy, which refers to the capability of the fuzzy model to faithfully represent the modelled system [22, 23, 117]. However, since these are conflicting issues, it is generally not possible to achieve both criteria to a large extent. Depending

on the nature of a problem, more priority is given to one of them, leaving the other in the background, e.g., expert systems favour interpretability, control systems favour accuracy. Hence, fuzzy modelling techniques can be classified as linguistic fuzzy modelling (LFM), which aims at delivering good interpretability, and precise fuzzy modelling (PFM) which aims at achieving good accuracy [23]. The first approach, linguistic fuzzy modelling, usually generates models with good interpretability, but their accuracy is very low, while the latter can produce very accurate models, but with very low levels of comprehensibility. The most common linguistic fuzzy model is the Mamdani model which considers both the antecedent and consequent as standard fuzzy propositions. The most common precise fuzzy model is the TSK fuzzy model whose antecedent is a fuzzy proposition while the consequent is a crisp function. Details of the two types of fuzzy models have been given in previous sections.

2.1.3.2 Fuzzy modelling techniques

This section provides a brief overview of some common fuzzy identification algorithms that have been used for both LFM and PFM. Regardless of the approach, a common scheme observed in most fuzzy modelling tasks is as follows:

1. First, the main objective is determined, i.e., priority is given to interpretability or accuracy, which in turns determines the model structure, e.g., Mamdani or TSK fuzzy structure, etc.

2. Then, the model components (*model structure* and/or *modelling process*) are learned or tuned by a different mechanism to achieve the main objective. Typically, there are two ways to improve the system model components

    - Improve the *modelling process* by extending the model design to other components besides RB such as the DB design, or consider a more sophisticated RB learning method. Some examples of such techniques in the literature include deriving the membership functions [11, 32, 33, 56, 58, 103], learning/tuning membership function shapes [11, 32, 51], using cooperative rules

[20, 21, 24], using clustering techniques, simplifying and reducing fuzzy sets and rules [22, 141], etc.

- Improve the *model structure* by slightly changing the rule structure to make it more flexible or more interpretable depending on the purpose. Some examples include using linguistic modifiers [13, 142], weighted rules [29, 65, 100, 116], hierarchical knowledge bases [35, 66], etc.

Earlier approaches of fuzzy modelling usually involved only a single algorithm that was used for both the premise and consequent part identifications. Recent development of fuzzy modelling has shown a very important step towards a high-quality identification method, i.e., the combination of several techniques that simultaneously aim at optimising different sets of parameters (e.g., premise and consequent parts). Integrated algorithms have shown a dramatic improvement in performance compared to stand-alone methods. The most famous integrated algorithm is perhaps the ANFIS [68] in which the gradient descent and the least squares methods are combined to tune the parameters of a TSK fuzzy system. The primary innovation of the approach is the use of LSM for consequent parameter identification. This not only improves the convergence speed of the optimisation algorithm but also significantly increases the accuracy of the tuned system. Today, one of the best hybrid methods for precise fuzzy modelling involves a combination of clustering techniques, optimisation search algorithms (e.g., genetic algorithm, simulated annealing) and least squares methods. For a comprehensive survey of past research efforts in the field of fuzzy modelling which attempted to improve the interpretability and/or accuracy of the models, please refer to [22] and [23]. It is important, however, to emphasise that the performance of the algorithms is very application-specific and as such no general methods for improving either interpretability or accuracy can be derived.

The following section briefly reviews the different attempts to integrate contextual information into fuzzy models.

2.2 Context modelling in fuzzy systems

Context refers to the set of facts, conditions or circumstances under which an event exists or occurs. Context plays an important role in the human reasoning process due to the fact that almost all reasoning activities rely heavily on a generally implicit background or experience yet acts as a necessary contextual dimension to knowledge. Everybody uses context in their daily lives, e.g., every person behaves according to the moral values and behaviours in the society in which he or she lives. However, these cultural and moral values, as well as socially-accepted behaviours may change over time, and most likely will differ from society to society. Fuzzy set theory and fuzzy logic, although being great tools to deal with vague and imprecise knowledge by modelling the human reasoning process, have attracted criticism for neglecting the effect of the surrounding contexts. In fact, the vague concepts modelled by fuzzy sets are defined with an implicit assumption that the environment in which the system is working is either universal or fixed. As a consequence, the meaning of a concept or term represented by fuzzy sets is always constant, while, in fact, the context might change, as consequently, might the meaning of a concept. This inflexibility could on occasion lead to an over-fitting problem in which an FRBS trained to work in one situation might perform poorly or even fail to operate when applied to a new context. Of course, one could try to model the influence of context on the understanding of a term using standard fuzzy logic by introducing terms to represent the context and integrating these new terms directly into the existing rule base. For example, a simple driving rule such as "*if the car speed is too fast, then reduce the speed*" is valid in all contexts. However, the understanding of the linguistic term "too fast" might change depending on the driving condition, for example, 60 km/h might be too fast when the car is taking a tight corner but is rather slow in a high speed motor-way. Standard fuzzy sets cannot directly model this kind of context dependence. However, this could be achieved by introducing new linguistic terms to model the context as well as the corresponding representation of the term "*too fast*" in each context, e.g., $(context_1, too\ fast_1)$, $(context_2, too\ fast_2)$, etc. The same rule has to expand into a series of rules of the form "*if the car speed is too* $fast_i$ *and the context is* $context_i$*, then reduce the speed*" where i is the index of the context. Naturally,

the introduction of new linguistic terms causes the number of rules to increase significantly, which reduces both the interpretability and the efficiency of the system. When the input variables are inter-dependent or closely related in their meanings, it is not always possible to fully model the inter-relationships no matter how many additional rules and terms are used. For example, the linguistic variable "too fast" might depend on the value of another variable "radius of the turn", which could take any real value from zero to infinity. To fully represent this relationship would require an unlimited number of fuzzy rules, which is impossible due to computer memory restriction. As a result, the loss of accuracy using standard fuzzy sets is unavoidable in this case. Thus, the standard fuzzy approach, despite having many advantages due to its simplicity, is not sufficiently flexible to model the influence of context on the meanings of linguistic terms in an accurate and concise manner. This section briefly reviews some of the past attempts in the literature to extend the model of fuzzy sets and/or the design of standard FRBS to incorporate contextual information into the fuzzy inferencing process.

2.2.1 Context modelling in mathematical field

In the mathematics field, the first person who coined the term "*context dependent fuzzy set*" was Thiele [124] in 1999. Based on the ideas of Kripke semantics, Thiele introduced the notion of a fuzzy set, the value of which depends not only on its own universe of discourse, but also on a set of worlds W. Later, in [113, 123], he defined approximate reasoning with context-dependent fuzzy sets as a generalisation of the classical fuzzy approximate reasoning based on the compositional rule of inference. In a separate work, Huynh [61–64], based on the context model of Gebhardt and Kruse [49] and the meta-theory of Resconi et al. [112], introduced an alternative context-dependent interpretation of linguistic variables. Huynh defined the set of context C as a finite, non-empty set of context values, e.g., the set of contexts used for formulating the concepts of Height of humans consisting of Japanese, American, British, and so on. This means that for each linguistic term, there are a fixed number of possibilities (context) of different meanings. In Thiele's theory, there is no restriction on the interpretation of set W of worlds, which makes it a very generic definition but difficult for direct

application into a fuzzy system. Huynh's idea can be thought of as a special case of Thiele's. These aforementioned studies mainly focus on mathematical models of context representation with limited applications in the computer science field found in the literature.

2.2.2 Context integration in fuzzy expert & control systems

Context modelling has been applied in several fuzzy applications in the fields of expert and control systems. In 1997, Turner [132] proposed a mechanism for determining the context-dependent meaning of fuzzy subsets. In his approach, three types of context are considered, viz. static context (unchanging knowledge), dynamic context (knowledge which changes under certain circumstances) and ephemeral context (temporary context). Static contexts can be thought of as a collection of default membership functions for fuzzy sets and are created by human experts. Dynamic contexts are structurally identical to static contexts, the only difference being that the dynamic contexts are created by context managers (users) in order to represent the current context adequately. The ephemeral context is used to uniquely identify which linguistic variable is being referred to when a new context is entered. Based on the outline of the three types of context, context dependence is solved manually by letting user define the appropriate fuzzy set which best describe the context. When no customised context is applicable, the default (static) context is used.

A similar concept has also been successfully applied in medical expert systems [120]. In the MedFrame/CADIAG-IV diagnosis system, the design of rule base and the construction of data base, i.e., the representation of the actual patient data, are two different processes. A rule base and a default data base, which is used whenever no customised context is applicable, are defined as the first step. Subsequently, any number of fuzzy contexts can be defined (or reused) by medical experts and will replace the default context in the final inference process. This separation allows the decision-makers to concentrate only on the inferential reasoning process, knowing that the experts maintaining the system can model the data at a later stage. The main advantage of this approach is that it allows a linguistic term to have different interpretations depending

on other entities. For example, many medical entities, especially quantitative medical data, can have different meanings depending on age, sex and health conditions. A drawback of this approach is that the creation of contexts and their corresponding fuzzy sets requires expert involvement, which hinders the automation process.

An alternative context modelling approach [147] has been introduced in the field of fuzzy relational databases. In this approach, two types of context-dependent interpretations are identified, i.e., query-dependent interpretation and attribute-dependent interpretation. The former represents fuzzy terms whose meanings are relative to a partial answer of a query. For example, the same term "fast" has different interpretations in the following two queries: "find names of athletes who are fast relative to sprinters" and "find names of fast athletes". The term "fast" in the first query refers to those who are relatively fast among sprinters although in fact, sprinters are usually very fast. The second refers to athletes who are fast in general. The attribute-dependent interpretation is used to model fuzzy terms whose meanings are highly dependent on several attributes. For example, the meaning of the term "good" in the following query "find names of runners in the London Olympic who have a good record", needs to be interpreted according to age, sex and running distance. Although this approach allows the definition of different MFs depending on their external attributes, the MFs of a linguistic variable is still dependent only on that variable and therefore the automatic derivation of MFs using the dependent attributes cannot be carried out.

In summary, the use of context modelling in fuzzy expert and control systems was designed specifically for experts to define the interpretations of linguistic terms in different scenarios. It is entirely up to the experts to decide which information is relevant to a context and how to construct a representation of fuzzy sets based on this information. For example, experts can use different shapes, such as Gaussian, triangular, trapezoidal and so on, to represent membership functions of the same linguistic term in different contexts. Although this flexibility can significantly improve both the accuracy and the interpretability of FRBSs in modelling context dependence, the fact that the methods carried out by experts are largely problem specific and more heuristics than exact algorithm makes the process automation a challenging task. As a result,

to the best of our knowledge, there is no paper in the literature that has attempted to extend any of the aforementioned approaches for the automatic identification of fuzzy system tasks.

2.2.3 Context in fuzzy modelling

One of the most important issues in the design of FRBSs is the derivation of the knowledge base. Clearly, the quality of the knowledge base determines the performance of the system, therefore the process of knowledge acquisition plays a critical role in fuzzy modelling methods. Traditionally, experts capable of providing system knowledge are often required. However, in some complex systems about which expert knowledge is incomplete or absent, some form of learning or adaptation is always needed to automate the knowledge acquisition process. In the literature, the notion of adapting the meaning of linguistic terms to specific contexts has been adopted as a modelling method for the automatic design of a DB. Contexts are modelled as some types of scaling functions which are used to either adjust the DB to the context or map the input and output variables onto a normalised universe of discourse over which the fuzzy sets are defined. Scaling functions could be linear or non-linear. Depending on whether the variables or the DB are scaled in a context, two different context adaptation approaches can be distinguished.

The first approach, which is often used in modelling control systems, considers a DB with its MFs defined in a normalised interval. The input variables are scaled to fit into the normalised universe of discourse defined in the DB prior to their linguistic encoding (fuzzification), and the normalised outputs are converted back to their real values at the end of the inference process. Several research studies have investigated the learning of the scaling functions and MFs in the DB.

- In *[85, 102, 104]*, scaling functions are fixed (predefined). Learning is focused on the normalise MFs of the DB
- In [85, 87, 122], the DB is unchanged while scaling functions are determined dynamically through adaptive learning

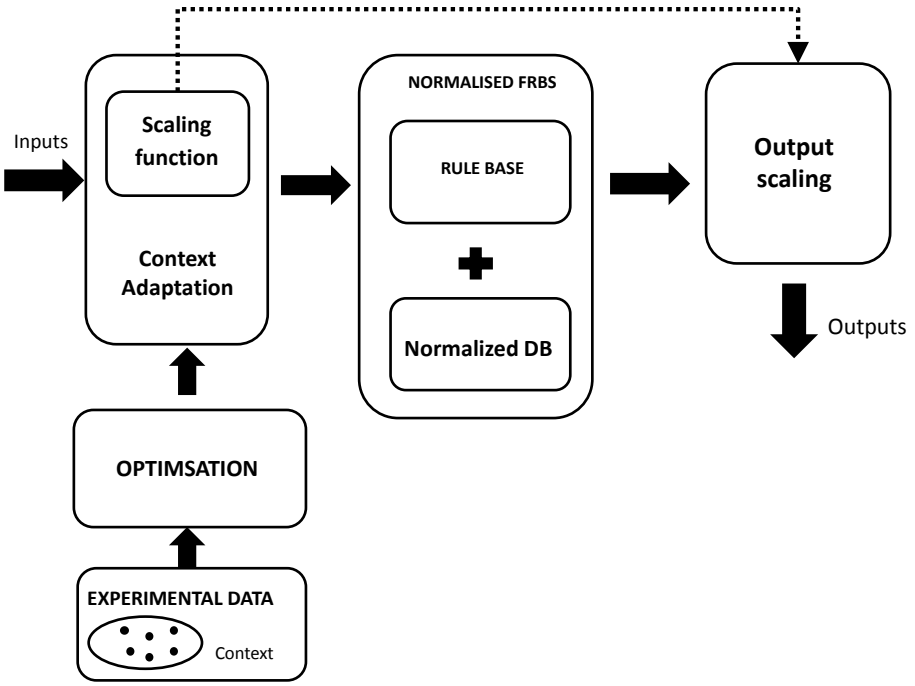


Figure 2.9: Context adaptation approach that scales variables

- In [46, 80, 96, 118], both scaling functions and the DB is tuned to achieve better performance.

Figure 2.9 shows an overview of this context adaptation approach.

The second context adaptation approach, which was proposed by Gudwin and Gomide [53], also considers a normalised DB, which is adjusted to the context according to the actual values of the variable under consideration. A similar notion of using linear and polynomial-based modifications of the universe of discourse has been proposed by Pedrycz [101] in the context of fuzzy control. In this approach, a normalised FRBS, which contains a fixed set of fuzzy rules as well as linguistic terms associated with fuzzy sets uniformly distributed on a normalised universe of discourse, is created by experts or by using identification methods. The RB and the normalised DB are always fixed while scaling functions are subject to learning. Figure 2.10 provides an overview of the process in this approach. This approach has been applied in several function approximation and classification applications [6, 12, 34, 54, 74, 86, 105].

There are two key points that are common in the two aforementioned approaches. First, the fuzzy rules are expected to be universal to a large extent, and therefore are always

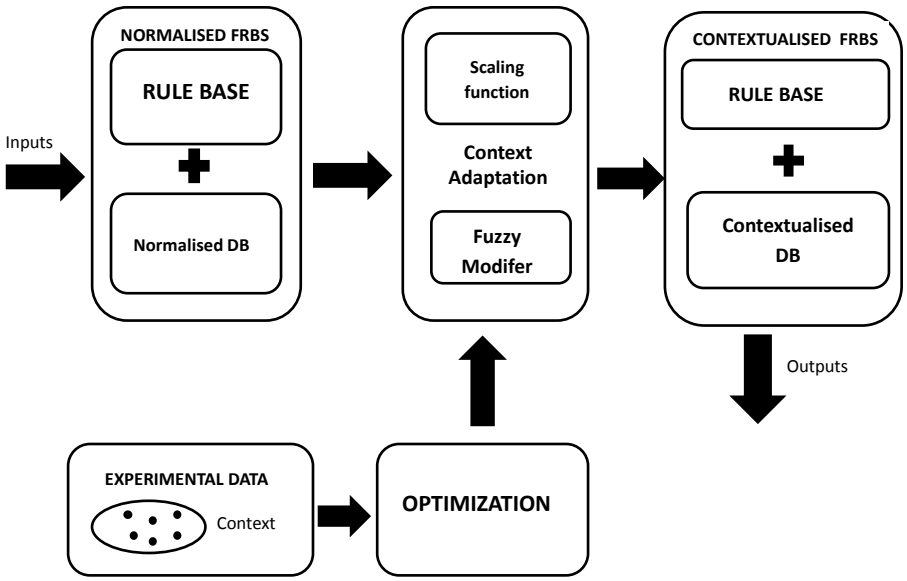


Figure 2.10: Context adaptation approach that modifies the DB

fixed; while the concepts represented by the fuzzy sets may change. Second, the normalised MFs defined in the DB only represent relative semantics of the linguistic variables (context independent). The absolute semantics of the linguistic variables (context dependent interpretation) are obtained through the use of scaling functions. The main difference lies in the way in which scaling functions are applied. Despite the similarities, the context adaptation approach which adjusts its MFs to the context is not only simpler, but also more intuitive as it is more similar to the way human beings adapt to the real world. As a result, nowadays, this approach has been widely considered as the standard context adaptation method. More details about this method is elaborated in the following part of this section.

Another approach, proposed by Moeljono and Ly [135], models contexts by means of context variables. These context variables define the behaviour of the system locally by filtering out irrelevant empirical data. Put simply, they essentially assign a value between zero and one to each element of empirical data based on its relevance to the context. In each context, the membership functions are truncated by the value of the context variables which indicate the adherence of the data to the context. However,

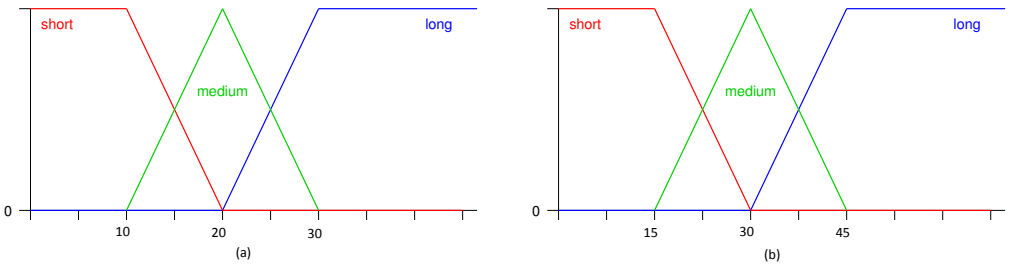


Figure 2.11: Linguistic variable "serving time"

apart from a few numerical examples to illustrate the implementation of the technique, no further application of this approach has been found in the literature.

2.2.3.1 The Gudwin's context adaptation approach

As mentioned earlier, the majority of literature has pointed to the approach proposed by Gudwin [53] as a standard context adaptation approach. The idea is to use scaling functions to re-evaluate a known concept, given that a different context is presented. In other words, the meaning of a concept is modified to adapt to a new context. The following example will help clarify this approach. Assume that fuzzy sets of the linguistic variable "serving time" are used to model the average waiting time of a customer after ordering in a restaurant, as shown in Figure 2.11. Figures (a) and (b) correspond to normal and peak hours respectively. As can be seen, in normal hours, a 30-minute serving time is considered long, but in peak hours, the same value is viewed as medium. The same concept (short, medium and long) might have different understanding in different context. However, the relative semantics of their original meanings remain unchanged (i.e., long is greater than medium, and medium is greater than short, etc.). To model these context-dependent fuzzy sets using Gudwin's approach, a standard frame of reference must be defined as the first step (Figure 2.12). The frame of reference, defined over a normalised universe of discourse, is used to model the relative semantics of a linguistic variable. The relative semantics of a term is context-independent and therefore is always fixed. The absolute meaning of a fuzzy set in a context is obtained by applying a context-specific scaling function to the base set. The overall procedure of the method is illustrated in Figure 2.13

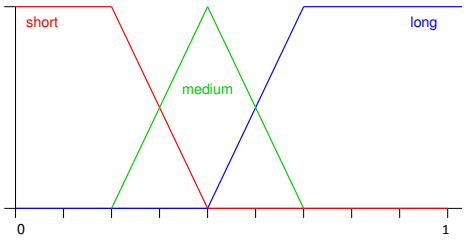


Figure 2.12: Frame of reference

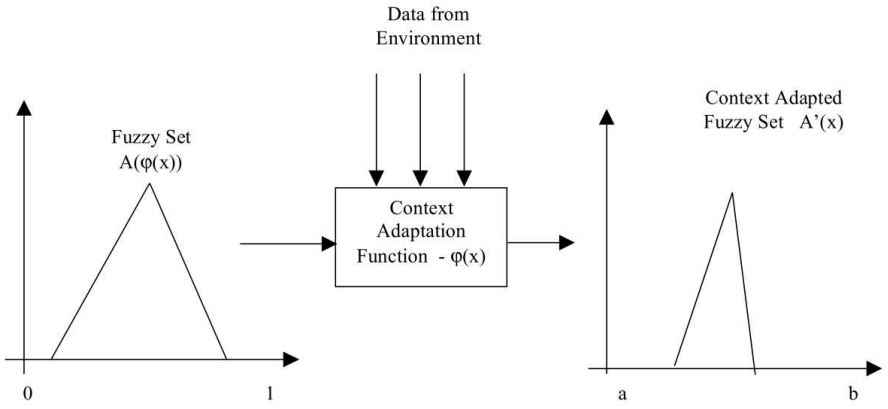


Figure 2.13: Context adaptation process

Formally, consider a collection of normalised fuzzy sets, which constitutes a frame of reference.

$$A = \{A_1, A_2, ..., A_3\} \tag{2.2.3.1}$$

Clearly, A must be created in such a way that the requirements of semantic integrity [104], such as unimodality and normality of the MFs of A_i, are satisfied.

Now, assume that a data set of experimental outcomes (coming e.g., from a certain process of expert polling or some measurements of physical systems) is given, which can be arranged in the form of $(c+1)$-tuples, namely

$$\begin{gathered} (d_1, (\mu_{11}, \mu_{12}, ..., \mu_{1n})), \\ (d_2, (\mu_{21}, \mu_{22}, ..., \mu_{2n})), \\ \vdots \\ (d_N, (\mu_{N1}, \mu_{N2}, ..., \mu_{Nn})), \end{gathered} \tag{2.2.3.2}$$

The target of the context adaptation approach is to best accommodate these data by adapting the context of A. The essence of the Gudwin's adaptation process is to map the unit interval of the base universe of discourse into the universe of discourse specified by the data set, i.e., $[a, b]$ where $a = \underset{i=1...N}{min}\, d_i$ and $b = \underset{i=1...N}{max}\, d_i$. A context C can be properly formalised as a tuple of (a, b, φ) where $\varphi : [0, 1] \rightarrow [a, b]$ is the scaling function transforming the base normalised fuzzy set to an appropriate one. According to Gudwin and Pedrycz [105], φ must satisfy the following criteria:

- Continuity;
- Monotonicity - φ must be non-decreasing; and
- Boundary conditions - $\varphi(0) = a$ and $\varphi(1) = b$. This condition is needed to ensure maximum accommodation of all experimental data.

Once the scaling function has been constructed, the contextualised fuzzy sets are obtained by computing $A_i^{'} = \varphi(A_i)$. In other words,

$$A_i^{'}(x) = \varphi(A_i(x)) \tag{2.2.3.3}$$

where $i = 1, ..., N$. These new fuzzy sets form the required frame of cognition $A^{'}$. Note that, there is another context adaptation approach which scales the input variables to fit into the normalised universe of discourse defined in A. In that approach, the mapping of φ is defined as $[a, b] \rightarrow [0, 1]$ and the frame of cognition $A^{'}$ is computed as $A_i^{'}(x) = A(\varphi(x))$, $x \in [a, b]$. More details and examples of the Gudwin's approach can be found in [12, 54, 74, 105]

2.2.4 Limitations of previous approaches

As evident by the studies reviewed in this section, several attempts to integrate context into the fuzzy inference process have been successfully applied in several applications

in the fields of fuzzy expert, fuzzy control and fuzzy modelling. Regardless of the approach, one common theme across all context modelling methods is that the meanings of linguistic terms (membership functions) are adjusted as the context changes while the rule base is always fixed. The main difference lies in the manner in which the membership functions get changed. In fuzzy expert systems, the creation of contexts and their corresponding fuzzy sets requires a high level of expert involvement. In addition, design methods carried out by experts are usually more heuristics than exact algorithm. As a result, the automation of expert methods is very difficult to achieve and therefore these approaches are not suitable to be used in fuzzy modelling tasks where the involvement of expert knowledge is minimal. In the field of fuzzy modelling, major literature has pointed to the approach proposed by Gudwin [53]. The approach suffers from the following major limitations:

- A scaling function applied to a base MF of a linguistic term is only dependent on that linguistic variable. Therefore, similar to a standard fuzzy approach, the dependence of one linguistic term on others cannot be fully captured using this approach.

- Cognitively, finding a function that transforms the base MF to the expected shape and distribution is not always intuitive especially for non-linear functions, e.g., the task of mentally visualising the transformation of a triangular MF into a trapezoidal MF is not trivial. Therefore, the parameters in the transformation functions are better found using an automated optimisation or learning method rather than manual expert tuning.

- The number of contexts is dependent on the partitioning of the linguistic variables. Each partition will have a different scaling function. Accordingly, only a limited number of contexts can be realistically modelled using this approach.

- The use of non-linear transformation functions almost certainly garbage the interpretability of the fuzzy sets. Even with linear functions, care must be taken to preserve the semantic ordering of linguistic terms, e.g., the term "*high*" should always follow "*low*", "*young*" should always precede "*old*".

- There is a lack of comparative studies concerning the benefits of the approach in comparison with other fuzzy methods. In most papers, the context adaptation approach is applied for toy problems. The main purpose of these papers is to demonstrate the idea of the method rather than providing a comprehensive analysis of its advantages. Although some initial results for the context modelling approaches in fuzzy systems are promising, more sophisticated applications should be carried out before any concrete conclusion regarding the effectiveness of the methods can be made.

All the aforementioned limitations are the reason why this approach is generally considered as a fuzzy identification method rather than as a tool to model human knowledge.

2.3 Related work

As mentioned previously, existing approaches on context modelling cannot fully model the dependence of an input on another variable (or context). As a result, none of the existing context dependent fuzzy approaches can really achieve what they claim to be able to, i.e., capture the changes on an input when circumstances change. However, there are other approaches which were not specifically designed to solve the problem of context dependence but possess the capability to do so. One of such approaches is Hierarchical Fuzzy Systems [111] (HFS). In HFS, a number of lower-dimensional sub-fuzzy systems (SFS) are linked in a hierarchical manner. HFS are characterized by having several sub-fuzzy systems contributing to the computation of the final solution. The lowest leveled sub-fuzzy systems receive the original input variables, and their outputs are used as the inputs to higher leveled sub-fuzzy systems. In general, hierarchical fuzzy systems are characterized by the number of hierarchical levels, the number of sub-fuzzy systems in each level, and whether the original inputs are just used at the lowest level or not . Figure 2.14 shows a general structure for HFS. The outputs of certain sub-fuzzy system are used as the inputs of other sub-fuzzy systems (neighbouring upper leveled sub-fuzzy systems). The inputs to the lowest level are all

the original inputs. The inputs to a particular level can be the outputs from its lower level, or a combination of its lower leveled outputs and original input variables. The outputs from each sub-fuzzy system can be thought of as intermediate variables (crisp variables) which required to be converted into linguistic variables so that it can be used in other sub-fuzzy systems. There can be many different structures of HFS, each of which is suitable for different applications, e.g., fuzzy classification [39, 108, 130], clustering [26, 40], trajectory planning and tracking systems [42]. See [133] for a more comprehensive survey of HFS.

2.3.1 HFS and context modelling

Naturally, intermediate variables can be used to model context. Instead of representing the changes of inputs in each different context, intermediate variables can be used to store the relative meanings of original inputs in each context. Each distinctive value of an intermediate variable represent the changes of an input variable in a specific context. Let consider the simple car driving rule "*if the car speed is too fast, then reduce the speed*". Obviously, raw speed of a car is not sufficient to determine whether the car is slow or fast. A context is therefore given, e.g., a car is about to take over another car right in front of it. Instead of using the original inputs, i.e., raw speeds of the cars, an intermediate variable which represents the relative difference between the speeds of the two cars can be used to determine whether the approaching car is going too fast or not. The use of intermediate variables in this case not only makes much sense but also avoid the "curse of dimensionality" problems when dealing with such kind of relationships in standard fuzzy systems. In general, HFS can be used as a good solution to the context modelling in fuzzy systems. However, there are also some drawbacks of this approach:

- Intermediate variables also introduce additional rules. The number of fuzzy rules in general hierarchical fuzzy systems increases polynomially with the growth of input variables. In the case of standard fuzzy systems, the number of rules in the fuzzy rule base increases exponentially with the number of inputs. While it is required a lot fewer rules than standard fuzzy systems, the number of rules

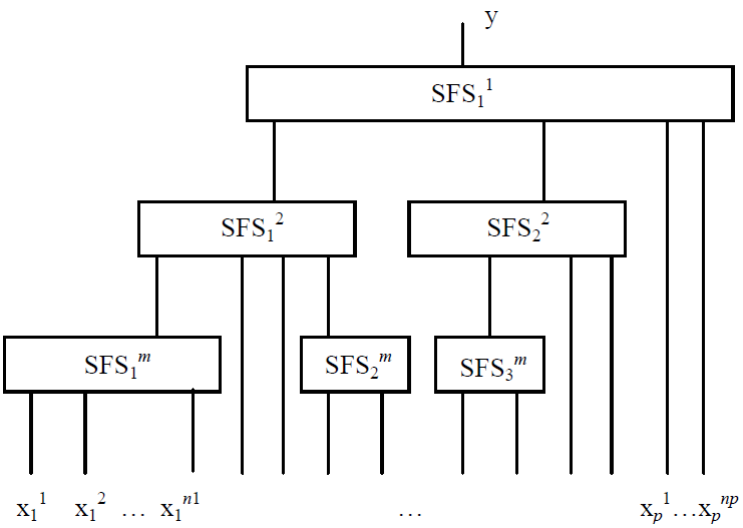


Figure 2.14: A general structure for HFS

is still significant which hinders the transparency and interpretation (important advantages of fuzzy systems) of the HFS.

- In the field of fuzzy modelling, expert knowledge about the problems is often lacking or very limited, it is not clear whether HFS is better than standard fuzzy systems both in terms of transparency and accuracy.

- Intermediate variables might or might not have any physical meanings. If these variables do not have any physical meanings but only introduced for calculation purposes, the interpretability of the system is reduced. It is therefore required to handle these variables efficiently to ensure both transparency and interpretation

- Determining when a HFS is better than a standard fuzzy system.

2.4 Summary

This chapter has provided the background knowledge about standard fuzzy approaches and their applications. It has also reviewed some of the methods to incorporate contextual information into the representation of fuzzy sets in the fields of fuzzy expert, fuzzy control systems and fuzzy modelling. It has been shown that the existing ap-

proaches are not sufficiently flexible to be extended to applications that they were not originally designed for. Approaches designed for expert systems are usually not easily generated automatically by learning methods, and likewise approaches designed to be tuned by learning methods are not suitable to be used by experts. None of the existing context modelling methods in the literature can fully capture the dependence of the meanings of linguistic terms on other terms or entities with infinite universe of discourse. In all previous work, the definitions of context models are largely ignored, or left entirely up to the experts. In other words, most solutions are only concerned with modelling the effects that are caused by the contexts rather than what the contexts are actually composed of. While it is commonly acknowledged that successful applications of fuzzy sets depend on, to a large extent, the transparent and meaningful representation of knowledge, this is quite surprising given the lack of a semantically well-defined context model. This chapter also presented HFS which is not specifically designed for context modelling but certainly can be used for the task. Despite some minor drawbacks, HFS is better than existing context modelling approaches when dealing with inter-relationship amongst inputs.

In the next chapter, our context modelling method based on a well-defined semantics and typing context model is presented. Our approach is designed with a specific aim to be amenable to both expert design and automatic generation from learning methods. A comparison between our approach and HFS is made to underscore the difference between the two.

CHAPTER 3

Context modelling in fuzzy systems

In this chapter, a novel context modelling method based on well-defined semantics and typing context model is presented. The approach aims at the following points:

- Address the lack of transparency pertaining to previous approaches by using a semantically well-defined context model;
- Improve the modelling capability both in terms of interpretability and accuracy of standard fuzzy sets so that the dependence of one linguistic term on others can be fully captured; and
- Create a new fuzzy approach that is amenable to both expert design and automatic generation from learning methods.

A number of examples will be provided to illustrate the interpretability and accuracy advantage of our approach in dealing with context dependence problems. An example also demonstrates the use of context modelling to automate the MF creation process based on contextual information. The automatic generation of MFs allows for the modelling of an unlimited number of contexts, which is not possible under previous context-dependent fuzzy approaches. This approach can be used model the interrelation among inputs, similar to the way HFS is used. The difference between our proposed approach and HFS is also highlighted in this chapter.

3.1 Context-dependent fuzzy sets

The main objective of a context-dependent fuzzy set is to model the influence of context on the meanings of linguistic terms represented by fuzzy sets. Thiele [124] first coined the term "context-dependent fuzzy set" in 1999. He defined a context-dependent fuzzy set as a fuzzy set whose values depend not only on the universe of discourse U but also on W, a non-empty set of possible worlds. Formally, according to Thiele:

> $\phi : W \times U \rightarrow [0, 1]$ is said to be a context-dependent fuzzy set (CDFS) on U with respect to W.

W can be thought of as a set of all possible contexts at which the meaning of a fuzzy set changes over. For example, the set of contexts W that affects the meaning of a vague concept *Tall* of a set of people may consist of, for instance, Russian, American, Asian, etc. Within a world $w \in W$, standard fuzzy set operations (i.e., fuzzy complements, fuzzy intersection and fuzzy unions), and fuzzy inference methods can be formulated in a straightforward manner. When quantifying over worlds, fuzzy inference is only correct under certain conditions. Interested readers can refer to Thiele's paper [123] for more details on these conditions. It is evident that Thiele's theory only gives a mathematical background of context-dependent fuzzy sets. Application of the concept necessitates an implementable representation of W. In this chapter, such a representation of W is proposed.

3.1.1 Context representation

Context is a set of facts or circumstances that affect the interpretation of a linguistic term. Each fact or circumstance describes the state of the environment, systems or users, or may represent higher level concepts such as functions, goals, plans, and so on. Clearly, the number of such facts is fixed and limited; and different constituent facts of the context may trigger contextual changes. This notion of context can naturally be thought of as a context representation as a collection of basic units comprising entities, acts and attributes that affect the meaning of a linguistic term. Such a representation

unit is called context attribute and the number of such units is always fixed. Each context attribute represents a certain fact or circumstance of the context model. Context attributes could be any type of variable, e.g., crisp or fuzzy variables.

Formally, let $C = \{C_1, C_2, ..., C_n\}$ be a nonempty finite set of variables, where C_i represents a context attribute i for context state C. Now, each context $w \in W$ is represented as a tuple of n attribute-values $w = \{c_1, c_2, ..., c_n\}$, where c_i represents the value of the attribute C_i, where $i \in \{1, 2, ..., n\}$. The context-dependent fuzzy set ϕ is now defined as

$$\phi : C_1 \times C_2 \times ... \times C_n \times U \rightarrow [0, 1] \tag{3.1.1.1}$$

Clearly, ϕ is a (usual) fuzzy set on $C_1 \times C_2 \times ... \times C_n \times U$. ϕ can be rewritten as follows

$$\phi = \int_{c_1 \in C_1} ... \int_{c_n \in C_n} \int_{x \in U} \mu(x, c_1, ..., c_n)/x/c_1/.../c_n \tag{3.1.1.2}$$

or

$$\phi = \int_{c \in C} \int_{x \in U} \mu(x, c)/x/c \tag{3.1.1.3}$$

The membership function of a context-dependent fuzzy set depends not only on its own universe of discourse U but also on the domain of its context attribute variables. Any change in any of the context attribute variables signals a change in the context. As an example of context-dependent fuzzy set, consider the linguistic variable AGE, a possible linguistic value for which is *old*. Typically, most people would tend not to judge their own age as old, therefore the fuzzy set *old* can be defined as context-dependent relative to the speaker's age. In this case, the set of context variables C contains only one context attribute, namely the variable AGE itself. The membership function of *old* can be defined as

$$\mu_{old} = triang \ \ (AGE + 5, AGE + 10, AGE + 15) \tag{3.1.1.4}$$

Of course, the same MF can be defined using different context modelling techniques. For example, using the context representation proposed by Gudwin and Gomide [53],

the fuzzy set *old* can be defined as a transformation function from a base set

$$\mu_{base} = triang(0, 0.5, 1) \tag{3.1.1.5}$$

$$\mu_{old} = \mu_{base} \times 10 + 5 + AGE \tag{3.1.1.6}$$

In this simplistic example, the two representations are equivalent in every aspect. However, in more sophisticated situations, e.g., where different shape functions to represent *old* in the case of $AGE > 35$ are desirable, finding a transformation function from the base set to the expected shape is not only much more problematic but also is much less interpretable than defining a function based only on the variable AGE itself. In general, compared with existing techniques, our context modelling is simpler and more intuitive while still achieving the expected level of accuracy.

3.1.2 Set-theoretic operations and fuzzy inference

Let A and B be two CDFSs and C_A, C_B be the sets of context attributes of A and B respectively. Let $C = C_A \cup C_B$ be the set of all context attributes used to describe the contexts of A and B. Since C_A and C_B are a collection of variables each of which represents specific contextual information, it is evident that C_A and C_B are representatives of the same larger context C. Within the same context C, any fuzzy operation on CDFS is guaranteed to be correct.

3.1.3 A context-dependent interpretation of linguistic variables

The definition of Zadeh's linguistic variables can be extended to allow the meanings of linguistic variables to change according to context. Since linguistic variables represent the meanings of a single concept, it is intuitively reasonable to assume that all linguistic terms of a linguistic variable will have the same context of perception. Hence, a linguistic variable can be characterised by a 5-tuple $\langle L, T(L), U, C, M \rangle$, where:

- L is the name of the linguistic variable

- $T(L)$ is the finite term set of L.
- U is the universe of discourse of base variable.
- C is the finite set of context attributes that form the context of the variable
- M is a semantic rule that associates each term $l \in T(L)$ with its meanings. $M(l)$ is the membership functions of a CDFS representing the meaning of l under the influence of context C

Consider the variable TEMPERATURE of a patient. The universe of discourse U is the interval $[0, 100]$. The linguistic variable for TEMPERATURE can be defined as

$$\langle TEMPERATURE, T(TEMPERATURE), U, C, M \rangle \tag{3.1.3.1}$$

where

$$T(TEMPERATURE) = \{normal, fever, hyperthermia, hypothermia\}$$

$$C = \{AGE, SEX\}.$$

$$Domain(AGE) = [0, 150]$$

$$Domain(SEX) = \{male, female\}.$$

The semantic rule M must define all MFs for each term such as

$$M(normal) = \mu_{normal}(temperature, age, sex),$$

$$M(fever) = \mu_{fever}(temperature, age, sex)$$

$$M(hyperthermia) = \mu_{hyperthermia}(temperature, age, sex)$$

$$M(hypothermia) = \mu_{hypothermia}(temperature, age, sex)$$

where $temperature \in U$, $age \in Domain(AGE)$ and $sex \in Domain(SEX)$.\

3.2 Context-dependent fuzzy system

Let $(X_1, X_2, ..., X_n)$ be a set of context-dependent linguistic variables. Let $(C_1, C_2, ..., C_n)$ be the corresponding set of context attributes for each of the variables. The combination of all context attributes is defined as $C = C_1 \cup C_2 \cup ... \cup C_n$. It is not difficult to extend the model structure of conventional fuzzy systems to allow for a context dependent fuzzy system as shown in Figure 3.1. In context-dependent fuzzy systems, the MFs are constructed at the time of evaluation when all contextual information are given. The fuzzification, fuzzy inference and defuzzification processes are identical in both types of fuzzy systems. The linguistic rule expression of the context-dependent fuzzy systems is also extended to make it more flexible. In particular, the linguistic rules are defined with the following structure:

IF X_1 is A_1 and ... and X_n is A_n (with weight w) THEN Y_1 is B_1 and ... and Y_m is B_m

Each rule is associated with a weight w which determines the relevancy of the rule in the context of evaluation. w is defined as a function depending on the context

$$w : C \rightarrow [0, 1] \tag{3.2.0.1}$$

$w = 1$ means that the rule is totally relevant in the context

$w = 0$ means that the rule is irrelevant in the context and can be ignored.

w affects the final firing strength of a i-th rule as follows:

$$firingstrength = firingstrength_{old} \star_T w \tag{3.2.0.2}$$

where $\star_T$ denotes a T-norm operator, which usually is a product operator.

Please note that the use of rule weight for either the antecedent or consequent part has also been studied in conventional fuzzy systems. In conventional systems, rule weights are defined as fixed numbers and can be equivalently replaced by modifying the MFs used in the antecedent and consequent. However, in context-dependent fuzzy systems, the use of rule weights allows for entirely different sets of rules to achieve

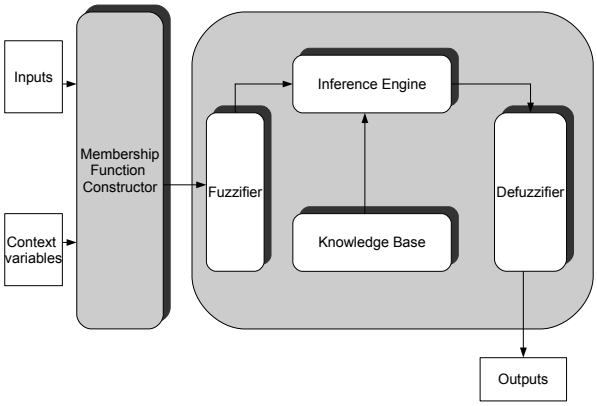


Figure 3.1: Structure of a context dependent fuzzy system

the same goal under different circumstances. For example, to drift the car around a corner, a driver might want to use a totally different set of rules from those under normal conditions. These rules may be counter-intuitive, which requires a different set of antecedents and consequent, rather than simple modification of existing membership functions. Apart from rule weights, the remaining structure of a context-dependent fuzzy system is identical to that of conventional fuzzy system's.

3.3 Interpretability and accuracy improvements in context-dependent fuzzy system

In this section, several examples are considered to illustrate how context-dependent fuzzy systems can achieve better interpretability and/or accuracy than standard fuzzy approaches.

3.3.1 Interpretability improvements

Let us consider the linguistic variable, i.e., Records, which is used to model the records of the runner athletes in the Olympic. It is obvious that the records of the athletes must be classified according to the race categories, i.e., distance, age and sex. In this case, the set of context variables C consists of $\{distance,\ sex,\ age\}$. If the linguistic values for Records are Excellent, Good and Poor, the MFs for each context such as

$$\mu_{Excellent,100m,M,18}(x) \quad \mu_{Good,100m,M,18}(x) \quad \mu_{Poor,100m,M,18}(x)$$
$$\mu_{Excellent,200m,M,around\ 20}(x) \quad \mu_{Good,200m,M,around\ 20}(x) \quad \mu_{Poor,200m,M,around\ 20}(x)$$
$$\mu_{Excellent,400m,F,19}(x) \quad \mu_{Good,400m,F,19}(x) \quad \mu_{Poor,400m,F,19}(x)$$
$$\ldots \quad \ldots \quad \ldots$$

must all be defined. Now, a simple fuzzy rule in a context dependent fuzzy system, e.g., "IF Records IS Excellent THEN Bonus IS High", can only be evaluated when provided with all the information about the context, namely distance, sex and age group of a particular runner. Thus, the actual MF of Records is changed according to the context. Of course, using standard fuzzy sets one can achieve the same purpose by introducing new linguistic terms, e.g., $Excellent_1$, $Excellent_2$, etc., each of which having the meaning of the "*Excellent*" record in a particular context, and explicitly including the context into the rule base, i.e., the same rule needs to be extended into a series of rules, e.g., IF Records IS $Excellent_1$ AND Sex Is Male AND Distance IS 100m AND AGE THEN Bonus IS High. However, introducing new linguistic terms causes the number of rules to rise significantly, which may make the system not only lose the capability of being interpretable by human beings but take much longer to process as well. If there are a large number of contexts, the context dependence cannot be modeled without losing accuracy using standard fuzzy sets. In the example, although the same accuracy can be achieved using standard fuzzy sets, the context-dependent fuzzy approach apparently provides a much better solution in terms of interpretability.

3.3.2 Accuracy improvements

The example in consideration is taken from the actual implementation of our fuzzy speed controller for the TORCS-based Simulated Car Racing Championship [78] (SCRC), which will be elaborated in Chapter 6. The TORCS-based SCRC provides a platform environment that allows researchers to develop their own computer-controlled car drivers (also called "bots") to race in solo or with other opponents in a number of tracks. SCRC effectuates the use of local information in a manner comparable to that of a driver in real life, e.g., knowledge of coming turns, current speed and current gear. Accordingly, the bot gives controlling commands the car just like a real driver, i.e., gear,

clutch, brake and gas. The speed controller implements his strategy using a series of simple rules such as "IF Speed is slow then accelerate, IF Speed is fast then brake, IF speed is moderate then keep neutral acceleration". Again, the meanings of the terms "slow", "moderate" and "fast" are dependent on the context. In this example, the use of standard and context-dependent fuzzy approaches to model these context-dependent terms as well as the model accuracy is investigated.

3.3.2.1 Context-dependent fuzzy approach

As before, a linguistic variable $Speed$ with three associated linguistic terms "slow", "moderate" and "fast" is defined to control the speed of the racing car when taking a turn of radius r ($r = [0, \infty]$, where ∞ means a straight segment). The domain of $Speed$ is $[0, 360]$ km/h. The current distance from the car to the incoming turn is $d < 100$. The idea is to adjust the speed of the car around a "moderate" value when taking a turn. Naturally there is no specific range of speed that suits every turn. In fact, an appropriate turning speed must be dependent on the radius r of the corner and the distance d from the car to the turn. ($C = \{r, d\}$). The MFs for "slow", "moderate" and "fast" can be defined as shown in Figure 3.2. The shapes and widths of the MFs are fixed regardless of the context. Only the position of the functions is changed. The parameters a, b, c, d and e completely define all three membership functions. Let $a - b = b - c = c - d = d - e = constant$, now only the parameter c is required to construct the membership functions of $Speed$ for each context. c is the fastest speed given an available braking distance d so that the car can reduce its speed from c down to the appropriate target speed specified by $maxSpeedAtRadius(r)$ when arriving at the apex point of the corner. Clearly, c depends on r and d. Note that when $d \to 0$, the car is getting closer to the turn. When $d = 0$, the value of c determines the maximum speed that allows the car to safely make a turn of radius r. c is defined as

$$c = maxSpeedAtRadius(r) + \alpha \times d \tag{3.3.2.1}$$

where $maxSpeedAtRadius(r)$ represents the maximum speed at which the car can safely

negotiate a turn for a given radius r.

In practice, coefficient α is dependent on many factors such as the average deceleration of the car, braking capacity, condition of the tyre, road surface, and so on. As a result, it is generally not possible to determine the exact value of α. However, a rough approximation of α can be derived using empirical methods. One such methods that our work has deployed is to record the distance required for the car to slow down from the initial to the predefined target speed on applying full brake. The relationship between the braking distance and the speed difference dv, given by the ratio $\alpha' = \frac{dv}{d}$, is recorded. The process is repeated for different combination of initial and target speed, and for different road surfaces. Subsequently, α is calculated as the average of all recorded α'.

$$\alpha = \frac{\sum_{i=1}^{n} \alpha'_i}{n} \tag{3.3.2.2}$$

The function $maxSpeedAtRadius(r)$ can also be learned empirically. A custom round track is built for this purpose (see Figure 3.3a) and a simple controller which drives the car in a circle at radius r around a chosen origin is created. The car speed is gradually increased, and the maximum speed at which the car is still able to stay in the circle is recorded. Data on the corresponding maximum speed at a number of different radius $(speed_1, r_1), ..., (speed_n, r_n)$ is collected. The relationship between speed and radius is obtained by applying the best-fit curve algorithm to the data set.

$$maxSpeedAtRadius(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 + a_5 r^5 \tag{3.3.2.3}$$

Finally, c can be calculated as:

$$c = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 + a_5 r^5 + \alpha \times d \tag{3.3.2.4}$$

where a_0, a_1, a_2, a_3, a_4, a_5, α are constants.

Given this definition of c, the remaining context dependent MFs can be easily derived

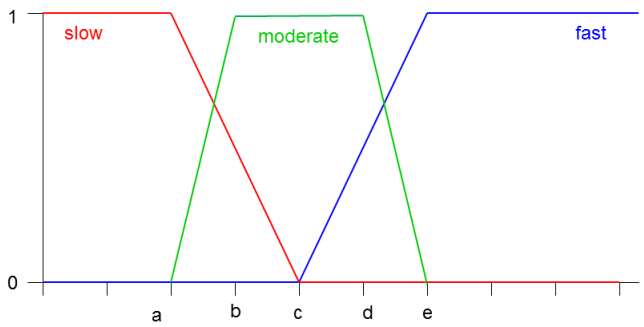
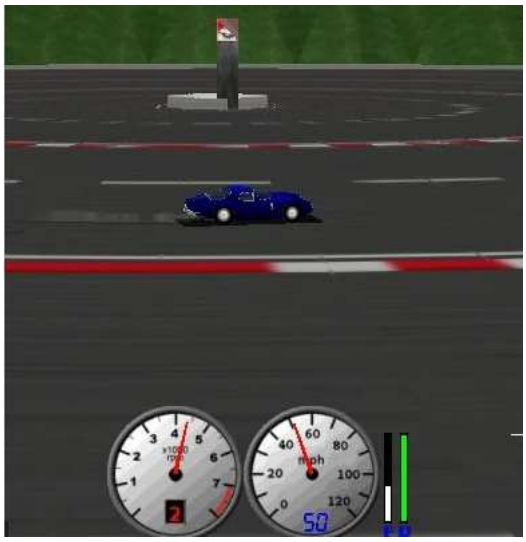


Figure 3.2: Representations of "slow", "moderate" and "fast" speed

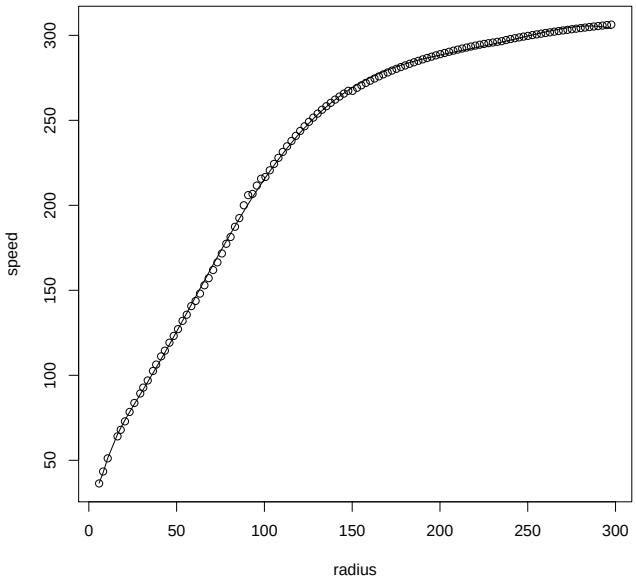
from the same context components, namely radius and distance. The fact that radius and distance could take any real value from zero to infinity implies that there is an unlimited number of contexts to be modelled. While existing fuzzy approaches in the literature can model context dependence by incorporating context into the RB, the fact that FRBS can only accommodate a fixed number of fuzzy rules indicates that it can only capture a limited number of contexts. By adding the context components into the representation of fuzzy sets, it is possible to automate the MF creation process that in turn allows any number of contexts to be modelled.

3.3.2.2 Mamdani fuzzy approach

It is clear that the dependence of the terms "slow", "moderate" and "fast" on the radius r and braking distance d cannot be directly translated into a single conventional fuzzy set. Instead, a collection of terms such as "$slow_1$",...,"$slow_n$","$moderate_1$",...,"$moderate_n$" and "$fast_1$",...,"$fast_n$", each corresponds to an interpretation of the terms in a specific context, must be created. Given memory capacity restrictions, only a finite number of such terms can be constructed while the number of potential contexts is infinite. As a result, the loss of accuracy using standard fuzzy sets is unavoidable in this case. Assume that the universe of the variable r can be divided into 10 partitions r_1,...,r_{10} where r_{10} represents a larger radius ($r > 300$) at which the car can safely negotiate the turn at its maximum speed. Assume that the variable d can be divided into 5 partitions



a



b

Figure 3.3: a) Custom track b) Speed at radius

$d_1, ..., d_5$ $(0 < d < 100)$. The number of fuzzy sets required to represent each of the terms "slow", "moderate" and "fast" in each of the contexts is $5 \times 10 = 50$. With this excessive number of rules and fuzzy sets, not only does the fuzzy system lose its interpretability but also its performance is severely affected.

3.3.2.3 Takagi-Sugeno-Kang fuzzy approach

Takagi-Sugeno-Kang fuzzy systems represent the output variables as polynomial functions of the input variables, therefore, the relationship between the variables $Speed$, r and d can be modelled in the consequent of the system. In theory, it is possible to find an equivalent Takagi-Sugeno-Kang fuzzy system to the context-dependent fuzzy system for this problem. However, the human interpretation on the action suggested by each rule is garbled in the Takagi-Sugeno-Kang system.

3.3.2.4 Comparison of different approaches

The previous example illustrates that conventional fuzzy approaches, due to its rigid partitioning of the input space, lose part of their capability to model the dependence between the input and some external variables in a human-understandable way. In the Mamdani fuzzy approach, to achieve the same result as context-dependent fuzzy systems, it is required to introduce new linguistic terms for the linguistic variables, new linguistic variables for each context attribute, and explicitly describes the context dependence into the RB. In other words, an equivalent conventional fuzzy system requires a greater number of variables, a great variety of linguistic terms and oversized RBs to achieve the same results, and thus the level of interpretability is not guaranteed. If the number of possible contexts is infinite, the Mamdani fuzzy approach not only suffers from an exponential growth in size due to the inflexibility of the fuzzy sets, but also lacks accuracy. In the Takagi-Sugeno-Kang fuzzy approach, the output is represented as polynomial functions of the input variables and therefore, in this example, it is possible to find an equivalent Takagi-Sugeno-Kang fuzzy system in terms of results. However, the interpretability is often very poor in TSK fuzzy approach. Only

a context-dependent fuzzy approach provides a solution with good interpretability as well as accuracy.

3.4 Context-dependent fuzzy system versus hierarchical fuzzy system

There is a similarity in the design between a CDFS and an HFS. Both methods require a prepossessing step before the final fuzzy evaluation is performed. In the case of CDFS, MFs of existing fuzzy inputs are created, while in HFS, new intermediate variables are created from other inputs and subsequently used as new inputs in the upper levels. Both methods can be used to overcome the curse of dimensionality inherent in standard fuzzy approaches when dealing with context-dependent problems. Consider the following example of a fuzzy medical system calculating the likelihood of heart attack of a patient. Among the inputs are the weight and height of the patient. The weight and height can be used to calculate the body mass index of the patient which is a more useful representation of weight and height. A new linguistic variable, called BMI, can be introduced to model the index. The HFS approach allows the BMI variable to be used as an additional input along with the original ones. Obviously, the MFs of the linguistic variable BMI are highly dependent on weight and height. This is very similar to the way a CDFS is defined, i.e., a MF of a CDFS can be created from its dependent context variables. However, despite the similarity, the fuzzy sets represented by BMI is not considered as CDFS because a CDFS is intended to represent the fuzzy set of an original input, which has a very specific range, meanings and must hold a generally independent value in the sense that it cannot be computed from other inputs. Intermediate variables created in HFS might or might not have a physical meaning, while in CDFS, the MFs always maintain the meanings of the original inputs. An equivalent context-dependent approach in this case could define the linguistic variable weight as a CDFS that is dependent on height and implicitly mapping the meaning of BMI into the CDFS weight. For example, $\mu_{obese}(weight, height) = \mu_{obese}(BMI)$, $\mu_{skinny}(weight, height) = \mu_{skinny}(BMI)$, and so on. The introduction of new inputs in

HFS will cause the number of rules to rise, and the rules themselves are more complex. Moreover, there is a clear distinction between a variable and a function. There is no restriction on how the MFs of a CDFSs can be defined from its context variables, while HFS always use fuzzy systems to create crisp intermediate variables. For example, when the context attributes are fuzzy variables, a CDFS ϕ can be expressed by a disjunction of conditions,

$$R_i : \text{IF } c_1 \text{ is } C_1 \text{ and ... and } c_n \text{ is } C_n \text{ THEN } \phi \text{ is } hlv_i$$

The process of defining ϕ can be thought of as an inference process of a standard fuzzy system whose inputs are $c_1, ..., c_n$ and the output is ϕ with the exclusion of the defuzzification step. Obviously, despite similarities in structure, there is clear difference in functionality between the two approaches. We are not stating that CDFS is necessary a better approach than HFS both in terms of transparency as well as interpretation. The question of when to use CDFS or HFS requires further study before any conclusion can be made. In summary, the main difference between the two approaches are:

1. CDFS will not introduce any additional rules nor intermediate variables. Therefore, CDFS usually require fewer parameters to train than HFS;
2. CDFS maintains the original meanings of inputs while intermediate variables of HFS might not have any physical meanings;
3. HFS creates new crisp variables while CDFS creates new fuzzy sets given the context variables.

3.5 Summary

This chapter presents a flexible and semantically-expressive context representation that can be used in various application scenarios. The model design structure of the conventional fuzzy system is extended to allow fuzzy terms to be interpreted according to the context within which they are used. The illustrative examples have demonstrated that the proposed approach can help improve both the interpretability and the accuracy

of FRBS in modelling context-dependent linguistic terms. The automatic derivation of MFs from the context components have also been demonstrated in this chapter. The capability of the proposed approach as a modelling tool for experts will be investigated and benchmarked against other techniques in a series of practical applications including the FUZZ-IEEE 2007 [57] and the 2007 IEEE CEC Simulated Car Racing Competition [128]. The successful applications of the proposed approach in the field of fuzzy modelling will also be studied later on in the thesis.

CHAPTER 4

Context-depedent fuzzy modelling and its application in point-to-point car racing simulations

This chapter presents the award winning implementations of our context-dependent fuzzy controllers in the FUZZ-IEEE 2007 [57] and the 2007 IEEE CEC Simulated Car Racing Competitions [128]. The former was the first competition to be associated with FUZZ-IEEE that acts as a testbed for fuzzy techniques. The latter was organised as part of the CEC and CIG conferences that aims at comparing car racing controller developed using evolutionary methodologies against other techniques. These competitions are based on point-to-point car racing simulations with fairly sophisticated physics models. To make the competitions more challenging, noise may be added to both observations and actions; both noisy and noiseless tracks are used to benchmark the performance of the controllers. The implementation of the context-dependent fuzzy controllers also demonstrate the use of mathematical functions to automate the derivation of MFs based on external context components, i.e., other input variables. No other fuzzy approach can implement the automation of the MF creation process.

4.1 Introduction

Car racing is a challenging problem, attracting public excitement, evident from the huge amount of capital invested in both practicing and spectating at physical car races. Developing a good car racing controller is also a very demanding undertaking which requires knowledge of the car behaviour in different environments and various forms of real-time planning, such as path planning and overtaking planning, etc. Computational intelligence techniques have been applied to physical car racing such as in the DARPRA Grand Challenge [138] or in the radio-controller toy car racing event that were run as a competition for IEEE Congress on Evolutionary Computation (CEC) for 2003, 2004 and 2005. These challenges have yet to see a controller good enough to be considered on a par with a competent human racer. On the other hand, simulations of the racing challenges are also intrinsically interesting and challenging. Togelius and Lucas [125–127] investigated various controller architectures and sensor input representations for simulated car racing. They found that controllers based on first-person sensory inputs and neural networks could be evolved to achieve better performance than humans over a number of racetracks, and display interesting behaviour when co-evolved with another car on the same track [82]. Various learning methods to develop controllers for car racing simulations or games are used by the researchers. Floreano et. al. [43] developed a first-person controller using a small part (5×5 pixels) of the visual field as an input to a neural network. The evolved controller successfully drove the car around the track. Chaperot [25] applied an advanced artificial neural network trained with Evolutionary Algorithms and a Back-propagation Algorithm in a motocross game purportedly better performance than any living player. Abbeel and Ng [1] trained a system using reinforcement learning in a flight simulator and drove a real radio-controlled car. Coulom [36] in his PhD thesis, presented an efficient path-optimisation algorithm that was used to train his K1999 car driver in the Robot Auto-Racing Simulator (RARS). K1999 won the 2000 and 2001 RARS formula one seasons.

With the aim of comparing the strengths and weaknesses of different methodologies, Simon Lucas [81] has run several simulated car racing competitions in the Fuzz-IEEE 2007, 2007 IEEE CEC and 2007 CIG Simulated Car Racing Competitions, in which our

submitted fuzzy controllers won first prize. In our implementation, the MFs of a term are defined according to the value of another input variable. This means that for every distinct set of input variables, the fuzzy controller has a different set of membership functions and therefore the inference process will also change. In other words, this is an implementation of a context-dependent fuzzy set where the set of context attributes contains the input variables. None of the existing fuzzy techniques can be used to model the dependence in meanings amongst the input variables. The non-stationary fuzzy sets that Garibaldi [48] proposed, can only model changes over time while probabilistic fuzzy sets can only model random alterations. As standard type-1 [142] and type-2 [89] fuzzy sets are always fixed even when the circumstance changes, they cannot model this kind of dependence. The use of linguistic modifiers or fuzzy optimisation methods often result in a permanent shift of membership functions while the changes of membership functions in this case are temporary. The sections below detail the implementation of our proposed context-dependent fuzzy controllers for the two competitions.

4.2 The Fuzz-IEEE 2007 car racing competition

This section describes the Fuzz-IEEE 2007 car racing competition in which our context-dependent fuzzy controller won first prize in both the noiseless and noisy tracks. The goal of the competition was to compare the performance of various hand-coded fuzzy controllers, fuzzy controllers tuned by co-evolution [52], neural networks trained using evolution, and using temporal difference learning [82], heuristic controller, non-stationary fuzzy controller or, context-dependent fuzzy controller, etc. Apart from the non-stationary and context-dependent fuzzy controller, all of the approaches involve the task of tuning the parameters for the underlying type-1 fuzzy controllers either by machine learning methods or by hand. There are many factors that may affect the performance of a fuzzy controller such as the numbers of rules, the format of each antecedent and consequent, the shape and details of each MF, the choice of defuzzification method, etc. As a result, it is very difficult to claim whether one particular approach is

superior to the others solely on the basis of the controller's performance without taking into account the differences in their design. In this section, an investigation into the modelling accuracy of the proposed context-dependent fuzzy approach as a stand alone method is presented. Two structurally identical fuzzy controllers are created: one type-1 and one context-dependent fuzzy controller. The design and parameters of the two controllers are almost identical. The only difference is in the representation of a single important membership function; each corresponding controller is used for the appropriate type of MF, i.e., context-dependent and standard type-1 MF. By using the same design, the performance difference between the two controllers is evidently due to the use of their respective types of fuzzy sets, i.e., context-dependent vs. standard type-1 fuzzy sets. The results have shown that context-dependent fuzzy sets can indeed improve the modelling accuracy of standard fuzzy sets.

4.2.1 General rules

The aim of the competition is to race a simulated car to a series of waypoints all within a unit square on a two-dimensional plane. The challenge is to find the best fuzzy controller at this task. However, there are several constraints to the current problem that add to the level of complexity:

- The car competes against another car.
- At any point in time, each car only knows the position of the current waypoint and the following two waypoints. These must be visited in order by the racing cars, but once any car has reached a waypoint, then the opponent car can skip this and go directly for the next waypoint.
- The waypoints are randomly distributed around a square area. Every track is different, and tests the general driving ability rather than how familiar a driver is to a particular track
- Only the first car to reach a waypoint scores a point for that waypoint.
- At each step, the position and status of the car are updated according to the re-

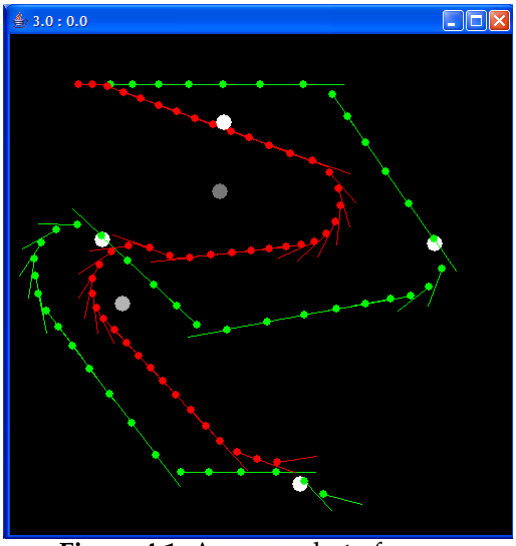


Figure 4.1: A screen-shot of a race

turned values of the controller. The car can jump over the target. Equivalently, the position of the car must precisely hit the waypoint in order to be counted as a hit.

- The performance of a controller is measured by the number of waypoints it can pass within a set time limit (500 time steps).
- Skidding and collisions are ignored. This allows for an entirely symmetric race. Each controller is fed with identical information, and the cars start at an identical position.
- Two types of tracks are used:
  - Noisy track: Gaussian noise is added to the observed position and velocity of the cars as well as to the positions of the waypoints; and
  - Noiseless track: no noise is added to the observed positions of the cars and the waypoints, or to the velocity of the cars.

Figure 4.1 shows an example run between two cars. A car is shown as a circle, with a line to indicate the current heading. In this example, the green car is faster and has a more aggressive driving strategy, which results in its out-performance over the red car.

4.2.2 Controller requirements

At each step, the controller is given the following data:

- The positions of the next three waypoints. The position is represented as a two-dimensional vector.
- The current state of the car, which consist of a velocity vector, a heading vector and a position vector.

Other information such as the radius of the car (a car is represented as a circle) and the radius of the waypoint are constant. The controller of each car is provided with the above inputs and is required to return the steering and acceleration commands that will be applied to the car at the next step. The steering and acceleration are restricted by pre-defined minimum and maximum values. Any values outside these boundaries will be truncated. The underlying physics is fairly simple, all motions are based on Newtonian mechanics of point masses.

4.2.3 Strategies and implementation of CDFS

There are two steps that a controller must consistently carry out

- First, choose a waypoint as the target out of the three visible waypoints.
- Head towards the chosen target without considering the other waypoints.

A simple and fast heuristic controller is used to estimate the number of steps required for each car to hit the first waypoint. If our car takes fewer steps to get to the first waypoint than the opponent car, then the first waypoint is chosen as the target, otherwise the second waypoint is selected. For brevity's sake, the details of the algorithm to choose the targets are not described in this work. From now on, it is assumed that the target has been selected and the aim of the controller is to find the best way to reach the target.

4.2.4 Strategies to reach the target

It is evident from our experiments that the car turns faster at higher speed and, as skidding is ignored, it would generally take less time to make a quick U-turn at high speed than to reverse. Therefore only forward motion is implemented. Although there is a large variety of situations that the car can be in, they can be categorised into two alternatives:

- The angle from the car to the target is so small that no steering is needed. The problem simplifies to that of finding the optimal acceleration for the car to hit the target in a straight-line path given the initial velocity (speed) u of the car and the distance to the target s. The optimal acceleration can be determined as follows:
  - The estimated number of step required for the car to hit the target can be calculated by rearranging the equation $v^2 = u^2 + 2as$ as follows:

$$v = u + at = \sqrt{u^2 + 2as} \tag{4.2.4.1}$$

$$t = \frac{\sqrt{u^2 + 2as} - u}{a} \tag{4.2.4.2}$$

    The minimum number of steps must be an integer and is achieved when the acceleration is at its maximum value. Therefore:

$$t_{min} = \left\lceil \frac{\sqrt{u^2 + 2a_{max}s} - u}{a_{max}} + 1 \right\rceil \tag{4.2.4.3}$$

    where a_{max} is the maximum possible acceleration.
  - The acceleration requires to hit the target in t_{min} steps is:

$$a = \frac{2 \times (s - u \times t_{min})}{t_{min}^2} \tag{4.2.4.4}$$

- The angle from the car to the target requires some steering by the car. The acceleration is set to its maximum value as it will make the car turn faster. From

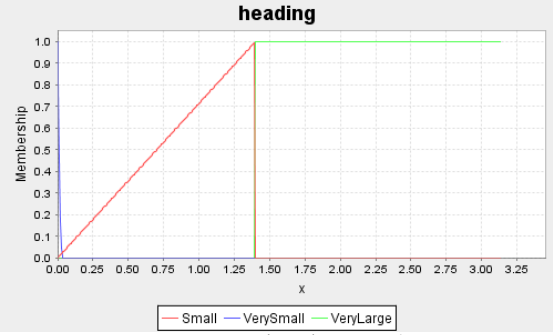


Figure 4.2: *heading* and θ

experiments, the steering output should be roughly proportional to $\frac{heading}{speed}$:

$$steering \approx heading \times 1/speed \tag{4.2.4.5}$$

where *heading* is the current angle from the car to the target as shown in Figure 4.2

No more steering is required when the current heading angle of the car is very small, or more precisely, when $|heading| \leq \theta$ (where θ is defined as in Figure 4.2). θ can be easily calculated given the distance to the target as follows:

$$\theta = arctan\left[(r_{car} + r_{waypoint})/s\right] \tag{4.2.4.6}$$

where r_{car} and $r_{waypoint}$ are the radius of the car and waypoint respectively.

There are situations in which the car tries to turn towards the target but ends up in a circular loop and will circle forever. It has been determined experimentally that after 15 steps, if the car is not in a the position to go straight, it is likely to be trapped in a loop condition. To avoid loop condition, 15 steps ahead is calculated and if the car is trapped, reverse the current steering ($steering_{new} = -steering_{old}$)

4.2.5 Controllers

Three different types of controllers are implemented using the same strategy described above. Each controller will compete with each other head-to-head and the one with the most wins is the winner. The three controllers are a type-1 fuzzy controller, a non-stationary fuzzy controller and a context-dependent fuzzy controller. See Figure 4.3 for

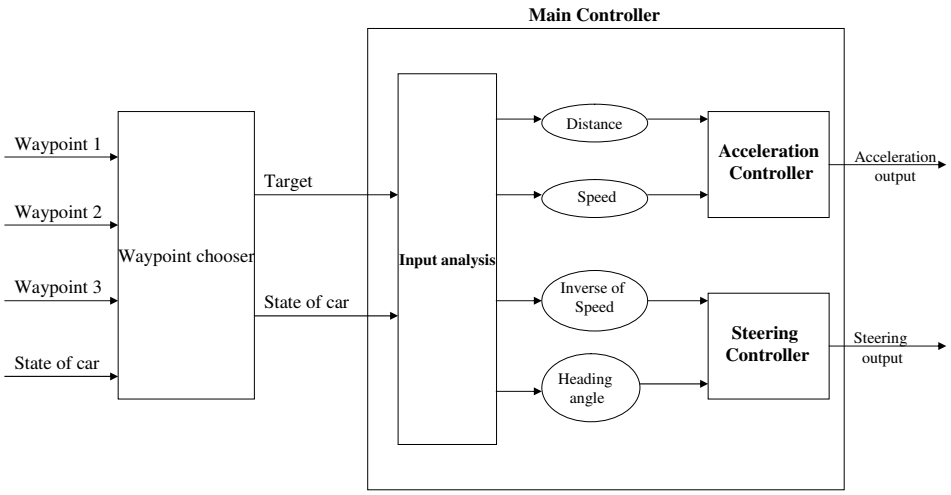


Figure 4.3: Structure of the controllers

an overview of the structure of the controllers.

4.2.5.1 Common details of the fuzzy controllers

As mentioned above, acceleration is set to the maximum when the car is turning and set to a specific value when no steering is required using a simple formula 4.2.4.4. The fuzzy system is used only to work out the steering angle of the car. Although the fuzzy system can only approximate the steering output in equation 4.2.4.5, it is surprising that the results achieved are superior to those using the formula directly, especially in noisy tracks. In this study, only the results of fuzzy controllers are analysed and discussed.

As shown in Figure 4.3, the structure of the three controllers is identical. The only difference lies in the use of different types of fuzzy sets for their respective fuzzy controllers. The input of the fuzzy systems are heading angle and inverse of speed, and the output is steering. All fuzzy systems use the same set of rules. Membership functions, parameters and the set of rules are chosen empirically. There is no training or optimisation involved. The rules are:

```
R1 : IF heading IS VerySmall
THEN steer IS VerySmall ;
R2 : IF heading IS VeryLarge
```

```
THEN steer IS Large;
R3 : IF heading IS Large
AND inverseSpeed IS Large
THEN steer IS VeryLarge;
```

To simplify the problem, only the case where $heading \geq 0$ are considered. In the case of $heading < 0$, the same fuzzy system can be used by negating the value of *heading* and steering output before and after the inference process, respectively. The second rule applies to the situation where the heading angle of the car is larger than the maximum possible angle that the car can turn, in which case the steering should be set to its maximum value. The third rule applies to the general situation in which steering is approximately equal to the product of heading and $1/speed$. The first rule will set the steering to 0 when the heading angle is very small, so that no more steering is needed. The definition of the term "*VerySmall*" encapsulates the meaning of that angle and the membership function of the term *VerySmall* is different for each fuzzy system. The defuzzication method used by the three fuzzy systems is LeftMax, which takes the left most maximum value of the output space. Three different types of membership functions are used: Gaussian, Triangular and Trapezoidal membership functions. The denotation of these membership functions can be formalised as follows:

- Gaussian membership function with centre c and standard deviation σ

$$gauss\ c\ \sigma = \mu(x, c, \sigma) \ = \ e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{4.2.5.1}$$

- Trapezoidal function:

$$trape\ a\ b\ c\ d \ = \ \mu(x, a, b, c, d) \ = \ \begin{cases} 0, & x < a,\ x > d \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c \leq x \leq d \end{cases} \tag{4.2.5.2}$$

- Triangular function:

$$trian\ a\ b\ c\ = \mu(x, a, b, c) \ = \ max\left(min\left(\frac{x-a}{b-a},\ \frac{c-x}{c-b}\right), 0\right) \tag{4.2.5.3}$$

FUZZIFY heading
TERM *VerySmall* := ?;
TERM *Large* := trian 0 1.4 1.41;
TERM *VeryLarge* := trape 1.38 1.4
3.14 3.14;
RANGE := (-3.14159 .. 3.14159);
END_FUZZIFY

FUZZIFY inverseSpeed
TERM *Large* := trape 0 87.267 100 100;
RANGE := (0 .. 100);
END_FUZZIFY

DEFUZZIFY steer
TERM *VerySmall* := gauss 0 0.01;
TERM *VeryLarge* := trian 8.6 8.9 12.8;
TERM *Small* := trian 0 122.173
122.173;
ACCU : MAX;
METHOD : LM;
DEFAULT := 0.0;
RANGE := (-122.1730 .. 122.173047);
END_DEFUZZIFY

Table 4.1: Details of membership functions

Table 4.1 shows the details of all variables and their membership functions. Figures 4.4, 4.5 and 4.6 show the graphical representations of the variables *heading*, *inverseSpeed* and *steer* respectively. In Figure 4.5, the scale of the figure is so small that the blue Gaussian membership function of the term '*VerySmall*' appears as a straight line. Figure 4.7 is the enlargement version for the membership functions of '*VerySmall*'.

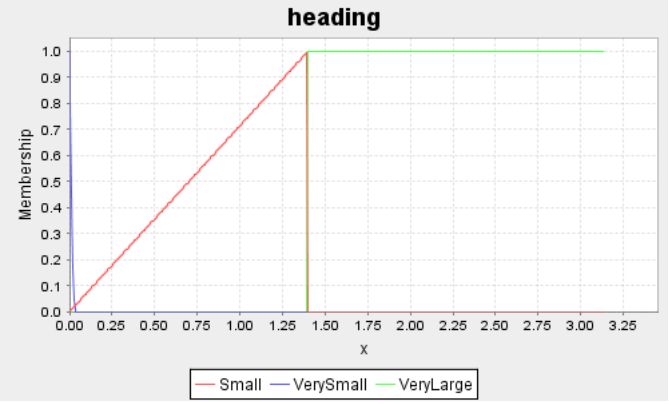


Figure 4.4: The underlying membership functions of fuzzy variable *'heading'* of the type-1 fuzzy controller

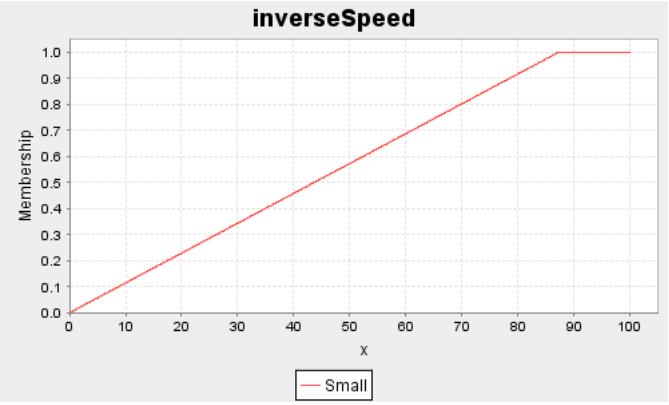


Figure 4.5: Membership functions of fuzzy variable *'inverseSpeed'*

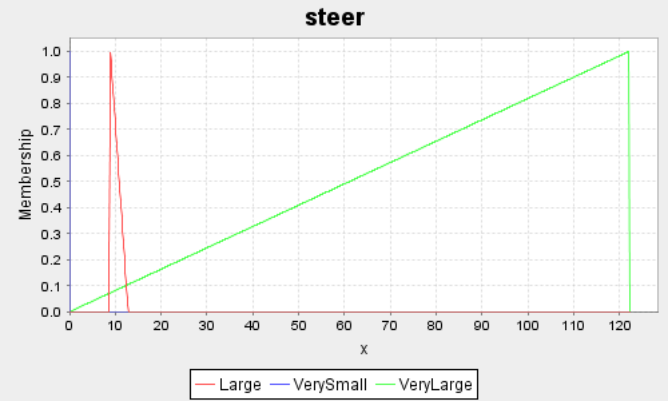


Figure 4.6: Membership functions of fuzzy variable *'steer'*

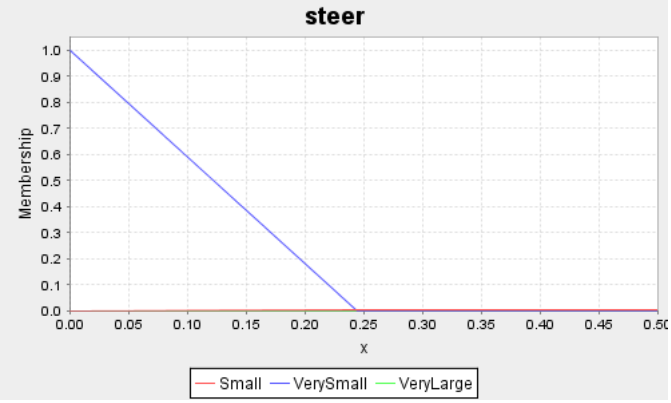


Figure 4.7: Membership functions of term *VerySmall* of fuzzy variable '*steer*'

4.2.5.2 Context-dependent fuzzy controller

The design of the context-dependent fuzzy controller is identical to that of the type-1 fuzzy controller. The only difference is in the representation of the membership function of the term "*VerySmall*" of the linguistic variable *heading*. In the type-1 fuzzy controller, the term "*VerySmall*" is defined as a Gaussian membership function with centre of 0 and width of 0.1

```
TERM VerySmall := gauss 0 0.1 ;
```

In the context-dependent fuzzy controller, the term "*VerySmall*" is defined as a context-dependent fuzzy set that represents a very small heading angle of the car to the target waypoint that no more steering is needed. As aforementioned, the heading angle could be considered very small when $|heading| \leq \theta$ where θ can be calculated given the distance to the target as in equation (4.2.4.6):

$$\theta = arctan\left[(r_{car} + r_{waypoint})/s\right] \tag{4.2.5.4}$$

where r_{car} and $r_{waypoint}$ are the radius of the car and waypoint respectively and s is the distance from the car to the target.

Therefore, we can define the context set $\mathcal{C}$ that influences the meaning of the term "VerySmall". In this case, the term only contains the variable *distance* that specifies the

current distance from the car to the target waypoint ($\mathcal{C} = \{distance\}$). The context-dependent fuzzy controller uses the trapezoidal membership function with context-dependent parameters to define the term "*VerySmall*"

```
TERM VerySmall := trape 0 0 θ θ;
```

θ can be considered a function of *distance* and therefore will change when the *distance* changes. The context-dependent fuzzy set of a term '*VerySmall*' of a linguistic variable *heading* is defined as:

$$\overline{VerySmall} = \int_{s \in \mathcal{C}} \int_{x \in X_{heading}} \mu_{\overline{VerySmall}}(c, x)/x/c. \tag{4.2.5.5}$$

where $\mu_{\overline{VerySmall}}(c, x)$ is the trapezoidal function with four parameters 0, 0, θ, θ

4.2.5.3 Non-stationary fuzzy controller

The non-stationary fuzzy sets was proposed by Garibaldi [48] to model the variations in the decisions which occur among a panel of experts, as well as in the decisions of an individual expert over time. In this type of fuzzy sets, variability is introduced into the membership functions through the use of random alterations. More details on non-stationary fuzzy sets and systems can be found in [47, 48, 98, 99]. Our implementation of non-stationary fuzzy controller used the same underlying MF as the type-1 fuzzy system. However, at each inference, both the centre and standard deviation are perturbed by a small Gaussian distributed random number.

4.2.6 Results & Discussion

The three controllers competed against each other in a round-robin manner, with 500 steps each match in a total of 500 matches in two rounds. Table 4.2 and 4.3 show the corresponding results of each pair in the case of noiseless and noisy tracks. The value at each cell represents the difference between the number of waypoints collected by the controller specified by the row compared to the controller specified by the column.

	Type-1	Non-stationary	Context-dependent
Type-1	-	-118	141
Non-stationary	236	-	221
Context-dependent	-115	-125	-

Table 4.2: Results for noiseless tracks

	Type-1	Non-stationary	Context-dependent
Type-1	-	-113	-391
Non-stationary	-13	-	-443
Context-dependent	285	248	-

Table 4.3: Results for noisy tracks

The non-stationary fuzzy controller performs better than type-1 and the context-dependent fuzzy controllers in the noiseless case, although the differences are insignificant. The type-1 fuzzy controller has roughly the same performance as the context-dependent fuzzy controller in the noiseless case. However, when random noise is added, the context-dependent fuzzy controller outperforms the other two. Overall, the context-dependent fuzzy controller has the best score, and scores higher than the other two in stochastic circumstances. For a detailed comparison of the context-dependent fuzzy controller with other competitors in the FUZZ-IEEE 2007 car racing competition, interested readers please refer to [81].

4.3 The 2007 IEEE CEC Simulated Car Racing Competition

This section describes the techniques that have been used by the winning entry of the 2007 IEEE Congress on Evolutionary Computation and the CIG2007 car racing competitions. The challenge is to race against an opponent around a track, trying to get as many points as possible. Previous research on similar problems are mostly based on either state-based or action-based controller architectures trained with machine learning techniques. In this work, a hybrid controller architecture is presented, which combines

the advantages of both existing architectures. The main component of the controller is designed as context-dependent fuzzy systems whose membership functions are adjustable according to the context. Finally, the competition results are given.

4.3.1 Review of existing approaches in car racing games

After a careful examination of existing control architectures, it appears that the state-based and action-based architectures are the two most frequently used in practice. The classical state-based approach consists in computing an optimal trajectory first. Then, a controller was built to track this trajectory. Examples of successful controllers of this type include the K1999 controllers of Coulom [36] in the RARS simulation, the berniw3 controller of Wymann [139] in the Open Racing Car Simulator (TORCS) [41], etc. This type of approach favours high-level reasoning capabilities, i.e., planning and often achieves excellent performance in the case when the track environment is completely known and fixed. The action-based controllers, on the other hand, favour reactivity as actions follow perceptions closely, almost like a reflex. Examples of successful controllers include the non-stationary fuzzy controller [57], and various machine learning controllers developed by Togelius and Lucas [3, 125, 126], etc. In general, state-based approaches often outperform action-based approaches if both can be applied. However, there are some limitations that hinder state-based architecture in dealing with real-time dynamic and uncertain environments. First, computing an optimal trajectory is often too costly to be performed online. Second, it cannot be assumed that the environment will stay the same as when the planning takes place. So, whenever an unexpected event happens, it is required to perform the expensive planning process again. And finally, the planning can only be applied when a forward model exists, which is able to predict the next state of the car given the current state and the selected actions. If such model is not readily available, it must be learned first. In this implementation, a hybrid control architecture is presented which combines both the high-level reasoning capabilities of the state-based approach and the reactivity of the action-based approach. The key component of the control architecture that was used to control the car is designed as a fuzzy controller, i.e., a controller based upon fuzzy logic, thus permitting

approximate reasoning and a human-like description of the car's reactive behaviour. This fuzzy component is implemented as a context-dependent fuzzy controller that allows the membership functions to be adjustable in accordance with the context.

4.3.2 The car racing model

The racing game model in this study is the same as that used for the simulated car racing competition at the 2007 IEEE Computational Intelligence and Games Symposium and the 2007 IEEE Congress on Evolutionary Computation. In this game, one or two cars race in an arena without walls. The objective of the game is to reach as many waypoints as possible within 500 time steps. These waypoints appear within a 400×400 pixel square area; the cars are not bounded by this area and can drive as far from the centre of the arena as they want. Waypoints are randomly positioned within a certain radius at the start of each trial, so that no two trials are identical; at any point two waypoints are visible to human or autonomous controllers, but only one waypoint (the current waypoint) can be "taken" by the first bypassing car. As soon as the current waypoint is reached, the waypoint counter is incremented for that car, the other visible waypoint (the next waypoint) becomes the current, and a new next waypoint is generated at a random position. See Figure 4.8 for a depiction of the game.

At any time step, the car controller is given a model of the sensors which include the current state of the car and its opponent, i.e., velocity, orientation, position, etc., as well as the position of the visible waypoints. The controller is then required to return one of the following actions: *back, back-right, back-left, neutral, left, right, forward, forward-left and forward-right*. The dynamics of the car are reasonably realistic, so acceleration and deceleration take time, and turning while travelling at high speed will cause considerable skidding. Turning on the spot is certainly not possible. Car- to-car collisions are possible and result in changes in both speed and directions of the cars. Despite the apparent simplicity of the rules, this game has plenty of hidden complexity that a controller needs to solve.

- Reach the current waypoint as fast as possible without overshooting it.

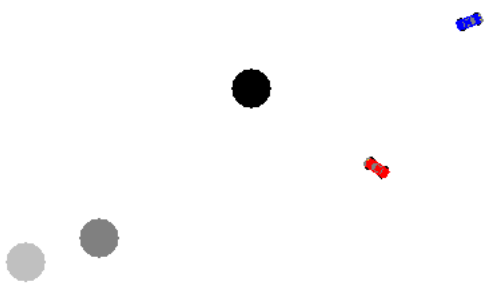


Figure 4.8: Two cars in the point-to-point racing game. The black circle represents the current waypoint, the dark gray circle the next waypoint, and the light gray circle the next waypoint after that

- Reach the current waypoint in such a way that the next waypoint can be reached quickly.
- Predict which car will reach the current waypoint first and take appropriate actions.
- Handle collisions effectively.
- Predict opponent's behaviour and take appropriate actions.

More details about the competitions can be gathered and its complete source code downloaded from the organiser's website at

http://julian.togelius.com/cig2007competition

and

http://julian.togelius.com/cec2007competition. Interested readers could refer to the paper of Togelius and others [128] for more in-depth details of the rules and the physical model used in this game.

4.3.3 Control Architecture

On the top layer, the control architecture consists of two main components:

- Waypoint chooser: uses a simple internal heuristic controller to estimate the number of steps required for each car to hit the first waypoint. If our car takes fewer steps to get to the first waypoint than the opponent car, then the first is selected

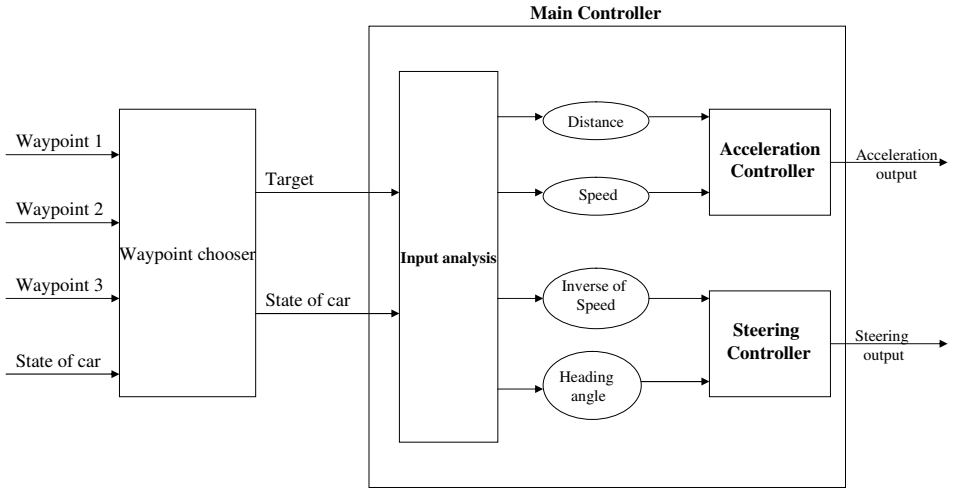


Figure 4.9: The control architecture of the racing car

as the target and the second waypoint is the next target, otherwise the second waypoint is selected as both the target and the next target because there are only two waypoints visible at a time.

- The main controller: takes the current states of the car and the output targets of the waypoint chooser to calculate the final action. In the case of a solo race, the waypoint chooser will always output the first waypoint visible as the current target that the car should drive to and the second visible waypoint as the next target that the car should prepare for before visiting the current target. The aim of the main controller is to drive the car to the target so that when the car hits the target, it has a heading angle generally towards the next target.

The focus of this study is on the main controller component only and the reader is referred to Figure 4.9 for a complete presentation of the architecture.

4.3.3.1 The main controller

The following scenarios are implemented:

- The target and the next target are the same: the car slows down on the approach and stops exactly at the target waypoint. The car will wait for the first waypoint

to be taken by the opponent in which case the second waypoint becomes active and so will be taken immediately. To resolve the potential problem when the opponent car runs into an infinite circular loop trying to get the first waypoint while our car is waiting at the second waypoint, we allow our car to wait for maximum of 100 steps at the second waypoint. If the waiting time is elapsed and the first waypoint is not taken, the car stops waiting and changes the target to the first waypoint.

- The target and the next target are different: The aim is to drive the car to the target waypoint so that when the car hits the target, it has a heading angle generally towards the next target.

For brevity's sake, only the second scenario is detailed. From now, it is assumed that the target and next target waypoints are different.

4.3.3.2 Control architecture of the main controller

The main controller consists of two main modules that are complementary: the path planner and the execution controller. In classical state-based approach, the path planner is used to compute an optimal path and the execution controller pilots the car to follow that path as closely as possible. In our approach, the execution controller is implemented as the key component of the system with the capability of a stand-alone action-based controller without referring to any pre-computed path. We opted for a fuzzy approach when implementing the execution controller as we hope to endow the controller with the human-like reactive capabilities required in an uncertain and partially known environment like the point-to-point racing game. The path planner module uses the execution controller to pre-compute different trajectories and returns the best option. This is a very expensive operation and thus is only used when the situation is simplified enough. The purpose of the path planner is to incorporate the power of machine computations to enhance the performance of the human-like solution given by the execution controller. Both of these modules will be detailed in the following sub-sections.

4.3.3.3 Execution controller

As aforementioned, the goal of the execution controller is to generate commands for the car so as to react to situations in real-time. The execution controller is designed as two fuzzy controllers, the speed controller and the steering controller, to control the acceleration and the steering of the car respectively. These two controllers take into account the positions of the two target way-points and the current state of the car in order to produce the desired speed and desired steering that the car should achieve at the next step. The outputs of the controllers are then combined into a single command that minimises the difference between the current state of the car and the desired state. Before detailing the execution controller and its main features, let us briefly recall what a fuzzy controller is.

Fuzzy controller:

A fuzzy controller is a control system based on the Zadeh's theory of fuzzy sets [142], thus permitting approximate reasoning and a human-like description of system's behaviour. A typical fuzzy controller consists of four components:

- A knowledge base: encodes the desired behaviour of the process. It is made up of:
  - Fuzzy sets: sets of membership functions associated with each input and output variable of the system.
  - Fuzzy rules: human-like rules of the form "IF condition THEN action".
- A fuzzifier: turns real input values into fuzzy values, e.g., a set of (fuzzy set label, membership degree) couples
- An inference engine: is the kernel of the fuzzy controller. It performs fuzzy inference from the input fuzzy sets based on the rules of the system and output fuzzy sets
- A defuzzifier: turns an output fuzzy set to an output real value.

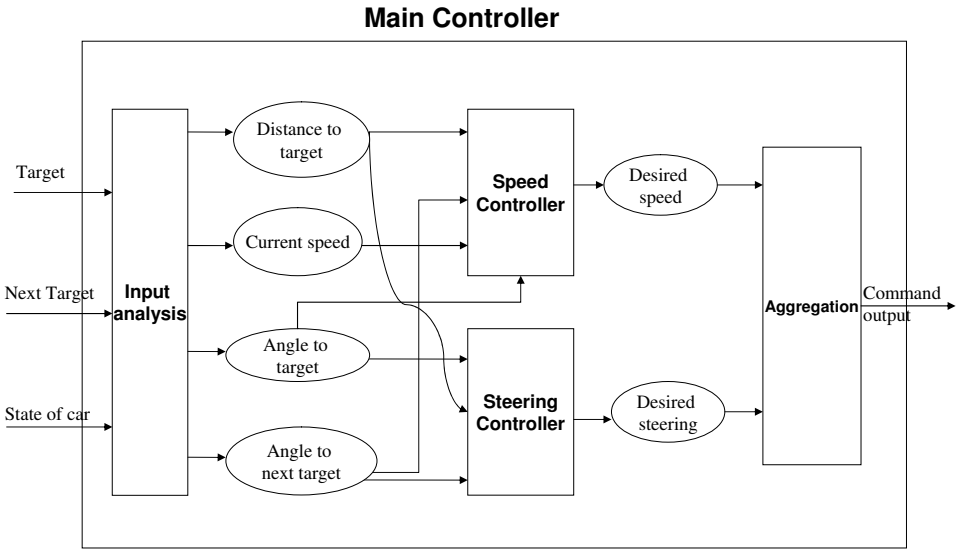


Figure 4.10: Design of execution controller

The interested reader is particularly referred to [128] for a summary tutorial and for more details.

Main features of execution controller:

The speed and steering controllers of the execution controller are designed as fuzzy controllers, as such, each of them includes four components as described above. However, they differ from the classical fuzzy controllers in that the membership functions of the fuzzy sets are not fixed but dependent on the values of other input variables. This can be easily implemented using context-dependent fuzzy sets with the context set consisting of the dependent input variables. For a comprehensive explanation of context-dependent fuzzy sets and the structure of context-dependent fuzzy systems, interested readers please refer to the previous chapter. Figure 4.10 shows the design structure of the execution controller.

Linguistic variables and fuzzy sets:

There are six linguistic variables: *distance*, *speed*, *heading angle*, *next angle*, *desired speed* and *desired steering*. The first four variables are inputs to the fuzzy controllers and the final two are the outputs. Five fuzzy sets labeled *VeryNear*, *Near*, *Medium*, *Far* and *VeryFar* are associated with the *distance* variable. Similarly, nine fuzzy sets labelled *BehindLeft*, *BehindRight*, *HardLeft*, *HardRight*, *LeftDir*, *RightDir*, *SmallLeft*, *SmallRight* and *StraightAhead* are associated with the current *heading angle* variable and the *next angle* variable. The *speed* variable of the input and the *desired speed* variable of the output associated with seven fuzzy sets: *VerySlow*, *Slow*, *Medium*, *Fast*, *VeryFast*, *BFast* and *Backward*. There are only three crisp sets associated with the desired steering output: 1 for *Left*, 0 for *Neutral* and -1 for *Right*. Figure 4.11 depicts the fuzzy sets associated with the *distance* variable. Figure 4.12 depicts the fuzzy sets associated with the *heading angle* and the *next angle* variables. The fuzzy sets of the *speed* variable and the *desired speed* variable are shown in Figure 4.13. The *desired steering* output are drawn in Figure 4.14. All the fuzzy sets are chosen empirically and at no time has any training or parameter tuning taken place in the implementation of the controller. All the linguistic variables with their fuzzy sets used in the speed controller are exactly as shown in the figures. In the steering controller, the fuzzy sets *LeftDir*, *RightDir*, *SmallLeft*, *SmallRight* and *StraightAhead* of the current heading angle variable are implemented as context-dependent fuzzy sets whose MFs are defined according to the distance variable. Let θ be a function of distance as follows:

$$\theta(s) = \frac{\left|\arctan\left(\frac{waypointIntersectionRadius}{s}\right)\right|}{2} \tag{4.3.3.1}$$

where s is the value of the *distance* variable and *waypointIntersectionRadius* is a constant. Let denote the trapezoidal membership functions as follows:

$$trape(a,b,c,d)(x) = \mu_{trape}(x,a,b,c,d) = \begin{cases} 0, & x < a,\ x > d \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c \leq x \leq d \end{cases} \quad (4.3.3.2)$$

The MFs of the context-dependent fuzzy sets of the *heading angle* variable are defined as follows:

- *StraightAhead: trape*($-2 \times \theta$,$-\theta$,θ,$2 \times \theta$)
- *SmallLeft: trape*($-5 \times \theta$,$-3 \times \theta$,$-2 \times \theta$,$-\theta$)
- *SmallRight: trape*(θ,$2 \times \theta$,$3 \times \theta$,$5 \times \theta$)
- *LeftDir: trape*($-\pi$,$-\pi$,$5 \times \theta$,0)
- *RightDir: trape*(0,$5 \times \theta$,π,π)

Please refer to Figure 4.15 for more details.

The speed controller:

The inputs of the speed controller are *distance, speed, heading angle* and *next angle* variables. The output is the *desired speed*. The controller applies Mamdani's inference. There are ten rules implementing the following strategy:

- The car goes forward most of the time (positive speed). It should only try to reverse in the circumstance that the car is moving at a slow speed away from the target and the distance to the target is not far away.
- Once going backward (negative speed), the car will maintain its direction until one of the following alternatives happens
  - The target is in front of the car at a very far distance
  - The target is in front of the car at a reasonably far distance and the current backward speed is slow

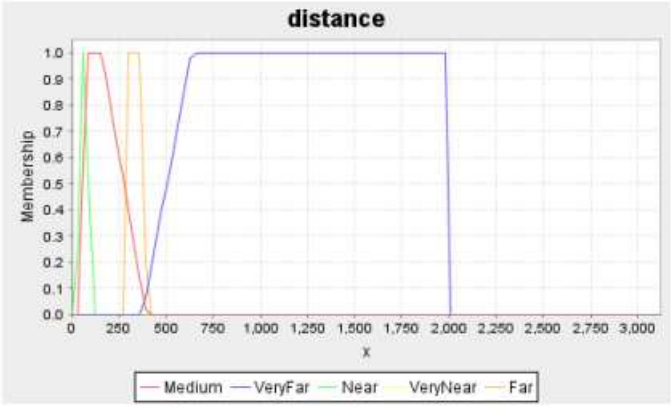


Figure 4.11: Five fuzzy sets associated with the *distance* variable

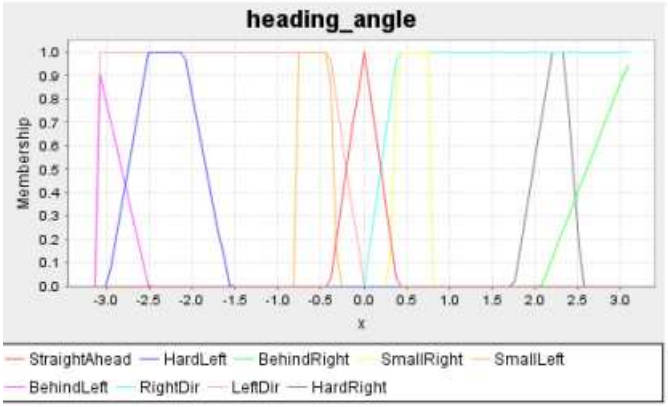


Figure 4.12: The nine fuzzy sets associated with the *heading angle* and *next angle* variables

- The speed of the car is always adjusted to be proportional to the distance to the target. The further the distance, the faster the speed and vice versa.

The steering controller:

The inputs of the steering controller are *distance, heading angle* and *next angle* variables. The output is the *desired steering*. The controller applies Takagi-Sugeno's fuzzy inference with nine rules. The steering controller always turns the car towards the target until the angle towards the target is within a small tolerance defined by the fuzzy set *StraightAhead*, in which case, the following rules apply

- If the distance to the target is too near or the car is at a small angle away from the next target, keep neutral steering.

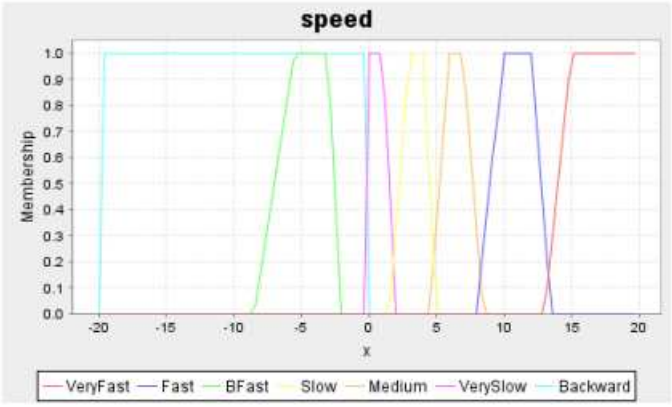


Figure 4.13: Seven fuzzy sets associated with the input variable *speed* and output *desired speed*

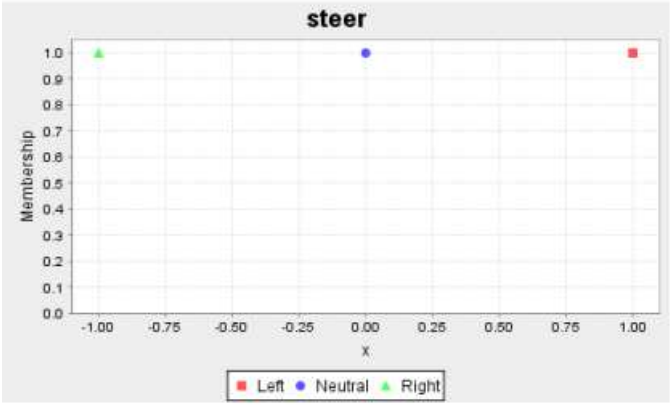


Figure 4.14: Three fuzzy sets associated with the *desired steering* output

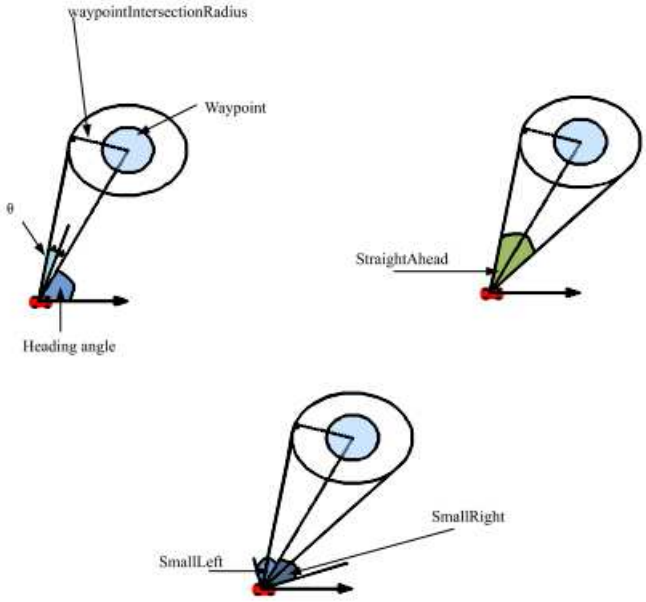


Figure 4.15: The meaning of θ and the fuzzy sets of the *heading* variable

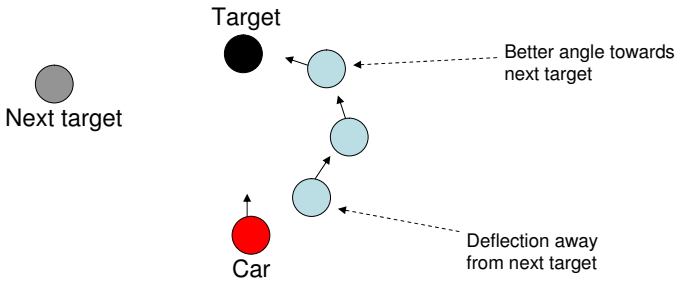


Figure 4.16: Steering controller

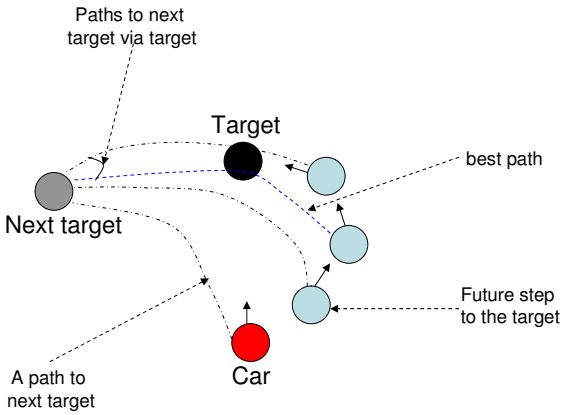


Figure 4.17: Best trajectory to the next target

- Otherwise, the car will deflect itself from the next target by a very small angle. See Figure 4.16 for an example. This will set up the car with a better angle towards the next target in the future when the car turns itself back to the target.

4.3.3.4 The path planner

The *execution controller* described above guarantees to find at least a solution in all situations, i.e., a path that visits both the target and the next target waypoints. The solution also ensures that the car reaches the target waypoint in such a way that the next waypoint can be reached quickly. The speed of the car is reasonably controlled along the path, i.e., increase the speed when the distance is far and smoothly slow down on the approach to avoid overshooting. However, the solution given by the execution con-

troller is far from optimal. From observation, it is easily found out that there are situations where the car slows down too much on the approach without realizing that it can go quicker if it follows the next target directly as the path to the next target will pass through the current target waypoint. To overcome this problem, a *path planner,* which basically tries different trajectories to the next target waypoint directly and takes only the first path that visits the target waypoint (see figure 4.17), is introduced. The *path planner* is called on when the car is at a specific distance close to the target. After the path is computed, all actions for the car to follow that path are stored. Subsequent calls to the controller will return the pre-computed actions corresponding to the current state of the car. If any collisions happen, the path planner is called again to compute a new path. The following steps are performed by the path planner:

Algorithm 4.1 Path Planner

Step 1 : Pre-calculate the path to the target
Step 2 : Store all the car's configurations, i.e., position, speed, angle, etc., and the corresponding control commands to follow the path
Step 3: Take the next configuration of the car on the predicted path and plan a new path to the next target directly. If the current target waypoint is found on the path, stop the planning and return all the corresponding control commands from the current configuration of the car to when the car hits the target. If it is obvious that the current path will not visit the current target, i.e., the car is heading away from the current target, repeat step 3 until a solution is found.

4.3.4 Experiments & Results

How good a controller is is measured by how many waypoints it can pass within a set time limit of 500 time steps. The number of waypoints collected is the score of the car after a race. The average score of a controller after 500 races is the final score for that controller. The higher the final score, the better the controller. The competition was organised as two rounds:

- In the first round, each controller was required to race on its own, versus a weak controller and versus an intermediate controller. The weak and intermediate controllers were provided by the organiser for example purposes only. The average of the final scores of these races were used to rank the submitted controllers. Only the top four controllers went to the next round. Table 4.4 lists the Competition

Score of the final submitted versions of all the controllers in the competition.

- In the final round, the top four controllers competed head-to-head in a round-robin manner. The controller having the most wins was the winner of the competition. Table 4.5 lists the names, controller representations and training methods of the top four controllers with highest Competition Score. Table 4.6 lists the solo scores of the top four controllers. And finally, table 4.7 lists the results of running the four controllers with highest Competition Score against each other.

  As can be seen, our controller (Ho & Garibaldi) performed consistently well in both solo racing and in direct competitions. The other controllers in the top four had variable performances in the different types of races. Surprisingly, the second top-scoring controller in terms of Competition Score, Tomoharu's, was roundly beaten by a large margin by the other finalists in head-to-head racing. After careful examination, it seems that this controller was more specialized towards beating the heuristic controllers used for the scoring function, at the expense of general racing skills. The Binatix, Inc. controller had a very good solo score but has not shown any significant difference in terms of performance compared to the controller of Chin Hong in a direct competition, even though the competition between the two has been run 3000 times.

Surprisingly, given the very similar scores, there is a quite a bit of behavioural variation even between the top four controllers. Some controllers rely on stopping and reversing direction when a waypoint is behind them, whereas others keep a fairly constant speed. Some controllers, e.g., the Binatix, Inc and Chin Hong's controllers, drive backwards only. Despite their counter-intuitive driving styles, these controllers are among the best controllers in the competition. Interested readers are particularly referred to [128] for a comprehensive comparison of all the controllers in this competition.

4.3.5 Discussion

The focus of this work is the main controller that actually pilots the car to the target with regard to the next target waypoint. The main controller does not follow the clas-

Controller	Competition score
Ho & Garibaldi	20.3
Tomoharu	19.5
Chin Hiong	19.1
Binatix, Inc.	19
Phil Hingston et al.	18.7
Bob MacCallum	16.9
Thomas Haferlach	16.5
Aravind Gowrisankar	15.4
Pete Burrow	15.2
Matt Simmerson	14
Example controllers (Julian Togelius)	12.6

Table 4.4: Competition score

Competitors	Models	Training methods
Ho & Jon	CD fuzzy systems	-
Tomoharu	Neural networks	GA, reinforcement learning
Tan Chin Hong	Magnetic force field systems	Force coefficients tuned by a GA
Binatix, Inc.	Feed-forward neural net	Temporal difference learning

Table 4.5: Top four competitors and their methods

Controllers	Score
Ho & Jon	24.4
Binatix, Inc.	21.03
Tan Chin Hong	20.07
Tomoharu	19.2

Table 4.6: Solo score

	Tomoharu	Chin Hiong	Binatix, Inc.
Ho & Jon vs	17.6/11.2	19.2/19.1	18.3/17.5
Tomoharu vs	-	13.0/18.8	13.7/18.5
Chin Hiong vs		-	19.4/19.4
Binatix vs			-

Table 4.7: Head-to-head competition results

sical approach of planning the optimal path first and then following the path as closely as possible, instead it was built upon an underlying execution controller that is able to react in real-time based on the current state of the car without referring to any pre-computed path. The execution controller differs from the classical fuzzy controllers in that it utilizes the idea of context-dependent fuzzy sets rather than the standard type-1 fuzzy sets, thus allowing more flexible fuzzy controllers that can adapt to different contexts. The purpose of the execution controller is to quickly give a generally good reaction in all situations without performing any expensive high-level reasoning operations, e.g., planning. Only in a few particular situations where path planning could be quickly performed do we pre-compute the optimal trajectory and follow it as in the classical approach. This approach is a balance between the reactivity and accuracy features of the existing approaches. There are additional advantages of this approach

- The number of trainings is dramatically reduced, thus making this approach more suitable to a sophisticated physical model, e.g., the real-world model, where the training process is very expensive and time consuming. In fact, we do not use any training in this study at all. However, we believe that by applying a reasonable amount of machine learning techniques we could tune-up the membership functions of the fuzzy controllers to achieve even better performance.

- The fuzzy approach could avoid some of the unexpected behaviour or counter-intuitive driving style that could be found in evolved controllers. For example, the three best evolved controllers of the CEC 2007 simulated car racing competition all exclusively drove backwards.

- The fuzzy model in this study differs from the classical fuzzy approach in that the membership functions of the variables are not fixed but dependent on some variables. This will hopefully enhance the adaptability of the controllers in more diversified circumstances.

It is hoped that the techniques used in developing the main controller especially the fuzzy approach that applied in the execution controller module, are relevant to more complex car racing simulations as well as to real-world car driving applications.

4.4 Summary

This chapter investigates the expressive capability of the proposed context-dependent fuzzy approach as a stand-alone method compared to conventional fuzzy approaches in two practical applications.

- The FUZZ-IEEE 2007 car racing competition [57]: a simple point-to-point car racing game in which multiple car agents compete with each other for waypoints in a racing field. The racing problem is fairly simple and ignores skidding or collision. To make a fair comparison and to ensure that the difference in performance between the controllers are not due to exogenous factors such as rule base, inference process and so on, a standard type-1 FRBS is developed with identical structure and rule base. The only difference lies in the representation of one single MF that is defined using context dependent fuzzy set in the context-dependent FRBS.

- The 2007 IEEE CEC Simulated Car Racing Competition [128]: a similar but more sophisticated point-to-point car racing competition which was held as part of the CIG 2007 and CEC 2007 conferences. With the introduction of skidding and car-to-car collisions as well as much more sophisticated underlying physics, the car controller needs not only to get as quickly as possible to the next waypoint, but also to anticipate other opponent's actions and plan a good response strategy. In this application, our controller competed against other strong competitors from both the fuzzy and computational intelligence communities.

Our hand-tuned context-dependent fuzzy controllers won first prize in both competitions in solo and dual races, in both noisy and noiseless tracks. Moreover, in the FuzzIEEE competition, our proposed fuzzy controller also outperformed the nearly identical standard type-1 fuzzy controller despite our best effort to tune its MFs. This has proven the modelling capability of context-dependent fuzzy sets. The two applications have also illustrated how experts can automate the creation of MFs using the context components to represent an unlimited number of contexts.

CHAPTER 5

Context dependent fuzzy systems with application to time-series prediction

5.1 Introduction

This chapter proposes an alternative evolutionary process for designing and tuning Takagi-Sugeno-Kang (TSK) based fuzzy models for approximation, classification and time-series prediction tasks. The proposed approach is different from existing methods in several distinctive ways. First, the underlying fuzzy system is a so-called context-dependent fuzzy system (CDFS) because it has the capability to adjust its MFs depending on the context. The flexibility of the MFs is inspired by the ability of humans to adapt the behavioural rules to a dynamic changing environment without learning the rules from scratch. The construction of such a system is achieved via two steps. First, a standard frame of reference type-1 fuzzy system which models universal knowledge that can be reused in many different contexts is built. The design of the system is influenced by the information granulation (IG)-based fuzzy model as used by various authors. As an example, the K-means clustering algorithm [84] is used to get the information granules (centres of clusters) which act as both the initial location of apexes of the MFs and the prototypes of polynomial functions appearing in the consequent

parts of the fuzzy rules. After the construction of the type-1 fuzzy system, the next step is to model the influence of context on the understanding of fuzzy terms. In other words, we need to define the context and the context adaptation process in which the MFs are modified so as to improve accuracy of the fuzzy system. With the aid of a K-means clustering algorithm, we can extract the features of the inputs and assign them into several groups. The criterion for a set of inputs to belong to a specific group forms the context that modifies the MFs of the system. In each context, the process of finding a transformation matrix used to scale the MFs to the situation is called the context adaptation process.

Both the parametric optimization of the type-1 fuzzy system and the tuning of the contextual transformation matrix presented in this paper are novel and efficient in their own right. The optimization techniques used for tuning the type-1 system is a combination of GA, memetic algorithm (MA) [92] and least-squares methods (LSMs). In particular, the MFs of the premises are explored in different search trajectories through the search space using the island model parallel genetic algorithm (IMPGA) [94], while the parameters of the consequent parts are solved in a robust manner using an improved QR Householder LSM [45]. To speed up convergence, we implemented a space search MA, which is one of the recent growing areas of research in evolutionary computation, as one of the search trajectories of the IMPGA. The MA that was used in our study is a combination of a local search and a GA, which guides the search to the promising region, and, hence accelerates convergence to the global optimum. To the best of our knowledge, the combination of IMPGA, MA and QR Householder LSM for parametric optimization of type-1 fuzzy systems has not been reported previously. The performance of such a combination of methods on the well-known Mackey-Glass time series benchmark [83] shows a significant improvement in accuracy with fewer rules compared to existing approaches in the literature. With the introduction of the CDFS, the tuning of the contextual transformation matrix is performed via an iterative algorithm based on the aforementioned MA. The context tuning algorithm is not only simple and fast but improves the performance of the CDFS beyond that of the optimised type-1 fuzzy system as well.

The content of this chapter is organised as follows. Section II details the structure of the type-1 fuzzy system, along with its parametric optimisation. The context representation and context adaptation process are discussed in Section III. The experiments and the results of this chapter are detailed and discussed in Section IV, while the conclusion is drawn in Section V of this chapter.

5.2 Design and Optimization of the Type-1 Fuzzy System

In this section, the design of the reference type-1 fuzzy system and the optimization technique used to tune the parameters of the system are described. The optimization of the fuzzy system is carried out by considering the parametric optimization of the premise and consequent parts. This section inspects the optimisation of the premise parts by means of a combination of the IMPGA and MA, and the optimisation of the consequent parts by means of an improved QR Householder least squares method.

5.2.1 Structure of the Information Granulation-Based Fuzzy Inference System

It is a well-established fact that the performance of intelligence models highly depends on structure and learning algorithm. In recent years, the use of Radial Basis Function Neural Networks [14] (RBFNNs) with a clustering algorithm is an effective combination which has been proven in many practical applications. The RBFNN is conceptually a very simple and yet intrinsically powerful network structure, i.e., it is an artificial neural network that uses radial basis functions (RBFs) as activation functions. RBFNNs are characterized by a transfer function in the hidden unit layer having radial symmetry with respect to a centre [129]. Therefore, clustering algorithm is naturally fit for RBFNNs. Every cluster has a centre, which can be chosen as the centre of a new RBF. However, the use of clustering algorithms can sometimes lead to a local optimum solution, depending on the inital locations of cluster centres. RBFNNs are universal approximators and thus best suited for function approximation problems. RBFNN-based approaches achieved fairly good prediction accuracy when testing on

the Mackey-Glass time-series benchmark [5, 55, 119, 148].

In [69], it has been shown that RBFNN and a simplified class of fuzzy systems are functionally equivalent under some mild conditions. This functional equivalence has made it possible to combine the features of these two systems, which has been developed into a powerful type of neurofuzzy systems [70]. In [90], a fuzzy radial basis function neural network (FRBFNN), which integrates the principles of a RBFNN and fuzzy sets, has been developed. The advantage of FRBFNN is that it does not suffer from the curse of dimensionality in comparison with other networks based on grid portioning. However, the accuracy of model is not better due to use of zero-order polynomial type in the computation of the consequent parts. In [30, 31], extended versions of the conventional FRBFNN, the IG-based FRBFNN, with higher order polynomial functions used in the consequent parts, are considered, and have shown significant improvement in prediction accuracy when tested on the Mackey-Glass time-series benchmark. To the best of our knowledge, the IG-based FRBFNN presented in these works are one of the state-of-the-art RBFNN-based optimization methods with highest prediction accuracy for the Mackey-Glass problems.

The use of information granule technology (clustering algorithms) also plays an important role in the learning of RBF-based systems. Clustering algorithms are usually used to extract information granule from data, e.g., K-means [9] and fuzzy c-means [15, 131] clustering are used to get information granule (centres of clusters) for generating fuzzy sets while Mountain clustering [140] and subtractive clustering [109] are used for automatic generations of fuzzy rules. However, as previously mentioned, clustering algorithms can sometimes lead to local optimum traps, and therefore in some applications [59, 60, 95], clustering algorithms are only used to create initial solution that will be improved by some other optimisation methods.

The structure of our fuzzy system is inspired by the IG-based TSK fuzzy model proposed by Oh et al. [95]. The K-means clustering algorithm is used to extract information granules that are the centers of clusters which help determine the initial values of the premise part and the prototype of consequent polynomials of the fuzzy rules. Given the set of data $U = \{x_1, ..., x_n; y\}$, where $x_k = [x_{1k}, ..., x_{mk}]^T$ and $y = [y_1, ..., y_m]^T$,

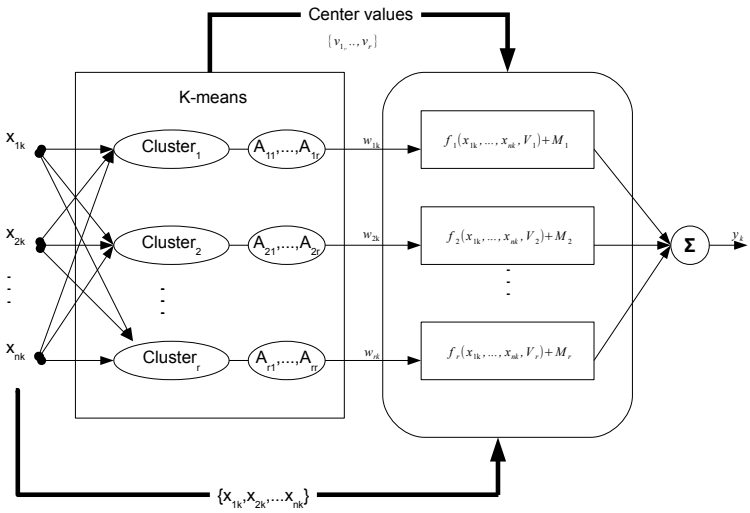


Figure 5.1: Architecture and initial parameters of the IG-based TSK system

where m is the number of data and n is the number of input variables, the structure and initial values of the premise and consequent part are constructed using the algorithm 5.1. Figure 5.1 shows a graphical illustration of the structure and initial parameters of the premise and consequent parts of the system. The calculations of the numeric output of the model, based on the activation levels of the rules, can be completed using the weighted sum defuzzification method [76]

$$y^* = \sum_{i=1}^{r} w_i y_i \tag{5.2.1.4}$$

The activation level w_i of rule i is calculated as $w_i = \mu_{A_{i1}}(x_1) \bullet ... \bullet \mu_{A_{in}}(x_n)$, where $\bullet$ is a t-norm. The two most commonly used t-norm operators are the minimum and the multiplication functions. In our experience, multiplication function usually achieve better accuracy than minimum function. However, using a multiplication function can sometimes lead to "underflow" activation levels whose values are smaller than the smallest representable floating point number and usually replaced by zeroes automatically. To minimize the effect of underflow conditions, a simple and effective method is to scale the activation level w_i by a large value N and, at the same time, scale the consequence parameters by the inverse of N.

Algorithm 5.1 Parameter initialization

[Step 1] Arrange U into a matrix of $m \times (n+1)$ containing the pair of input and output data in each row. A row X_k is defined as $X_k = [x_k; y_k] = \{x_{1k}, ..., x_{nk}; y_k\}$

[Step 2] Use the K-means algorithm to group the data set X_k into r clusters (r is also the number of fuzzy rules) and calculate the centre $\mathbf{v}_i$ of each cluster. $\mathbf{v}_i = \{V_i; M_i\} = \{V_{i1}, ..., V_{in}; M_i\}$ where n is the number of inputs, V_i and M_i are the centre values corresponding to the input and output vectors respectively.

[Step 3] The i^{th} rule of the fuzzy system, denoted as R^i, can be constructed as an extended form of TSK fuzzy model:

$$R^i: \quad \text{IF} \quad \overbrace{\underbrace{(x_1 \text{ is } A_{i1})}_{Antecedent} \quad AND \ ... \ AND \ (x_n \text{ is } A_{in})}^{Premise}$$

$$\text{THEN} \quad \overbrace{y_i - M_i = f_i(x_1, ..., x_n, V_i)}^{Consequent}$$

The consequent polynomial of the fuzzy rule can be considered one type of the following polynomials.

- Type 1. Zero-order polynomial (constant type):

$$f_i = a_{i0} \tag{5.2.1.1}$$

- Type 2. First-order polynomial (linear type):

$$f_i = a_{i0} + a_{i1}(x_1 - V_{i1}) + ... + a_{iI}(x_n - V_{in}) \tag{5.2.1.2}$$

- Type 3. Second-order polynomial (quadratic type):

$$\begin{gathered} f_i = a_{i0} + a_{i1}(x_1 - V_{i1}) + ... + a_{in}(x_n - V_{in}) + \\ a_{i(n+1)}(x_1 - V_{i1})^2 + ... + a_{i(2n)}(x_n - V_{in})^2 + \\ a_{i(2n+1)}(x_1 - V_{i1})(x_2 - V_{i2}) + ... + \\ a_{i((n+2)(n+1)/2)}(x_{n-1} - V_{i(n-1)})(x_n - V_{in}) \end{gathered}$$

- Type 4: Modified second-order polynomial (modified quadratic type):

$$\begin{gathered} f_i = a_{i0} + a_{i1}(x_1 - V_{i1}) + ... + a_{in}(x_n - V_{in}) + \\ a_{i(n+1)}(x_1 - V_{i1})(x_2 - V_{i2}) + ... + \\ a_{i(n(n+1)/2)}(x_{n-1} - V_{i(n-1)})(x_n - V_{in}) \end{gathered}$$

where $x_j (1 \leq j \leq n)$ is an input variable and y_i is an output variable. Each fuzzy set A_{ij} is defined as

$$\mu_{A_{ij}} = exp\left(\frac{-(x_i - c_{ij})^2}{2 \times \delta_{ij}^2}\right) \tag{5.2.1.3}$$

where c_{ij} and δ_{ij} are the centre and width of the Gaussian membership function respectively.

[Step 4] Set the initial values of the c_{ij} with the centre vector V_i. The initial values of δ_{ij} is calculated as the standard deviation of the i^{th} cluster.

5.2.2 Optimisation

The optimisation techniques used for tuning the type-1 system is the combination of IMPGA, MA and LSM. In particular, the membership functions of the premises are explored in different search trajectories through the search space using the IMPGA while the parameters of the consequent parts are solved in a robust manner using an improved QR Householder LSM. To speed up the convergence rate, we implemented a space search MA as one of the search trajectories of the IMPGA. The MA that was used in our study is a combination of a local search and a GA, which guides the search to the promising region, and hence accelerate convergence to the global optimum.

5.2.2.1 Island Model Parallel Genetic Algorithm

Standard GAs have proved to be useful optimisation techniques supporting search of global nature. However, the generic method may get trapped in some local-optimal and therefore be unable to find high quality solutions. This deficiency becomes quite apparent in a larger search space. The IMPGA [94] is designed to have multiple subpopulations instead of a single large population, which helps to preserve genetic diversity, since each island can follow a different search trajectory through the search space. In particular, the Island model runs a serial GA on each of several loosely connected islands (processors or threads). Each GA is independent of the others and is usually started with a different random seed. Periodically, individuals are transmitted between the islands in a process called migration. The IMPGA favours the interactions within subpopulations rather than between them and eliminates the need of global synchronization that is required in the standard approach. There are two important parameters that control the migration process of the algorithm, i.e., *migration interval*, the number of generations (or evaluations) between a migration, and *migration size*, i.e., the number of individuals in the subpopulation to migrate. The IMPGA has often been reported to yield better performance than the standard GA, both in terms of the quality of the solution and effort, as measured in the total number of evaluations to get the solution [134]. One reason for the improved search quality is that each island maintains a de-

gree of independence and thus explores different regions of the search space while at the same time sharing information by means of migration.

5.2.2.2 Combination of Island Model Parallel Genetic Algorithm and Memetic Algorithms

In traditional IMPGA, each subpopulation is implemented as a serial GA, exploring different regions of a search space. However, the partially isolated nature of the island populations suggests that it is possible to use different population-based searches other than GA to implement the evolution of island populations. In particular, we are interested in implementing GA-based approaches with local improvement procedures, i.e., an MA, in some of the subpopulations. The main advantage of MA over GA is that once a good solution is found during the evolution process, the local area around the solution is explored, searching for more promising trajectories instead of continual jumping around the search space. In [114], there is a study that shows that the use of MA improves the convergence properties of the search dramatically, i.e., the use of a hill-climbing algorithm to explore the region surrounding the individual whose fitness value is higher than the best obtain so far can improve the results from five to six times. Our proposed search algorithm combines the advantages of both IMPGA and MA, i.e., the IMPGA guides the search to promising regions by exploring multiple search trajectories at the same time, while the MA helps accelerate convergence to the global optimum. However, it is not advantageous to implement more than one MA subpopulation because of the following two reasons. First, the MA is usually much more computationally expensive than GA, and therefore, the use of many MA subpopulations might significantly slow down the overall search progress. Second, with the frequent migrations of elite individuals between the islands, it is guaranteed that the best fitness individual of the whole population will eventually be migrated to the MA islands and subsequently, the local surrounding space of the islands will be searched for improvements by the MAs. In other words, one MA population is enough to explore the local area around the best solution found in the whole population. Hence, we decided to implement the MA in only one of the subpopulations, i.e., the first subpop-

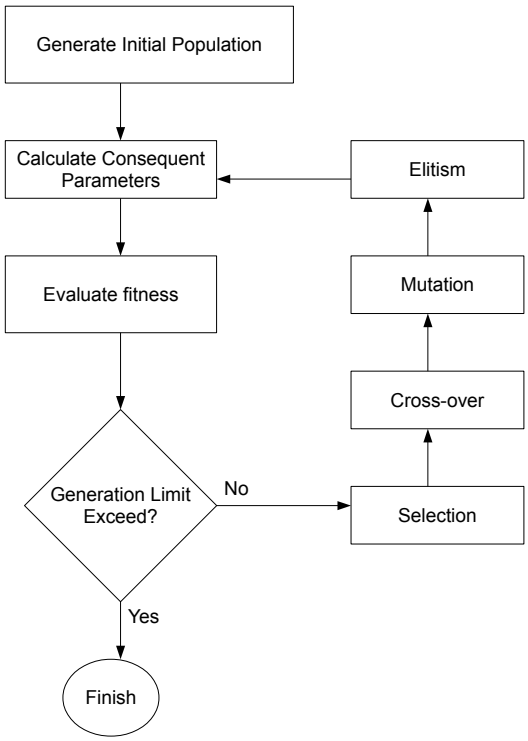


Figure 5.2: Genetic algorithm flowchart

ulation, and the rests of the subpopulations were implemented using standard GAs.

5.2.2.3 Premise Part Identification

5.2.2.3.1 Design of the island model parallel genetic algorithm The IMPGA consists of n islands. Each island contains S individuals. Each individual is defined by its chromosome. The first subpopulation implements the MA, which will be described later. The rest of the subpopulations are implemented as traditional serial GAs, each of which has its own recombination and mutation operators. The genetic encoding of the fuzzy rules in each island involves only the premises of the rules. However, to work out the fitness of a fuzzy system, both the parameters of the premises and consequent parts are required. Fortunately, the best consequent parameters can be found in a robust manner using the LSM, given the premises parameters. The combination of both IMPGA and LSM allows us to train the premises part of the fuzzy system only and thus reduce the search space dramatically. Figure 5.2 shows the steps implemented by the GA subpopulations.

Elitism is used for each GA to reserve a few slots in the next generation for the highest

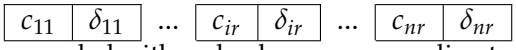


Figure 5.3: Premises are coded with real values corresponding to the center and width of the input fuzzy sets - n, r are the number of input variables and the number of fuzzy rules, respectively

fitness chromosomes of the current generation, without allowing the chromosomes to be crossed over or mutated in the next generation. The purpose of elitism is to ensure the survival of the fittest individuals during evolutionary process. The differences between the GA subpopulations lie in the choices of genetic operators used in each island, i.e., the choices of selection, cross-over and mutation operators.

Each island will evolve independently for a number of generation specified by the parameter *migration interval* before *migrations* occur, in which a number of individuals specified by the parameter *migration size* will be moved from an island to the next. The feasible migration paths between islands is defined by the *archipelago topology*, e.g., one-way ring topology, bidirectional ring topology, complete topology, isolated topology, etc. On arrival, the migrating individual replaces a random in the destination population if it has better fitness value. Migrations ensure that good fitness individuals will eventually transfer across the islands and have chances to follow different trajectories in the search space. The first subpopulation implementing the MA is dedicated to explore the local space surrounding the island, while the GA islands search many regions in the search space and, therefore, helps avoid the local optimum traps.

5.2.2.3.2 Genetic encoding As mentioned earlier, the premises part of the fuzzy system uses Gaussian MFs as specified in equation 5.2.1.3. It is, therefore, necessary to store the information concerning the center c and the width δ of each antecedent. There are r numbers of fuzzy rules, and each rule involves n input variables; therefore, each individual in the population is made up of a set of $n \times r$ fuzzy sets. Therefore, to completely represent all MFs in the premise part, we need a chromosome containing $n \times r \times 2$ genes. Figure 5.3 shows the genetic coding in details. All the subpopulations use the same representation of chromosome.

5.2.2.3.3 Genetic operators For the GA subpopulations, the operators are chosen from the set of standard GA operators that are widely used in the literature.

5.2.2.3.3.1 Crossover Any valid crossover operators in the literature can be used, e.g., one-point crossover, two-point crossover, uniform crossover, and line recombination. The line recombination is commonly used when the chromosomes are made up of a vector of real values. In line recombination, each offspring's real parameter is obtained from the two relative parents' parameters as a weighted mean. Therefore, if p and q are the two parents' parameters (a Gaussian width or center), the offspring's parameter p_o will be given by:

$$p_o = qa + p(1 - a) \tag{5.2.2.1}$$

where a is a real value in $[0, 1]$. The value a is randomly selected once for each offspring, i.e., all real parameters of the same offspring will use the same value of a. The offspring generated after crossover will replace their parents in the population.

5.2.2.3.3.2 Mutation We only use one type of mutation operator for all GAs. Chromosomes are randomly selected to mutate with a different probability for each island and if chosen, each parameter p_o is modified as

$$p_o = p_0 \left(1 - \frac{\delta}{2} + rnd(\delta)\right) \tag{5.2.2.2}$$

where $rnd(\delta)$ is a random real value in the range of $[0, \delta]$.

5.2.2.3.3.3 Selection The selection schemes can be chosen from the set of standard selection operators, for example, tournament selection, uniform selection, roulette-wheel selection, elitist selection, and truncation selection. For brevity's sake, the details of these selection strategies will not be included here. See [137] for more details.

Algorithm 5.2 Memetic Algorithm

Step 1: Initialize the population
Step 2: Evaluate all individuals and sort the population in descending order according to the fitness values.
Step 3: Randomly select M individuals for recombination or crossover.
Step 4: Recombine the selected individuals to generate new ones. Only the new offspring with better fitness than the current worst individual of the population will be added to the population. When an offspring is added, the worst individual is removed. This ensures that the size of the population is unchanged. New offspring are added in such a way that preserves the order of the individual's fitness of the population.
Step 5: Randomly select another N individuals for mutation. The selection must ensure that the best fitness individual is selected at least once.
Step 6: Mutate the selected individuals and add the new offspring to the population as described in step 4.
Step 7: Repeat step 3 to step 6 until the stopping criterion is satisfied.

5.2.2.3.4 Memetic algorithm The proposed algorithm is the GA-based MA. The idea is to incorporate local searches into the GA by exploring the neighborhood space of the current population and include all the individuals that increase the overall fit of the population. The MA can converge faster because in most local areas, a solution with better fit is closer to the optimal solution. The algorithm is outlined in Algorithm 5.2**.** The algorithm proposed is similar to that of GA. The main differences are the criterion to add offspring into the population as mentioned in step 4, the multi-parent crossover operator, and the mutation operator. In the mutation process, a random number of chromosomes are selected. A random number of genes of the selected chromosomes are chosen to undergo the mutation process. Each selected gene is mutated by

$$o_i' = o_i + \alpha \times (o_i - min) + (1 - \alpha) \times (max - o_i) \tag{5.2.2.3}$$

where o_i is the mutating gene. o_i' is the new value of the gene, α is a random number in the range of $[0, 1]$. min and max are the minimum and maximum possible values of the gene o_i, respectively. In the case of mutation producing invalid values, the mutation process is repeated several times trying to find a valid one. If all fails, a random value between min and max is selected as the mutated gene. The mutated chromosome will replace the worst fitted individuals in the population if its fitness is better. The steps performed by the crossover operation are shown in algorithm 5.3**.** The whole process

Algorithm 5.3 Crossover operation

Step 1: Generate a sequence of M random numbers $a_1, ..., a_M$ in the range of $[0, 1]$
Step 2: Each gene of the offspring's chromosome is generated by the weighted mean of that same gene in each parent.

$$o_i = \frac{\sum_{j=1}^{M} a_j \times parent_{ij}}{\sum_{j=1}^{M} a_j} \tag{5.2.2.4}$$

where o_i is gene i^{th} of the offspring. $parent_{ij}$ is gene i^{th} of the j^{th} parent's chromosome.

of selection, mutation and recombination is repeated until the stopping criterion is met. The simplest form of stopping criterion which was used in this study is the loop counter limit. The algorithm stops when the number of loops reaches the specified limit value.

5.2.2.3.5 Consequent parameters and fitness evaluation The calculation of the outputs of the fuzzy model from (2.1.2.4) can be rearranged into a matrix form as follows:

$$y_k^* = \sum_{i=1}^{r} w_{ik} y_i = \sum_{i=1}^{r} w_{ik}(f_i(x_{1k}, ..., x_{nk}, V_i) + M_i) \tag{5.2.2.5}$$

$$y_k^* - \sum_{i=1}^{r} w_{ik} M_i = \sum_{i=1}^{r} w_{ik} F_{ik} A_i \tag{5.2.2.6}$$

where k is the index of sample data, A_i is the consequent parameters of the i^{th} fuzzy rule, which will be the same, regardless of inputs. $A_i = [a_{i0}, ..., a_{ip}]$, where p is the number of terms of the consequent part depending on the type of the consequent polynomial. F_{ik} is a matrix which is rearranged with input data and information granules (centres of clusters) of the k^{th} data. For example, in the case where the consequent polynomial is linear, F_{ik} can be defined as:

$$F_{ik} = \begin{bmatrix} 1 & (x_{1k} - V_{i1}) & \cdots & (x_{nk} - V_{in}) \end{bmatrix}^T. \tag{5.2.2.7}$$

In (5.2.2.6), we can rearrange the right hand side of the equation as

$$y_k^* - \sum_{i=1}^{r} w_{ik} M_i = \sum_{i=1}^{r} (w_{ik} F_{ik}) A_i = \sum_{i=1}^{r} G_{ik}^T A_i \tag{5.2.2.8}$$

where $G_{ik} = (w_{ik} F_{ik})^T$. In case the consequent polynomial is linear, G_{ki} is defined as:

$$G_{ik} = \begin{bmatrix} w_{ik} & w_{ik}(x_{1k} - V_{i1}) & \cdots & w_{ik}(x_{nk} - V_{in}) \end{bmatrix}. \tag{5.2.2.9}$$

Let $G_k = \begin{bmatrix} G_{1k} \vdots & G_{2k} \vdots & \cdots & G_{rk} \end{bmatrix}$ be the matrix of $1 \times (r \times p)$ dimension combining all G_{ik}. $A = \begin{bmatrix} A_1 \vdots & A_2 \vdots & \cdots & A_r \end{bmatrix}^T$ is the matrix of $(r \times p) \times 1$ dimension combining all A_i. The matrix A contains all the consequent parameters of all r fuzzy rules combined in one column. The right hand side of equation (5.2.2.8) can be represented as a matrix multiplication

$$y_k^* - \sum_{i=1}^{r} w_{ik} M_i = G_k A \tag{5.2.2.10}$$

Let

$$Y = \begin{bmatrix} y_1 - (\sum_{i=1}^{r} w_{i1} M_i) & \cdots & y_m - (\sum_{i=1}^{r} w_{im} M_i) \end{bmatrix}^T \tag{5.2.2.11}$$

$G = \begin{bmatrix} G_1 & G_2 & \cdots & G_m \end{bmatrix}^T$, where m is the total number of data. The best consequent parameters are the solutions of the linear overdetermined system $Y = GA$ and can be solved using the LSM. Once the input variables of the premise and parameters have been already specified, the optimal consequence parameters that minimize the assumed performance index can be determined. The performance metric used in this study is the standard root mean squared error (RMSE).

$$PI = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (y_i - y_i^*)^2} \tag{5.2.2.12}$$

where m is the total number of data and y_i^* is the output of the fuzzy model.

5.2.3 Consequent identification

5.2.3.1 Least-Squares Methods

There are three different algorithms for computing the least-squares minimum that are most commonly used:

1. normal equations, e.g., the Moore–Penrose pseudo-inverse [91, 106]: not computationally efficient, less accurate;

2. singular value decomposition (SVD) [50]: expensive, more reliable;

3. QR Decomposition [45]: faster than SVD but less stable.

Most existing works in the Mackey-Glass problem either used the Moore–Penrose pseudoinverse or the SVD LSM to find the best consequent parameters. In our study, we investigated which LSM was most suitable for the Mackey-Glass problem by using all three methods in the fitness evaluators of the parametric optimization and comparing the results. Our experiment has shown that the QR methods produced better results than the SVD in close to 50% of the test cases which both SVD and QR were able to solve. However, the number of tests where the QR was unable to solve is higher than that of the SVD. Both the SVD and the QR have shown better accuracy than the Moore–Penrose pseudoinverse. The QR methods have the fastest computation time among the three. In the test cases that the QR was unable to solve, the solutions contained values which were greater than the highest representable floating number which led to overflow exceptions and subsequent failure of the algorithm. In our approach, we propose an improved version of the QR LSM that uses the Householder transformation to compute the QR decomposition. The proposed method has the same accuracy as the standard QR Householder algorithm in general cases, while minimizing the occurrence of overflows where possible.

5.2.3.2 QR Householder least-squares solver

5.2.3.2.1 QR least-squares solver First, let us very briefly review the standard QR method for solving the least-squares problem. The solution of least-squares problems

$$min \|b - Ax\| \tag{5.2.3.1}$$

where A is a $m \times n$ matrix. The matrix A is subjected to a QR decomposition which decomposes A into a product $A = QR$ where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ triangular matrix which is partitioned into a $n \times n$ upper triangular block, R_n, and a $(m - n) \times n$ matrix consists entirely of zeroes.

$$R = \begin{bmatrix} R_n \\ 0 \end{bmatrix} \tag{5.2.3.2}$$

To find a solution to the overdetermined problem $Ax = b$ which minimizes the norm $\|Ax - b\|$, first find the QR factorization of A: $A = QR$. The solution can then be expressed as

$$x = R_n^{-1} PQb \tag{5.2.3.3}$$

where R_n and Q are the same as before, but now, $P \in m \times (n \times m)$ is a projection matrix that maps a vector in R^m into R^n. There are several methods for actually computing the QR decomposition, such as by means of the Gram–Schmidt process, Householder transformations, or Givens rotations [50]. Each has a number of advantages and disadvantages. However, the Householder transformation method has greater numerical stability than the others and therefore is chosen for the Mackey-Glass time series problem. Details of the standard QR Householder transformation can be found on Appendix A.

5.2.3.2.2 Improved QR Householder least-squares solver Based on the standard QR Householder least-squares solver (see appendix A), we propose a few additional steps to the algorithm so that the method can search for solutions beyond the highest representable floating numbers. The idea of the algorithm is based on the fact that $\|Ax - b\| = \|(As)(x/s) - b\|$. If we can find the solution of the right hand side, we can deduce the solution of the left. We choose the value of s is powers of base 10 so as to minimize the loss of accuracy in arithmetic operations. The steps of the algorithm are given in algorithm 5.4.

5.3 Context-Dependent Fuzzy Approach

This study proposes a context dependent fuzzy system in which the influence of context on the membership functions is represented by some sort of filtering applied on

Algorithm 5.4 The improved QR Householder least-squares solver

[Step 1] : Perform QR Householder decomposition of matrix A.
[Step 2] : Find the solution of x using equation (5.2.3.3).
[Step 3] : If there is no overflow value, return the solution. Otherwise go to step 4.
[Step 4] : Find the largest element r_{max} of R_n.
[Step 5] : Scale the matrix R by the value s calculated by

$$s = 10^{Nmax - \log_{10}(r_{max})} \tag{5.2.3.4}$$

where $Nmax$ is the maximum positive integral exponent that the floating-point type can represent as a finite value when a base of ten is raised to that power.
[Step 6] : Find the solution of x using equation (5.2.3.3) with the scaled matrix R.
[Step 7] : If there still exists overflow value, return fail.
[Step 8] : The solution x is returned along with the scaling factor s. The actual solution is sx. However, we cannot represent sx using floating numbers, we have to return s and x separately.

the base type-1 MFs to suit the context particularities. In particular, let the i^{th} rule of based fuzzy system be:

$$R^i : \begin{array}{cc} IF & IF(x_1 \ is \ A_{i1}) \ AND \ ... \ AND \ (x_I \ is \ A_{in}) \\ THEN & y_i \ = f_i(x_1, ..., x_n) \end{array} \tag{5.3.0.1}$$

The output of the base system is calculated based on the activation levels of the rules using the following expression:

$$y = \sum_{i=1}^{r} w_i f_i(x_1, ..., x_n) \tag{5.3.0.2}$$

Let the context attribute set $C = \{X_1, ..., X_n\}$ where X_i is the i^{th} input variable. Each context dependent MF is now defined as a function dependent not only on its own universe of discourse U but also on the values of the input variables of the system. The i^{th} rule of context dependent fuzzy system is of the form:

$$R'^i : \begin{array}{cc} IF & (x_1 \ is \ A'_{i1}) \ AND \ ... \ AND \ (x_n \ is \ A'_{in}) \\ THEN & y_i \ = f_i(x_1, ..., x_n) \end{array} \tag{5.3.0.3}$$

In this study, we consider one of the simplest forms of context adaption where the MFs in each rule are scaled by the same value. In other words, $A'_{ij} = A_{ij} \times a_i(x_1, x_2, ..., x_n) =$

$A_{ij} \times a_i$. The new activation level w'_i of rule i is now calculated as $w'_i = w_i \times c_i$, where

$$c_i = \begin{cases} a_i & if\ AND\ is\ min() \\ a_i^n & if\ AND\ prod() \end{cases} \tag{5.3.0.4}$$

The output of the context dependent fuzzy system is calculated similar to that of the base system:

$$\begin{aligned} y' &= \sum_{i=1}^{r} w'_i f_i(x_1, ..., x_n) \\ &= \sum_{i=1}^{r} w_i \times c_i \times f_i(x_1, ..., x_n) \end{aligned} \tag{5.3.0.5}$$

where c_i is the value by which each rule i of the base fuzzy system is scaled to the current context. If $c_i = 1$, the CDFS reduces to a regular type-1 fuzzy system. Let K be the matrix of dimension $1 \times r$ where r is the number of fuzzy rules. K_{0i} =the output of rule $i^{th} = w_{ik} \times f_i(x_1, ..., x_n)$. The set of c_i is represented as a matrix M of dimension $r \times 1$. The output can now be expressed as a matrix multiplication:

$$y' = KM \tag{5.3.0.6}$$

The matrix K is fixed, while the matrix M is changed according to the context. It can be seen that the matrix K is the same in both the type-1 and the CDFS. The learning of the CDFS can be split into two stages: the learning of the standard type-1 fuzzy system in which the matrix K is calculated, and the learning of the matrix M. The first stage was mentioned in the previous section. In the following, the context representation and the methodologies used to tune the matrix M is provided.

5.3.1 Context Definition

Suppose that the training data set contains m pairs of **{input variables, output variable}**. We can cluster such data pairs into p clusters using K-means clustering algorithm. The obtained centre points are regarded as typical for each cluster. A set of inputs $(x_1, ..., x_n)$ is said to belong to cluster v_i if the distance to the centre point of v_i is the smallest among all the clusters. The grouping of data set into clusters defines

the contexts that influence the inferencing process of the fuzzy system. The matrix M can be tuned for each cluster separately using only the points in the cluster without considering other clusters.

5.3.2 Tuning of the Transformation Matrix

The tuning of the transformation matrix M is rather simple and can be done using any optimization algorithm. In our work, an extension of the MA described in Section III is used to tune the matrix M for each cluster. The genetic encoding is a real vector of r elements representing the matrix M. The fitness function used is the standard RMSE of all data points in a cluster as in equation (5.2.2.12). To ensure that the solution found by the MA must improve the result of the type-1 system, when initialize the population, it is necessary to have at least one individual whose genes are all ones. When M contains all ones, the output of the CDFS is the same as the type-1 fuzzy system. The MA will run for a large number of generations before returning the best fitness individual as the transformation matrix M for a specific cluster. Once the transformation matrix for each cluster is obtained, the actual outputs for the whole data set is calculated by:

$$y' = KM = \sum_{i=1}^{r} w_{ik} \times c_i \times f_i(x_1, ..., x_n) \tag{5.3.2.1}$$

Note that (5.3.2.1) can be rearranged into matrix form as follows:

$$\begin{aligned} y' &= \textstyle\sum_{i=1}^{r} (w_{ik} \times c_i) \times f_i(x_1, ..., x_n) \\ &= W_n(XF) = (W_n X)F \end{aligned} \tag{5.3.2.2}$$

where X is the input matrix, F is the consequent parameter matrix, and W_n is the matrix which stores the scaled activation levels of each rule for each set of inputs. From (5.3.2.2), LSM can be utilized to find the best consequence parameters F that would minimize the RMSE of data set. If the new RMSE is an improvement, the algorithm is repeated using the new consequence parameters. The pseudocode of the tuning algorithm is presented in Algorithm 5.5.

Algorithm 5.5 Pseudo code of the tuning algorithm

1. repeat
2. for cluster = 1 to p do
3. begin
4. $M_{cluster}$= memetic_algorithm_on_cluster(cluster)
   (see alg 5.2, section III).
5. end
6. Calculate the real output of CDFS for the training data set .
7. Calculate the current RMSE using equation (5.2.2.12).
8. If current RMSE is better than previous RMSE then
9. Use LSM to solve the consequence parameters in equation (5.3.2.2)
10. until no_improvement or loop count exceeds a threshold;

5.4 Experiment and Results

As previously mentioned, the well-known Mackey-Glass chaotic time series is used to validate the performance of the proposed methods. The equation used to generated the series is of the form:

$$\frac{dx(t)}{dt} = \frac{0.2 \times x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1 \times x(t) \tag{5.4.0.1}$$

The most common setup found in the literature is used to compare our method with others in the most straight forward manner. From the Mackey-Glass time series $x(t)$, 1000 input-output data pairs of the format: $[x(t-30), x(t-24), x(t-18), x(t-12), x(t-6), x(t), x(t+6)]$ between $t = 118$ and 1117 is extracted. Among them, the first 500 pairs were used as the training data set, while the remaining 500 pairs were the checking data set. In the literature, either four or six input variables were commonly used. In our experiment, we examined our approach using both four and six input variables. In the case of four inputs, the format of the input-output data pairs used was $[x(t-18), x(t-12), x(t-6), x(t), x(t+6)]$. The number of fuzzy rules used for both cases was 16. The performance metric used was the standard RMSE as in equation

(5.2.2.12). The consequent polynomial of the fuzzy rule is the second-order polynomial (type 3). The training data (the first 500 pairs) was used to set up the type-1 system, i.e., the consequent coefficients (information granules), the initial premise parameters (centers and widths of Gaussian MFs), and to form the contexts for the CDFS. To set up and initialize the type-1 system, the training data was clustered by the K-means algorithm into 16 groups, corresponding to 16 fuzzy rules. The center and width of each cluster was then used to create the initial MFs of the premise part and the coefficients of the consequent part of the corresponding fuzzy rule. Note that, with the initial MFs created by the K-means clustering, the type-1 fuzzy systems produced the RMSEs of 0.0028 and 0.00048 on the testing data in the case of using four and six inputs, respectively, which are fairly good compared to existing results in the literature. In the development of the CDFS, the K-means algorithm was used to form the contexts. The number of contexts (clusters) used for this experiment was 16. Table 5.1 lists the common parameters used by all islands in the IMPGA. Table 5.2 lists the selection and crossover operators used in each island. Table 5.3 summarizes the results of comparative analysis of the proposed type-1 and CDFS with respect to other existing models. The models in comparison include adaptive network-based fuzzy inference system [68], RBFNN [5], fuzzy neural network [88], genetic fuzzy learning [115], HFS trained with Probabilistic Incremental Program Evolution (PIPE) and Particle Swarm Optimization (PSO) [27, 28], mutable rule base [37], passive membership function validity string (MFVS) [73], fuzzy c-mean clustering with IG-based fuzzy radial basis function neural networks [30, 31], PSO and space search algorithm (SSA) for ANFIS-based fuzzy models [59], and Simulated Annealing [4]. All the cited work have used the same sequence of Mackey Glass series, i.e., between $t = 118$ and 1117, the same number of training and testing data as well as the interval of the inputs used for prediction. The number of inputs used are either four or six as aforementioned.

As can be seen, our type-1 fuzzy approach has produced the lowest error with the least numbers of rules in both four- and six-input cases. When using more input variables, there are a larger number of parameters in both the premise and consequent parts of the fuzzy rules. The increase in number of consequent parameters allows the system to

IMPGA parameters	Values
Number of generations	4500
Island topology	Ring
Number of subpopulation	5
Size of subpopulation	[30,30,30,30,30]
Crossover rate	[0.75,1,1,1,1]
Mutation rate	[0.01,0.01,0.01,0.01,0.01]
Migration interval	10
Migration size	3
Migrants type	Random individuals

Table 5.1: Common parameters of the island models

Island number	Selection	Crossover
1	NA	NA
2	Tournament	Two-point
3	Uniform	Line
4	Weighted	One-point
5	Elitist	Uniform

Table 5.2: GA operators used in each island

Model	No. of rules	No. of inputs	RMSE training	RMSE testing
Anfis [68]	16	4	0.001600	0.001500
RBFNN [5]	16	4		0.003000
FNN model [88]			0.014000	0.009000
Gefrex [115]	20	4	0.005400	0.006100
Passive MFVS [73]	24	4		0.001100
Hierarchical FS+PIPE[27]	28		0.012000	0.012900
Simulated Annealing [4]	16	4	0.003640	0.003660
MRB [37]	26	4	0.000990	0.000884
Proposed type-1	**16**	**4**	**0.000150**	**0.000250**
Proposed CDFS	**16**	**4**	**0.000120**	**0.000160**
IG-FRBFNN [30]	15	6	0.000230	0.000320
IG-FRBFNN [31]	16	6	0.000150	0.000390
PSO+IG [59]	16	6	0.000330	0.000350
SSA+IG [59]	16	6	0.000120	0.000150
Hierarchical FS +PSO [28]	10	6	0.010500	0.015100
Proposed type-1	**16**	**6**	**0.000014**	**0.000032**
Proposed CDFS	**16**	**6**	**0.000008**	**0.000021**

Table 5.3: Comparative analysis of selected models

achieve a much lower error. All experiments were performed on a Windows 7 laptop with Core i5 at 2.5-GHz processor. Only one single run is made and the best results are reported. The computation time to achieve the results for the type-1 fuzzy systems were 4-5 hours. The RMSE reported was the fitness of the best individual of the whole population found when training using the training data. The corresponding RMSE for that same individual on the testing data is reported in the last column of table 5.3 (RMSE testing). The results have shown a dramatic improvement of accuracy (up to 10 times) if six inputs were used instead of four. In the case of four inputs, our type-1 fuzzy approach has shown a very good performance which is bettered only by a very small margin by the SSA proposed by Huang et al. [59] using six inputs. Our fuzzy type-1 system is the only approach that can produce competitive performance with the six-input approaches albeit using fewer inputs and rules. In the case of six inputs, the proposed approach also achieves the lowest RMSE of all the methods in the comparison. The RMSE produced by our approach is five times smaller than that of SSA which, to the best of our knowledge, is the current state of the art system with the best prediction accuracy for Mackey-Glass time-series in the literature. In any case, our type-1 fuzzy approach has shown a much better accuracy than any other existing approaches in the literature. Note that the approach used in [59] is similar to ours. The structure of the fuzzy systems used are almost identical; the main difference lies in the choice of optimization techniques and the LSM used to calculate the consequence parameters, i.e., in [59], PSO and SSA were used to optimize the fuzzy system while a combination of IMPGA, MA and an improved LSM were adopted in our work. It is evident that both our improved LSM, which designs to minimize the occurrences of the underflow and overflow problems when dealing with floating point numbers, and our optimization search algorithm (IMPGA+MA), which explores the search space in multiple trajectories simultaneously to avoid getting trapped in local-optimal, contribute greatly to the accuracy improvements.

The proposed context-dependent fuzzy approach has further improved the performance of the type-1 system by 50-60% and achieved an impressive accuracy of **0.000021** with six input variables. To the best of our knowledge, this is the best accuracy for the

Mackey-Glass time series prediction to date using a similar setup. The context adaptation process achieved the aforementioned results in a matter of minutes on the same testing machine. The performance improvement of 60% is impressive when taking into account that the fuzzy type-1 system is very well optimized. In fact, we have actually tried to improve the result of the type-1 fuzzy approach by letting the optimization algorithm to run for more than 24 hours after the best solution was found. However, the improvement is very minor. It is clearly shown that the proposed context adaptation algorithm is not only efficient in terms of computational time but also very effective as it can help further improve the performance of a well-optimized fuzzy system.

5.5 Summary

This chapter presented an improved optimisation methodology for the parameter tuning of the type-1 fuzzy systems and a technique to model the effect of context dependence into the fuzzy systems. The parameter tuning for type-1 fuzzy systems is a combination of IMPGA, MA and an improved QR Householder least squares method which results in a significant performance boost when tested on the well-known Mackey-Glass time series benchmark compared to other approaches in the literature. The context-dependent fuzzy approach which employs a K-means algorithm to create the contexts and an iterative algorithm based on the MA to adapt the MFs further improves performance. The final context-dependent fuzzy system provides the best performance yet reported for Mackey-Glass time series prediction. Our future works will concentrate on the following.

1. increase the number of data sets;
2. investigate other ways of adjusting the MFs to the context, e.g., linear and non-linear transformation of MFs;
3. apply the proposed approach to more complex problems (real-world applications).

CHAPTER 6

The TORCS-based Simulated Car Racing Championship

In this chapter, we present our context-dependent fuzzy controller designed for the TORCS-based Simulated Car Racing Championship [78] (SCRC). The aim is to build an effective autonomous car racing controller which can perform well in a variety of tracks while given a minimum amount of sensory information about the local track and car properties. The sensory data can be so limited that it is at times virtually impossible to guess the direction of the incoming corners or to, perform complicated path planning. Due to this scarcity of information, none of the SCRC's controllers to date has been able to achieve the performance of any regular TORCS controllers who have access to global and accurate track data. In this work, the rule of SCRC is slightly lessened by removing all added noise to the sensory track data so that the data is sufficiently reliable to construct local track models. The construction of track models is done by combining immediate sensory information with the previous learned models obtained while the car was racing in the track. The track modelling method is novel by itself and plays a very important role in the development of the controller. The local track models acts as a context which influences the meaning of fuzzy terms used by our fuzzy controller. The combination of our proposed track modelling and fuzzy approach results in a very robust performance of the SCRC's controller, which has been shown to outperform the best controllers of the regular TORCS in numerous tracks. Details of

the controllers as well as the performance comparison with other approaches are given in this chapter.

6.1 Introduction

The simulated car racing environment TORCS [41] is an open-source multi-platform car racing environment with a highly realistic track and car simulation engine. In the past ten years, TORCS has been extensively used as testbed for artificial intelligent algorithms where bots are created to illustrate the applicability of various techniques in the car control task. There are several competitions in the TORCS community. The most well-known competition is hosted by the TORCS Racing Board (TRB) which provides a platform to organise and perform racing championships for artificial intelligence robots. In this competition, every bot has access to global track information prior to the race which allows it to perform extensive calculations of the best racing lines and use them in the actual race. Moreover, detailed private setup of the cars as well as their states during the race are all provided to the bots. There is virtually no uncertainty in the race as every required piece of information is provided to the bots exactly as the originals without the effect of noise. As a result, the controllers can achieve excellent performance especially in solo races where they do not have to handle the unexpected events of car over-takings and car collisions. After careful examination of existing solutions to the car racing problems, existing techniques can be categorised into three groups, each with its pros and cons:

- The model-based approach. In this type of approach complex models of the track are required or must be built from the sensory data. These models are then used to perform high-level reasoning, i.e., planning. This is the most common approach and it generally produces very good performance, especially when there are not a lot of unexpected events or uncertainties about the track, e.g., in solo race. The drawback of this approach is that the path-finding algorithm is often a very time-consuming process, which makes this approach generally not suitable for dynamic and uncertain environments. Examples of this kind of approach are

[8, 17, 36, 75]

- The sensor-based approach. The philosophy of this approach is the opposite: it favours reactivity. Path-finding is not necessary and actions follow sensor perceptions closely, almost like a reflex. This type of approach is most appropriate to dynamic and uncertain environments in which unexpected events, e.g., avoiding obstacles, overtaking, recovery, are dealt with as soon as they are detected by the sensors. However, high-level reasoning is difficult to achieve (if not impossible) in this method and the overall performance of this approach is not as good as the model-based approach's, particularly in a solo race. Fuzzy controllers are examples of this type of approach. See [81] and [107]

- The hybrid approach. This combines the advantages of both model- and sensor-based approaches. Typically, a path-finding algorithm is first used to compute the optimal path, which will subsequently be used as a guidance for the car to follows. A sensor-based module will control the car following the pre-computed path as closely as possible while reacting to any unexpected events. Examples of this kind of approach are [44] and [38]

The three aforementioned approaches are applicable not only to TORCS but also to any vehicle racing games. In the original TORCS, the best overall controllers are normally those adopting the model-based or hybrid approach, and they also score the highest in a solo race. One of the best controllers available for TORCS is the SIMPLIX [8] which was developed by Wolf-Dieter Beelitz for the TORCS Racing Board season 2008 using the model-based approach. SIMPLIX proposed a very effective algorithm to find the ideal racing lines that was later adopted by many other authors as the standard technique for path planning tasks. Since the introduction of SIMPLIX, there nevertheless seems to be a general lack of interest in the TRB community due to the lack of innovation in the set up of the competition, i.e., the TRB environment is unrealistic in that everything is known and accurate while lacking the space for learning and improving the driving style. Most authors follow very a similar approach. First, calculating the best racing lines using either the K1999 method [36] by Remi Coulom or the SIM-

PLIX method [8] by Wolf Dieter. Then, adding some manual configurations for each track telling them how fast to drive, and what line to take/avoid, on various sections of the track to improve performance. Therefore, there is little scope for learning and improving during the race - either one configuration is better or it is not. Fortunately, since 2008, another TORCS-based competition has emerged; the Simulated Car Racing Championship [78], that was designed to encourage learning techniques. The competition was first introduced at the IEEE World Congress on Computational Intelligence 2008 (WCCI) and the IEEE Congress on Evolutionary Computation in 2009 (CEC). The main difference between the two competitions is the degree of information available to the robots. While the TRB's robot has complete information about the track before and during the race as well as comprehensive details about the properties and state of the car, the SCRC's robot only partially knows the immediate surroundings during the race and some very limited information about the state of the car. Hence, the SCRC's robot cannot plan ahead and needs to learn. The challenges faced by SCRC's bots are essentially very similar to the difficult position of a human driver in the real world, namely to make the best possible use of scarcity of information in an uncertain environment with full of unexpected events. This makes developing an SCRC bot a much more complicated task. However, successful strategies may be well-applicable in autonomous systems that have to navigate in unknown environments in the real world.

The SCRC consists of two stages: the warming up and the main race. In the warming up race, track information is given accurately to the robot. However, during the main race, noise is added to the observation of the track data. The separation into the two stages is meant to encourage two types of learning. The first is learning the track model, which is only possible in the warming up stage when noise is not present. The track model can be learned and saved for performance improvement in future races. The second is learning to race in an uncertain environment and improve the performance after each lap during the race. Both types of learning are required to be very fast given a response time restriction of 10 ms within which a bot must give a command to the car. While learning and maintaining a model of the track can help improve the performance significantly, it is a very challenging task, especially under strict time constraints. As a

consequence, most SCRC solutions avoid track modelling altogether and focus only on reacting to the sensory information. In other words, most approaches are sensor-based, i.e., actions follow perception almost like a reflex without any complicated planning. Note that both types of learning are not necessary in TRB as a TRB bot has complete knowledge of the track model prior to the race, which allows it to find the optimal racing lines that could not be improved further. All the learning in TRB is carried out and finished before the race starts while in SCRC, the learning process starts and carries on during the race. Apparently, due to these significant differences, the model-based approach which works well for TRB are not extendable to SCRC. Usually, an SCRC bot employs one of two approaches:

- Either a black box approach is used by providing all available data to a learning algorithm which adapts the controller without any insight concerning how the controller works [16, 18, 19, 93, 97, 107]; or
- They employ some methods to learn the track model and then use this model to navigate around the track [2, 110].

To the best of our knowledge, the best SCRC solution has outperformed average human players, but is yet to achieve the level of an average TRB bot. Interestingly, the latter approach which learns the track model first, have not shown significant performance boosts compared to those which only learn from experience. After a careful examination of existing track modelling approaches, it is detected that the track models used in these approaches usually provide only basic information about direction and some very approximate ideas about the speed allowance of the incoming corners while lacking very important pieces of information for any path-planning algorithms, i.e., the curvature of the corners. As a result, the models can only be used to prepare the speed of the car before the corners. The performance advantage is therefore insignificant. In this chapter, a much more sophisticated approach to track modelling is proposed by attempting to reconstruct the track segments as close to the originals as possible from the current sensory data and previous observations of the track. Although it is not possible to fully reconstruct the exact track segments, our method provides a good es-

timation of direction and radius of incoming corners which allows some planning to be implemented. We believe that processing the sensory data into a human understandable level, i.e., geometry shapes and sharpness of the bends rather than points on the edges, is one of the most fundamental steps towards developing a high-performance car driving controller. This is due to the fact that real-world human drivers, although not having global vision of the track, can most of the time derive a close to optimal solution from only approximated models of the track. Based on the learned track models, we develop a hybrid controller which can plan ahead and at the same time, react to whatever events that might affect the stability of the car when following the planned routes. Unlike other hybrid controllers which usually focus more on path planning, reactivity is in fact the focal point in our approach. The core of our controller is a context-dependent fuzzy system which contains all the general rules that define the driving behaviours of the bot. These rules are fixed regardless of the situation. However, actual representations of the linguistic terms are defined according to the shape of the visible parts of the track so that the same rules can be used to take any kind of corners. Our approach is very similar to that which a human driver has to process in the real-world, i.e., observe the track, plan a strategy and adapt the driving behaviours to the situations. Both the method to learn the track models and the context-dependent fuzzy system are novel and efficient enough to be applied in real-time during the races. Our bot, Drifter, is the only SCRC bot that can achieve competitive performances with the best TRB bots in their most comfortable situations, i.e., in solo races. In fact, Drifter can even outperform SIMPLIX in 16 out of 20 selected tracks from the TORCS distributions. In this chapter, for the sake of brevity, we only focus on two of the most important agents which are responsible for implementing the track modelling and on-road strategies. The rest of the chapter is organised as follows:

Section 2 presents the SCRC setup and its sensory model. Section 3 details the method for learning the track models. Section 4 describes the context-dependent fuzzy controller that was built upon the learned track models. Section 5 presents the experimental setup and results. And finally section 6 discusses the results and sketches some future research directions.

6.2 The Simulated Car Racing Championship setup

The Simulated Car Racing Competition provides a client-server architecture, in which the server sends information about the race while the client decides on the actual control commands. Particularly, at every time step of 0.02s, the local track data and the state of the cars are converted into simulated local sensory information by the server, which subsequently is sent to the client. The client, in turn, has to process the sensor information before sending the corresponding motor commands back to the server. These commands are then applied to the car within the TORCS simulator by the server. The major challenge of the championship is that the bot only partially knows the immediate surroundings during the race and some very limited information about the state of the car. The information provided by SCRC slightly varies from competition to competitions. However, the main idea remains the same. This chapter will follow the setup of SCRC 2008 which provides the least information about the track environment. Tables 6.1 and 6.2 list and describe all available sensors and the motor commands available to the client in the SCRC setup [79].

6.2.1 The sensory model

The client of SCRC has assess to 72 sensor values, 19 of which are track sensors (see Figure 6.1a). These track sensors provide the "free" length of the track up to 100 m from −90 to 90 degrees in steps of 10 degrees relative to the current heading of the car.

6.3 Track modelling

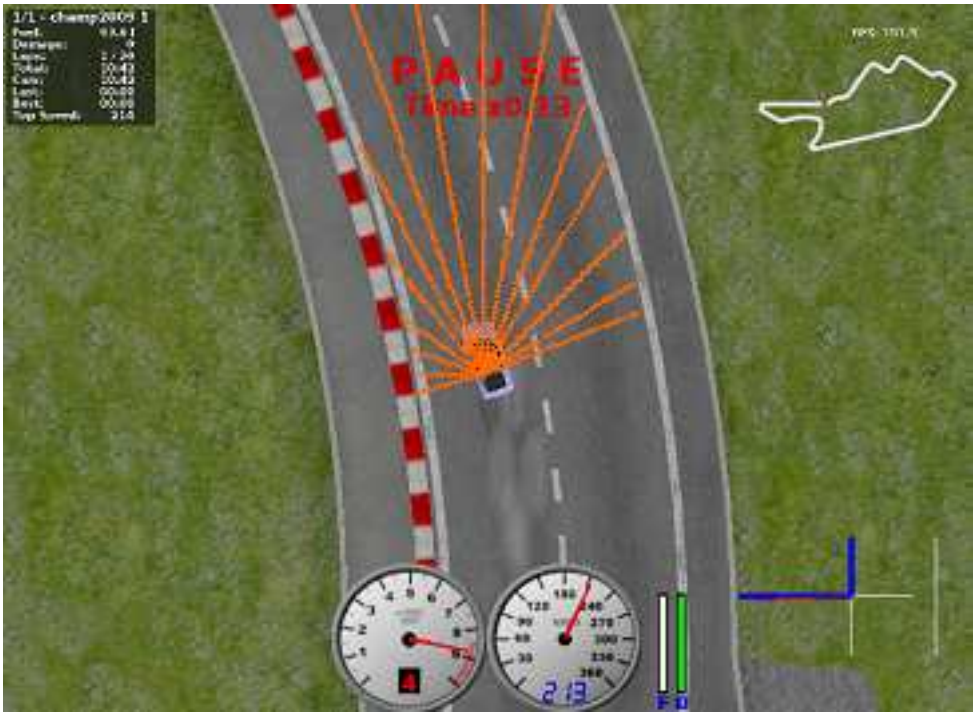
Track modelling is the task of integrating the information gathered with the robot's sensors into a given representation. Usually, a TORCS track is represented as a series of connected line or circle segments joined together and the same representation is used to model the track in our case. The main challenge of the modelling task is the scarcity of available sensory information to reconstruct the segments. Most of the time, it is not difficult to estimate the current track segment that the car is travelling on.

Name	Description
angle	Angle between the car and the direction of the track axis
curLapTime	Time elapsed during current lap
damage	Current damage of the car
distFromStartLine	Distance of the car from the start along the track line
distRaced	Distance covered by the car from the beginning of the race
fuel	Current fuel level
gear	Current gear: -1 reverse, 0 neutral, and gear from 1 to 6
lastLapTime	Time to complete the last lap
opponents	Vectors of 36 sensors that detects the opponent distance in range of 100m within the specific 10 degrees sector; each sector covers 10 degrees, from $-\pi$ to π around the car
racePos	Position in the race with respect to other cars
rpm	Number of rotation per minute of the car engine
speedX	Speed of the car along the longitudinal axis of the car
speedY	Speed of the car along the transverse axis of the car
track	Vector of 19 distance to track edge sensors in meters, which cover the half circle facing towards the car's orientation in equally spaced sectors. The maximum range of each sensor is 100 meters. When outside of the main track, then these sensors are undefined.
trackPos	Distance between the car and the track axis. The value is normalised w.r.t to the track width: it is 0 when car is on the axis and -1 or 1 when on the right or left edge of the track, respectively, and smaller -1 or greater 1 when outside of the actual track
wheelSpinVel	Vector of four sensors representing the rotation speed of the wheels.

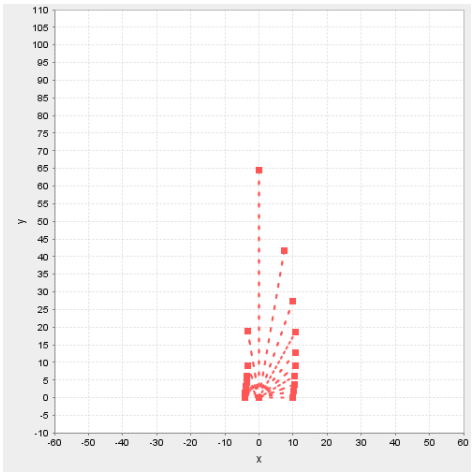
Table 6.1: Descriptions of available sensors

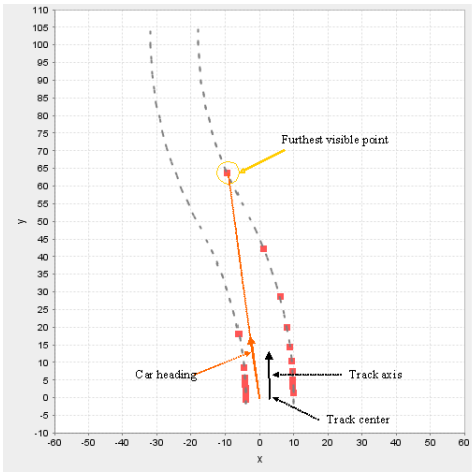
Name	Range	Description
accel	[0,1]	Gas pedal command (0 means no gas, 1 full throttle).
brake	[0,1]	Brake pedal command (0 means no brake, 1 full brake).
gear	[-1,0,...,6]	The gear to switch to or to stay in if the current gear is specified
steering	[-1,1]	Steering scaled between full left (-1) and full right (1), which corresponds to an absolute angle of 0.785398 rad.

Table 6.2: Effectors that can be used to control the car in the simulated car racing competition.



a





b c

Figure 6.1: Track sensors - a) Screen-shot b) Original track sensors c) Track-perspective view

However, there is usually a lack of information to model the segments that the car is approaching. Figure 6.2 shows an example of a car currently on a straight segment approaching a corner from a distance. What actually shows on the sensors are two vertical lines on the left and right hand sides of the car, and a single point in front of the car which belongs to the incoming corner. No matter how sharp or shallow the incoming corner is, from a distance, the car will always see maximum one point of the corner regardless of the position of the car on the track. From a single point, it is not possible to determine the direction of the incoming corner, let alone its radius. Only when the car is getting much closer to the corner can it see enough points from the turn to estimate its direction and sharpness. For a sharp corner, it is usually too late for the car to prepare itself in order to make the turn. As a consequence, unless the car is travelling very slowly, there is no guarantee that the car can react fast enough to make the corner. To overcome the problem of lacking data, a method is proposed that makes the most out of available knowledge and resources by combining the current available sensor data with previous estimations of the track. By taking previous knowledge of the track into account, not only the accuracy and reliability of the models improve, but also the computational time is significantly reduced. The track modelling is conducted in four important steps:

1. Localisation: maps the current track sensors into a local track-perspective coordinate system with the car at the origin, the vertical axis parallel to the current track axis and the horizontal axis passing through the track centre.

2. Mapping: maps the track segments learned in the last step into the local coordinate. In other words, this step is to answer the question: "Where are those segments the car has seen before?"

3. Segment Re-evaluation: verifies whether the current sensory data can be fitted into the current segments and makes further adjustments if necessary.

4. Modelling: creates new segments which fit the remaining sensory points.

The following of this section describes the segment representation and provides details about each step in the proposed track modelling approach.

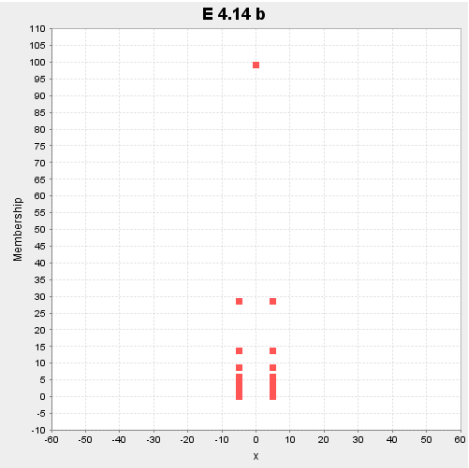
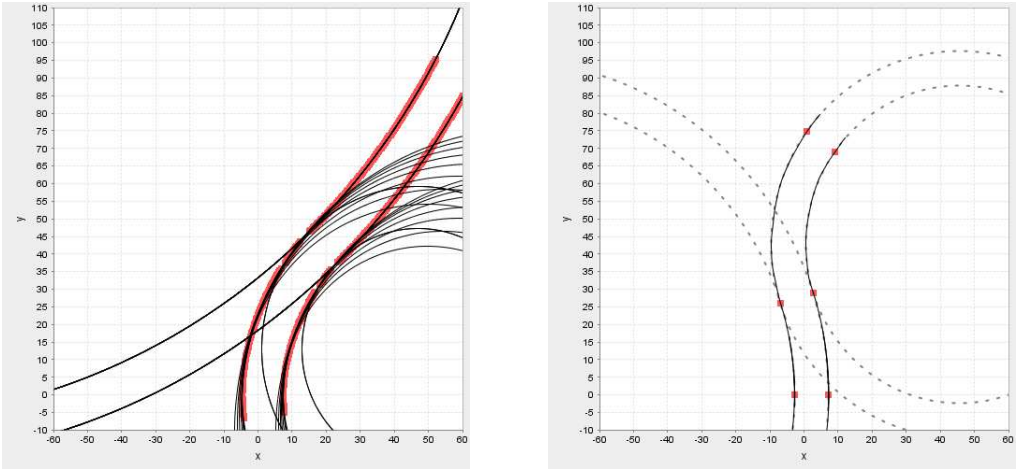


Figure 6.2: Approaching a corner from a straight segment

6.3.1 Track segment

In TORCS, the track is represented as a sequence of connected segments, which can be either straight, left or right turn. Each segment is completely defined by the coordinates of the four corners as well as the width and length (arc) covered by the segment. This segment representation allows the representation of any complex curve as a series of much simpler curves (each could be a line or circle) pieced together at their endpoints. (see Figure 6.3a) To ensure visual smoothness of the curve, the segments on either side of an endpoint on the curve must not only touch each other but also share a common tangent direction at the join point (tangential continuity G^1 [136]) . However, because each segment is either a line or a circle segment, when two segments touch each other (circle-line or circle-circle), there is always a common tangent line at the touching point. Therefore, the only requirement to ensure smoothness of the curve is that adjacent segments must touch each other at their endpoints (Figure 6.3b).

Our segment representation is very similar to that of the original TORCS in that each segment provides information about the direction of the turn, its radius and its start and end points. Information provided by a segment corresponds to the middle track line segment. The left edge and right edge curves are simply parallel curves which are displaced from the base middle track curve by a constant offset value of half the width of the track, both positive and negative, in the direction of the curve's normal. In regular TORCS, the track data is exact, so the left, right edge and middle track segment



a) Complex curve represented as a series of circles

b) Segments touch each other at their endpoints

Figure 6.3: Segment representation

can all share a common data structure. In our problem, two separate structures are adopted: *Segment* for the middle segment and *SideSegment* for edge segments. The radius of the middle segment must be exactly half of the total radius of the left and right segments. We also further require that radius of the middle segment must be an integer value and any radius value higher than 1000 will be considered a straight segment. Unlike in TORCS, the left and right segments in our representation have different centres although they all represent the same turn. The reason for that is that the correctness of the radius is never guaranteed, so the centre must be adjusted differently to best fit the current visible points on the edge. However, the distance between the centre should be small. Another point is that the edge segments are tightly coupled. If either a left segment or a right segment can be determined, the other edge segment as well as the middle track segment can be easily derived using simple vector calculations. If one of the edge segments is updated, e.g., change the starting or end point, or change the radius, etc., the other edge segment and the middle segment must be updated accordingly. Please refer to Table 6.3 for a comprehensive descriptions of these structures.

6.3.2 Localisation

Localisation is a task of estimating the pose of the robot relative to a map. In other words, the robot has to answer the question, "Where am I?". The client knows two

Segment	
Fields	Descriptions
type	0: straight, 1: right, -1: left
radius	0: straight, >0 : turn
start	Location of visible start point
end	Location of visible end point
upper	Location of estimated end of the segment
lower	Location of estimated start of the segment
leftSeg	Link to a SideSegment object representing the left segment
rightSeg	Link to a SideSegment object representing the right segment
map	A map between an radius and the number of its occurrences during estimation

SideSegment	
Fields	Descriptions
type	0: straight, 1: right, -1: left, 2 : unknown
radius	0: straight, >0 : turn
start	Location of visible start point of this segment on the edge
end	Location of visible end point of this segment on the edge
upper	Location of estimated end of segment
lower	Location of estimated start of segment
num	Number of points visible within this segment
points	Link to an array that stores the all points on this edge
startIdx	Index of the first visible point in the array points
endIdx	Index of the last visible point in the array points
opp	Link to the other edge
map	Link to the map in Segment

Table 6.3: Segment representation

important pieces of information which can be used to determine the position and pose of the car relative to the track, i.e., the angle between the car orientation and the direction of the track axis by the input variable *angleToTrack*. From the car-perspective track sensor s_i, it is fairly simple to transform the sensory view of the track into a local track-perspective coordinate system with the car at the origin, the vertical axis parallel to the current track axis and the horizontal axis passing through the track centre (see Figure 6.1b and c for more details)

$$\vec{s_i} = \begin{pmatrix} \cos(\pi - i \times \frac{\pi}{18} - angleToTrack) \times s_i \\ \sin(\pi - i \times \frac{\pi}{18} - angleToTrack) \times s_i \end{pmatrix} \tag{6.3.2.1}$$

where $i \in \{0, ..., 18\}$.

The coordinate of the vector $\vec{s_i}$ is the location of a point on either edge of the track if the length of $\vec{s_i}$ is less than 100m. $\vec{s_0}$is the leftmost point and $\vec{s_{18}}$ is the rightmost point with respect to the car orientation. The angle between $\vec{s_i}$ and $\vec{s}_{i-1}$ is 10 degree.

Let h be the furthest distance of all points and $s_{h_1}, ..., s_{h_n} \in \vec{s}_i$ be the points which has length of h.

Any point whose distance to the car is less than h must definitely belong to either the left or the right edge. If a line is drawn between the car location (the origin) and any of the furthest points, the set of points that must belong to edges is divided into two regions (see Figure 6.1c). It can easily be seen that points on one side of the line must belong to one edge and the rests belong to the other edge. If the car is heading up, in the same direction as the track axis, then points on the left of the line belong to the left edge and point on the right side belong to the right edge. To find out which edge the furthest points belonging to, two possibilities are considered:

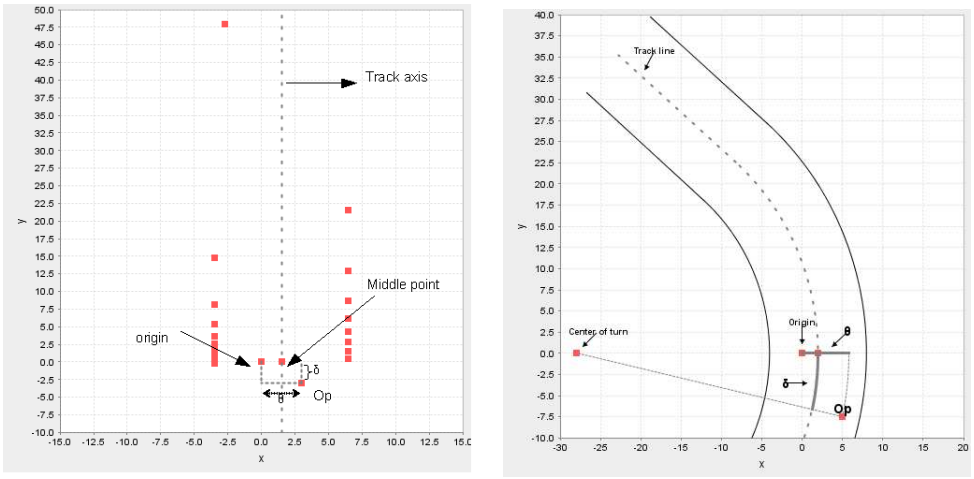
- $h \geq 100$: All the furthest points are "open" space which mean they do not belong to either edge.
- $h < 100$: All the furthest points must belong to either the left or the right edge. Due to the nature of the track sensor, it is impossible to have two points of the same distance to the car that belongs to the same edge unless one of the two lies below the x-axis, which can be safely ignored. Therefore, if $h < 100$ then there is only one furthest point. Unfortunately, only the exact edge that the point belonging to in some special cases can be determined, e.g., when the point is "close" to another known point. In some cases, guesses can be made based on the current direction of the visible turn. In all other cases, it is not possible to determine the edge to which the point belongs.

At this point, the track sensor inputs have been classified into corresponding edges. Henceforth, $left[i]$ and $right[i]$ are two collections sorted in ascending order of height of the points on the left and right edges respectively. The remainder of the track modelling task is to find a sequence of segments (middle track segments) so that their corresponding left segments fit the points on the left edge as specified by $left[i]$, and consecutively

their corresponding right segments must also fit the points on the right edge as specified by $right[i]$. If there are multiple sequences that match the current visible points on the edges, the solution which satisfies tangential continuity is favoured.

6.3.3 Mapping

This task is an attempt to extract the features or landmarks of the track which the bot has observed before from the current view of the car. Assuming that at the previous time step, the track has been estimated as a series of n segments $p_0, ..., p_{n-1}$ with their corresponding left segments $pl_0, ..., pl_{n-1}$ and right segments $pr_0, ..., pr_{n-1}$. The goal is to find the estimated locations of these segments in the current local coordinate knowing that the car has moved a distance along and across the track axis. In order to be able to map between the previous and current local coordinates, one needs to know the displacement difference between them. The input variable *distRaced* gives the distance covered by the car from the beginning of the race. The distance along the middle track line the car has travelled from the previous to the current time step is $\delta = distRaced - prevDistRaced$. The input variable *trackPos* specifies the distance between the car and the track axis normalised with respect to the track width. From this information, the real distance from the car to the track axis is given by $toMiddle = trackPos * trackWidth * 0.5$. From the previous to the current time step, the car has moved an offset of $\theta = toMiddle - prevToMiddle$ away from the track axis. There are two possibilities that could happen: i) the car stays in the same segment ii) the car has moved from one segment to another during the time step. The latter case can also be thought of as two consecutive same segment movements: the car moves to the end of the last segment, and subsequently moves from the start of the current segment to the current position. However, it is not always possible to work out the distance the car has travelled in each segment every single time. In some situation, especially when the previous segment estimations are not reliable, the best one can do is to use any acceptable distances. Fortunately, accuracy is not the most important aim of the mapping task because the segments will be verified and adjusted later on in the track modelling process. For brevity's sake,only the techniques for the former case are



a) Straight segment b) Turn segment

Figure 6.4: Mapping previous segment

considered.

6.3.3.1 Same segment mapping

If the previous first segment and the current estimated segment are the same (same direction and radius) as in Figure 6.4, the mapping function f can be formulated as follows

- The current segment is a straight segment:

$$f(x, y) = (x + \theta, y - \delta) \tag{6.3.3.1}$$

- The current segment is a turn of radius r and centre (cx, cy):

$$f = T.R \tag{6.3.3.2}$$

where R is a rotation transformation around the centre (cx, cy) an angle $\alpha = \frac{type \times \delta}{r}$ and T is a translation transformation $T(x, y) = (x + \theta, y)$

To map a segment into local coordinates, the mapping function f is applied to every point contained in the segment including the start, end, centre, lower and upper points. In the case of the first segment being straight, the mapping function is relatively reliable. Only in this case can all the sensory points on the edges be mapped to the current

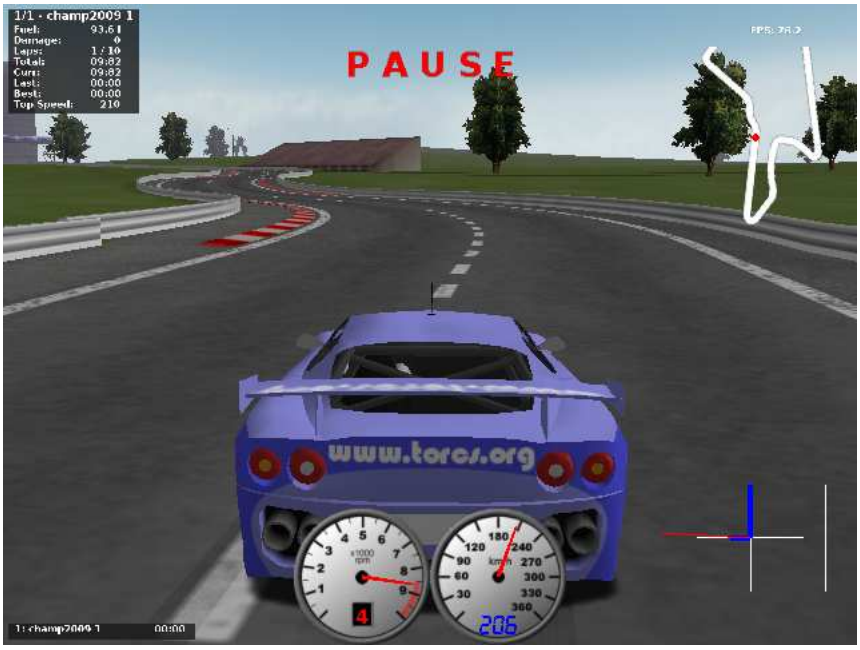
view and used as additional sensory points. Figure 6.5 shows an example of combining sensory data when the bot is travelling in the same straight segment. The example clearly shows that the combined data (as shown in figure 6.5 c) provides much more meaningful information about the incoming corners than the original sensory data (as shown in figure 6.5 b).

6.3.4 Segment re-evaluation

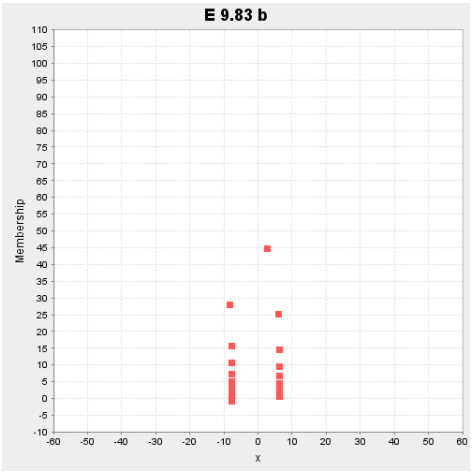
Due to the unreliability of segment mappings from the previous time step into the local coordinate, it is important to verify whether these estimations agree with the current track data. This task is to re-evaluate all mapped segments using current sensory points and to update the segments if necessary. Consider a segment s with its left and right side segments l and r respectively. From the track data on the left and right edges ($left[i]$ and $right[i]$ respectively), it is trivial to find the points that lie within the start and end points of l and r. Let $l[i]$ and $r[i]$ be the points belonging to the l and r respectively. From only the points in $l[i]$ and $r[i]$, new segments are created so as to best fit those points using the methods that will be described later in the modelling task. Note that the segment s has a data structure map which stores the frequency of all radii that the segment has been estimated with. The radii of the newly created segments will be recorded into the map of segment s to indicate that the current segment s could also be estimated with the recorded radius. Once a radius is inserted into the map, a corresponding number of occurrences is increased by one and the radius which has the highest frequency of appearance will become the current segment radius, and the centre, start and end points of the segment will then be updated accordingly. Thus, if there is any change in the radius of s, the side segments l and r will also be updated accordingly.

6.3.5 Modelling

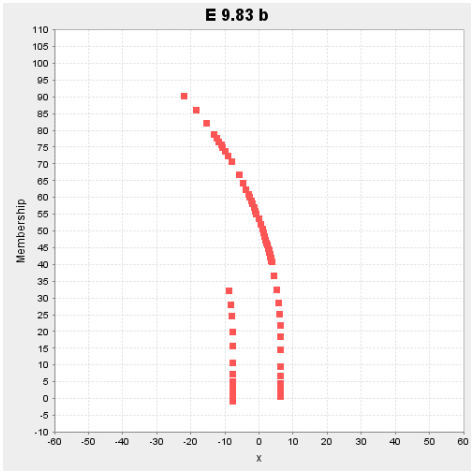
The main aim of the modelling task is to find a sequence of segments which not only best fits the current set of data points, but also maximises the smoothness of the track



a) Screen-shot



b) Original inputs



c) Combined with previous points

Figure 6.5: Combining data in straight segment

curves. The modelling is performed by finding a set of curves which fits the free points (points which are not yet assigned to a segment) on one of the two edges. Figure 6.6a) shows an example of a portion of a track which has been partially modelled as a series of three segments with three groups of free points in between the segments (grouped in circles). Note that, each segment is shown as a pair of double-arrowed curves (line or circle) on each side of the track. Straight segments must be two parallel lines on both sides while each circle segment must be a pair of circle arcs of the same centre and different only in radius. For every curve found from the free points, the curve to the other edge is mapped and checked whether or not the mapped curve also fits the actual points which are supposed to be on it. A curve which satisfies this condition is probably a good estimation and consequently a new segment is created to represent the curve. Figure 6.6b) shows that two new segments (in blue) which fit the free points have been created for the sample track. The modelling algorithm can be described in the following steps:

Step 1: find all the groups of free points from both edges.

Step 2: for each group, find the adjacent segments right before and after the free points on the same edge.

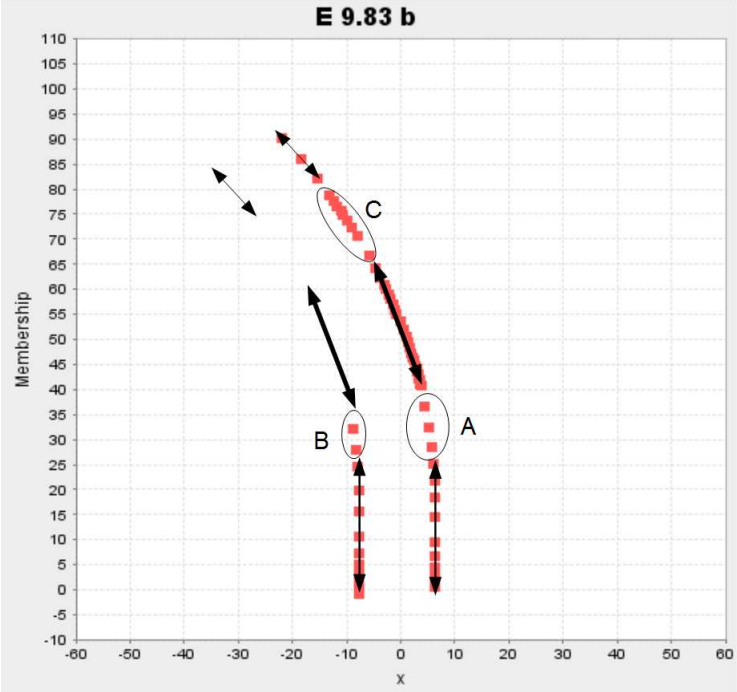
Step 3: find the best fit curve which maximises the connectivity with the adjacent segments using the methods described later on.

Step 4: if a curve is found, map the curve to the other edge and check whether the existing points on that edge are within a very small distance to the mapped curve. If the condition is satisfied, go to step 5, otherwise repeat step 3.

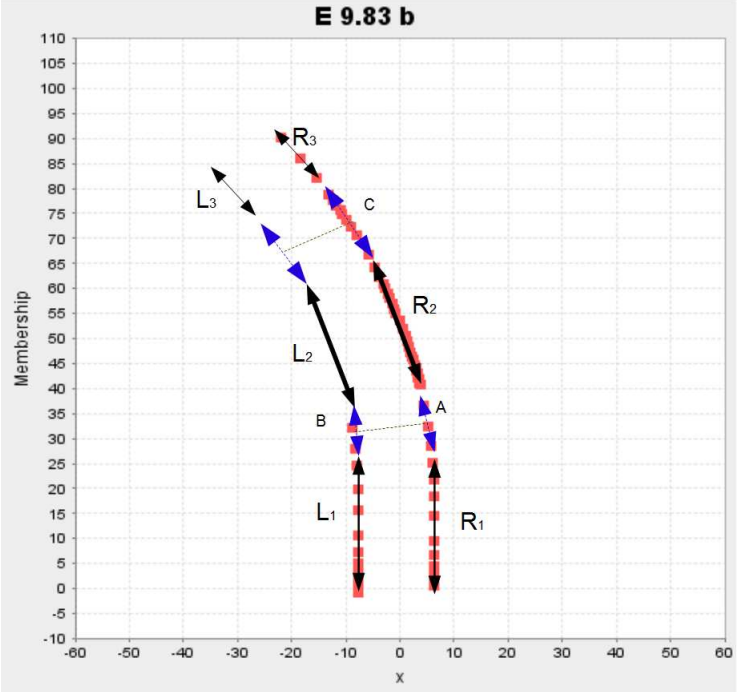
Step 5: create a segment corresponding to the found curve and record its radius into the private map of the segment

Step 6: If there are free points on the opposite edge curve, the techniques in step 3 are adopted to find the best fit radius and update it into the existing segment. Note that the radius of the segment will always be the one which has the highest number of occurrences in the private map of the segment.

Step 7: repeat step 1, until all the free point gaps are checked.



a)



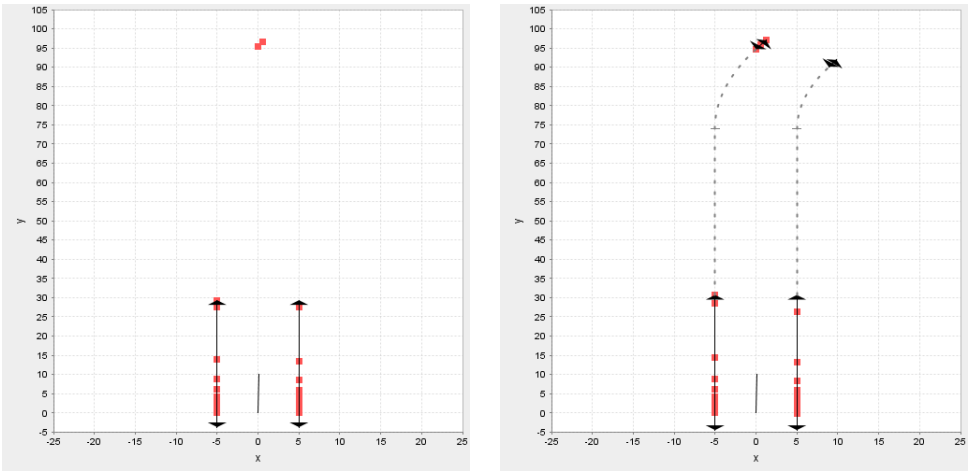
b)

Figure 6.6: Segment modelling a) before and b) after

The example from Figures 6.6a) and b) demonstrates the idea of the algorithm. As the first step, free points are classified into three groups A, B and C as shown. In step 2, let us start from the first free point group, i.e., group A. The adjacent segments that were previously estimated of A are denoted as R_1 and R_2 as shown in Figure 6.6b). In step 3, assume that a circle α is found (using the methods described later) which matches the points in A while maximising the continuity with R_1 and R_2. In step 4, as shown in Figure 6.6b), the mapped circle β of α on the other edge contains the free points in group B and the distance from these points towards β is very small. As the condition in step 4 is satisfied, proceed to step 5 to create a segment γ which corresponds to the circle arcs α and β on both sides. The segment has the same centre as α and β and its radius is exactly half of α and β. The number of occurrences of the radius during the best fit curve algorithm is inserted into the segment map. As shown in Figure 6.6b), two free points in group B also belong to β. Thus proceed to step 6, which is essentially a repetition of steps 3 to 5 for the free points in group B. However, instead of creating a new segment, the same segment γ is used and the radius of the curve estimated from the points in B is inserted into the private map of γ. At all times, the current radius of a segment has to be the most frequently estimated value during past and present estimations from both edges of the same segment. After a new segment is created and adjusted, new free point gaps are found. The whole process is repeated, until all of the free point groups are checked or cannot be found. The following elaborates on the most important algorithm which is used in step 3, i.e., finding the best fit curve from a set of points while maximising the tangential connectivity with the adjacent segment curves if possible.

6.3.6 Finding the best fit curve

The best curve fitting algorithm aims at not only constructing a line or a circle that best fits a series of data points, but also tries to achieve the visual smoothness when connecting the curve with its previously known adjacent curve segments. Note that, there can be from zero to two adjacent segments for any given set of data points. As aforementioned, an important property must be satisfied to a ensure visual smoothness when



a) b)

Figure 6.7: Segment from two points

connecting the two curves, i.e., the curves must not only touch but also share a common tangent direction at the join point (tangential continuity G^1 [136]). This property is particularly important since it allows a good estimate of the best fit curve from as little as two points based on the adjacent segments. Consider an example from Figures 6.7a and 6.7b. In Figure 6.7a, a straight segment is followed up by a right turn of an unknown radius with only two visible points. It is clear that there could be an unlimited number of curves (circles or lines) passing through the two points. However, among those curves, there are only one or two which satisfy the aforementioned property, i.e., the curves must touch the line created by the straight segment. In this example, the radius of the circle passing through the two points creating a new segment can be easily deducted, as shown in Figure 6.7b. While there is no guarantee that the two points indeed belong to the same segment next to the straight segment, the guess is correct most of the time. At the same time, it is usually preferable to provide a rough approximation early rather than to wait for an accurate but late estimation. The method to find the best fit curve through two points is described in Appendix B. Please note that this method can only be applied in cases where there is at least one known adjacent curve. There are two other scenarios in which the curve through two points can be reliable estimated. The first is when the points belong to the first segment where the car is located (refer to Appendix C for details). The other is when there are more points in between the two points in consideration. This is based on the fact that if two points belong to the curve

segment, then every other point in between them must also belong to the curve. The radii can be easily computed for all circles that can be built using all triplets consisting of two points in consideration and a point in between. If a circle has a very large radius (>1000), it is considered a line segment and is subsequently set a radius value of 0.

Using the above methods, multiple potential curves which pass through the two points can be found. If there exists a radius which appears in multiple estimations, it is likely that the radius is the correct one. Otherwise, unless there are fewer than three solutions, it is possible that the two points in consideration do not belong to the same segment. The methods could be extended to find the best fit curve through a series of points as described in algorithm 6.1.

Algorithm 6.1 Finding best fit radius

```
function bestGuess
 Input:

    points : list of points
    num : number of points
    prev,next : previous and next segment if any
    map : array stores radius's frequency

Output:

    best fit segment

begin

    for i:= 0 to num-2 do
    begin
        v←points[i]
        for j:=num-1 down to i+1 do
        begin
            q←points[j]
            if (is_first_segment(v,q))
              then return new_segment_from(C,v,q)
            C1 ←curve_from_2_points(prev,v,q)
            C2 ←curve_from_2_points(next,v,q)
            if (C1!=null) then map[C1.radius]++;//favour radius of C1
            if (C2!=null) then map[C2.radius]++;//favour radius of C2
            for k:=i+1 to j-1 do
            begin
              m←points[k]
              C←circle_three_points(v,m,q)
              if (C!=null) then
               begin
                r←C.radius
                map[r]++;
                if (map[r]>THRESHOLD) then
                 return new_segment_from(C,v,q)
               end
            end
        end
    end
    r←most_frequently_appeared_radius(map)
    C←find_curve_with_radius(r,points)
    return new_segment_from(C,v,q)

end
```

A very good matching is generally observed between the learned and the actual track

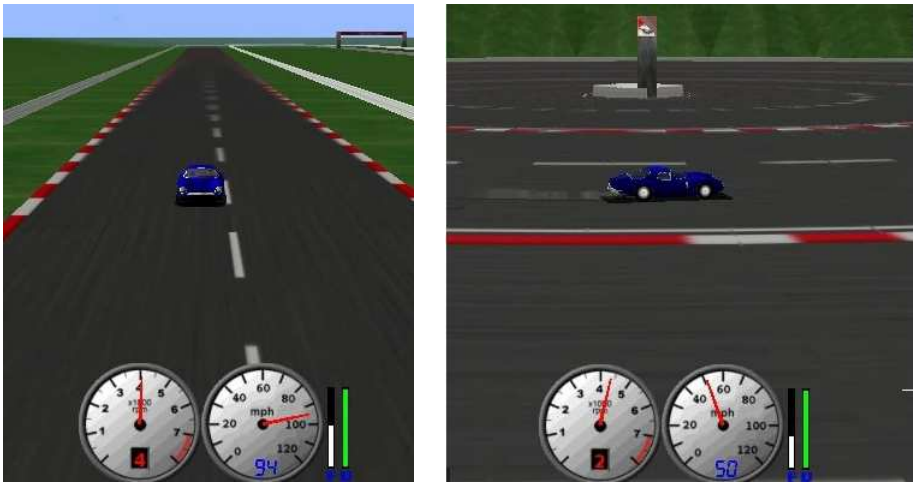
models used by TORCS. If the test tracks have a lot of turns with integer radii, the accuracy could be over 95%. However, due to space limitations, the results are not presented here.

6.4 Employing the track model

The last section proposes a track modelling technique which converts the sensory data into a sequence of straight or circular segments. In this section, how the models can be used to construct an efficient and easy to understand SCRC bot with planning and adapting capability is demonstrated. The controller consists of two main parts: the planning and the execution modules. Firstly, the planning module examines the current track model to plan a feasible path that the car should follow. The aim of the planning path is to not to find an optimal path, which could only be locally optimal at best, but to maximise visibility while still being able to deliver good performance. The output of the planning module is a point location to which it suggests the car to go, the corresponding steering to the point, and the speed that the car should maintain at the current situation. Secondly, the execution module takes the suggested guidance of the path planner, but decides whether to follow the suggested solution based on the current state of the car. The execution module is implemented as a context-dependent fuzzy controller with the aim of keeping the car stable while still being able to drive towards the suggested point. Before detailing the design and implementation of each module, the basic and fundamental information that every controller needs to know in order drive safely will be introduced.

6.4.1 Learning fundamental driving controls

In order to drive fast and safely, one needs to determine when to shift gears and figure out the maximum speed at which the car can safely negotiate a turn.



a b

Figure 6.8: Test tracks used for developing driving algorithms

6.4.1.1 Gear shifting

Shifting gears simply depends on the current speed of the car. We do not have direct access to the torque vs. RPM curve for the engine of the virtual car. However, the relationship between the speed and *rpm* of the car can be indirectly uncovered as follows.

$$speed \approx rotationSpeedOfWheel \times wheelRadius \tag{6.4.1.1}$$

and

$$rotationSpeedOfWheel \approx \frac{rpm}{gearRatio} \tag{6.4.1.2}$$

where $gearRatio$ is the ratio between the current *rpm* of the car and the rotation speed of the wheel for each gear.

Therefore

$$speed \approx \frac{rpm \times wheelRadius}{gearRatio} \tag{6.4.1.3}$$

In order to find the optimal shifting point, we need to calculate the optimal *rpm* for each gear. According to our experiments, it is best to shift gear when the car engine reaches close to the full spectrum of its gear. The maximum allowed rpm of the car can be seen from its description. We choose to shift gear when the rpm reaches 95% of the

maximum rpm allowance. The final formula for shifting gear is

$$shift[i] = \frac{enginerpmRedLine \times 0.95 \times wheelRadius}{gearRatio[i]} \tag{6.4.1.4}$$

with $i \in [0, maxGear]$; $enginerpmRedLine$: maximum value of rpm (958.395); $wheelRadius$: the radius of the wheel in meters (0.3276)

The gear is shifted up when $speedX > shift[currentGear + 1]$

The gear is shifted down when $speedX < shift[currentGear]$

The gear-ratio for each gear is calculated by building a custom map which is simply a long straight segment (see figure 6.8a). The car can go flat-out on the track. For each gear, the actual speed and the actual rpm are measured, and the estimated gear Ratio is computed according to formula 6.4.1.3. Finally, the gear Ratio for each gear is:

$$gearRatio = \{-9.0,\ 0,\ 13.5,\ 8.55,\ 6.3,\ 4.95,\ 4.05,\ 3.465\} \tag{6.4.1.5}$$

.

6.4.1.2 The relationship between speed and radius of turn

In order to drive safely, one needs to know the maximum speed at which the car can safely negotiate a turn for a given radius. And also the minimum radius that the car can turn at the current speed needs to be determined. In other words, two functions are required: $speedAtRadius(x)$ and $radiusAtSpeed(x)$. A custom round track map was built with a very large width (see figure 6.8b). In order to find $speedAtRadius(x)$, a small controller which drives the car in a circle at radius x around a chosen origin was written. The speed of the car was gradually increased, and the maximum speed at which the car is still able to stay in the circle was recorded. Data on the corresponding maximum speed at a number of radii $(speed_1, radius_1), ..., (speed_n, radius_n)$ was collected. The relationship between speed and radius is derived from applying the best fit curve algorithm on the data set. Finally, we have:

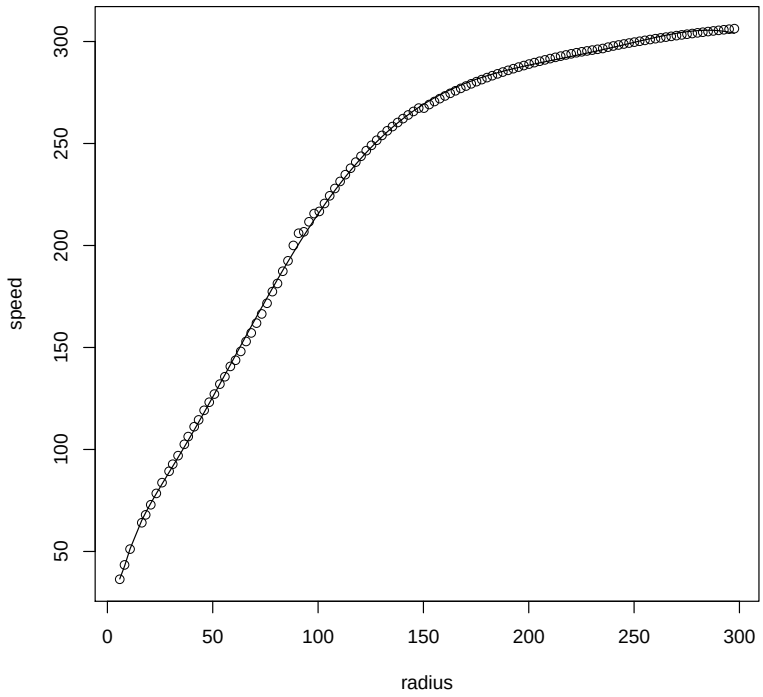


Figure 6.9: Speed at radius

$$speedAtRadius(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + \frac{b_1}{x} + \frac{b_2}{x^2} + \frac{b_3}{x^3} \quad (6.4.1.6)$$

and

$$radiusAtSpeed(x) = a'_0 + a'_1x + a'_2x^2 + a'_3x^3 + a'_4x^4 + a'_5x^5 + \frac{b'_1}{x} + \frac{b'_2}{x^2} + \frac{b'_3}{x^3} \quad (6.4.1.7)$$

The graphical representations of $speedAtRadius()$ and $radiusAtSpeed()$ are shown in Figures 6.9 and 6.10 respectively.

6.4.2 Driving strategy

Racing with other cars of the same type (car1-trb1) means that our controller has virtually no relative advantage over car engine powers or tyre traction. Every car construct is identical, the only difference between our controller and others is the ability to take

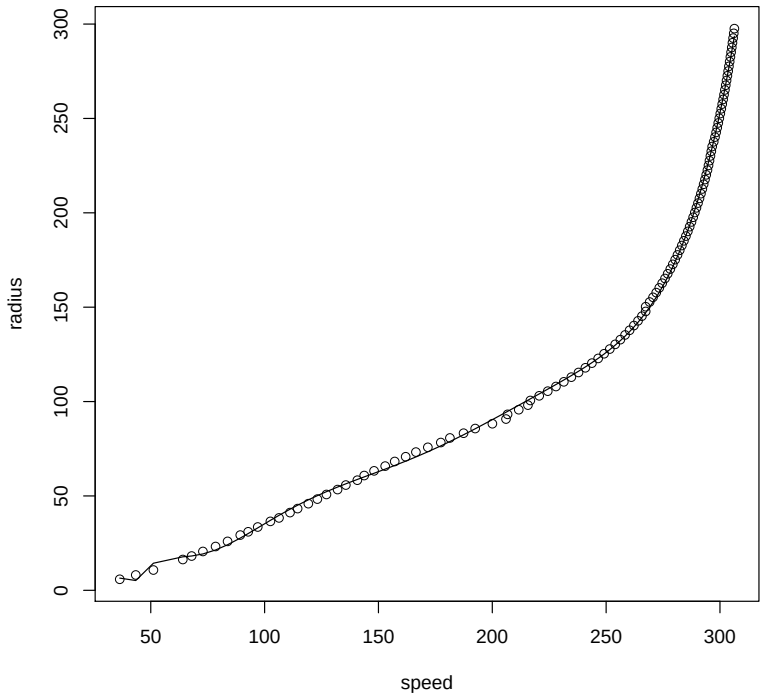


Figure 6.10: Radius at speed

corners. In order to win a race, a car has to take track corners in a minimum amount of time. From our observation, most cars in the regular TORCS adopt the "slow-in, fast-out" driving style, i.e., the car starts slowing down far from the turn, and takes the turn at a very reasonable, controlled speed. Most of the time, the driver applies the brake before the turn, then, releases it entirely when turning in. The racing lines taken by regular TORCS drivers are also optimal. Before entering the turn, the car positions itself on the track towards the outside edge of the coming corner. When committing

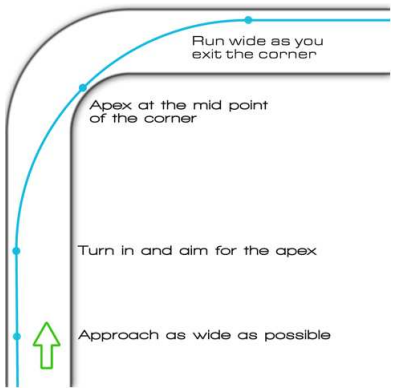


Figure 6.11: Traditional racing line

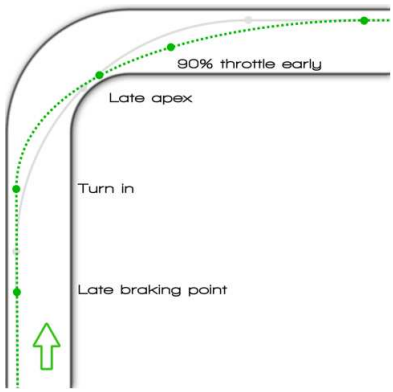


Figure 6.12: Late-apex racing line

the turn, the car follows a path that passes through the apex, which usually is on the innermost edge of the corner. Upon exiting the corner, the car move back to the outside edge of the track, following the widest path through the corner. (see Figure 6.11). This approach results in smoothing out corner turn in the most efficient way and reduces the likelihood of under- or over-steering.

In our approach, the "trail-braking" driving style is adopted. This involves applying the brake later and continuing to brake into the early phase of the corner before the apex. When taking a turn, our car will take a path similar to that of the traditional "slow-in, fast-out" approach, in which, the car is positioned to the outside edge before taking the corner, then takes the corner via the path through the apex, and finally exits the turn following the widest path through the corner. The only difference is that the turn-in point of our car is later than that of the traditional approach, which results in a late apex racing line as shown in Figure 6.12. This approach results in a slightly slower entry speed but a faster exit speed. There are several main advantages for our controller in following this approach:

1. Our car does not have global information of the track. Instead, the car relies only on the estimated segments of the points on the edge. This means that most of the time, our car will detect the turn late and therefore, the car will have passed the regular turn-in point, forcing it to take the late apex path.

2. By carrying speed into the turn, we can reduce the unnecessary slowing down of the car. Some turn might initially appear to be sharp, but it turns out later

on that the sharp part is in fact very short and is followed by a high speed turn. In the traditional approach, the car in this scenario would have to slow down unnecessarily when it first detects the sharp part. The regular bots in TORCS with global knowledge of the track have a definite advantage in this situation, as they can detect and work out the optimal speed to get through this kind of turn easily.

3. The car car1-trb1 has a natural tendency to under-steer which means that when taking a corner at high speed, the car has a propensity to go straight on, instead of round the corner. Carrying a slight brake into a corner can provide additional grip at the front wheels, therefore reducing the understeering problem.

4. When taking the corner at high speed, if the driver focuses on slowing down the speed by braking too hard, he exacerbates the under-steer problem and lower the likelihood of keeping the car on track.

5. A major part of braking should still be completed prior to the turn. However to maximise the car performance, our braking point is slightly delayed and the brake is continued to be in use in the corner prior to the apex.

This trail-braking approach can help improve the time taken to complete moderately sharp turns (less than 90 degree turns), which we believe account for the majority of the turns. However, it puts the car grips to the limit and fails to have a robust performance on longer sharp turns, e.g., hairpins.

6.4.3 Control architecture

Controlling a car to race in a dynamic, partially known environment requires both high-level reasoning capabilities and reactivity (to be able to deal with unexpected events in a timely manner). Accordingly, a hybrid control architecture is chosen for it fits the "perception-decision-action" paradigm as shown in Figure 6.13. The decision component consists of three main modules: *context analysis*, *path planner* and *executioner.*

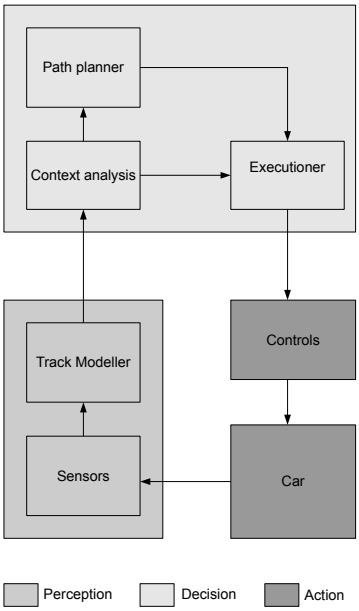


Figure 6.13: Control architecture of the driver

The context analysis module has two main purposes:

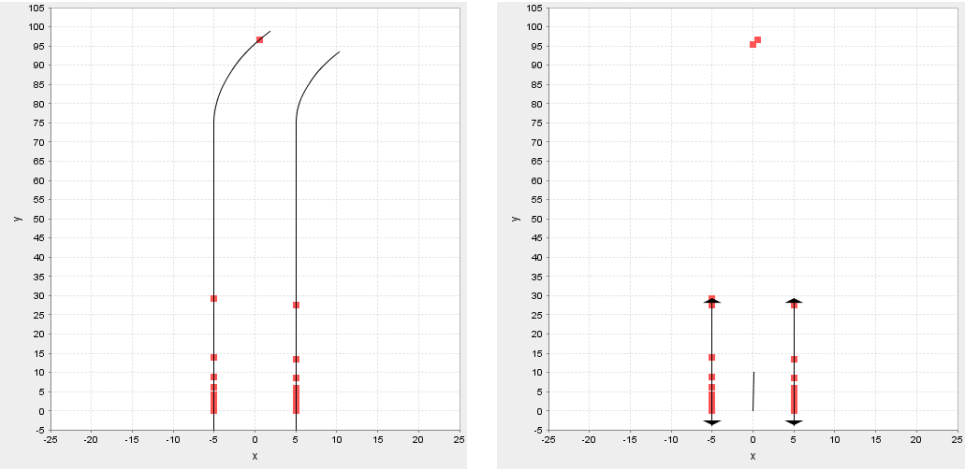
- It provides prediction of the possible turning direction of the track.
- It signals both the path planner and the executioner of the current situation of the car, e.g., approaching a turn, at extremely high speed or, exiting a turn, etc.

The main purpose of the path planner is to produce a suggested path that the car should follow based on the current situation of the car and the geometric shape of the track. The fuzzy-based executioner will then make the final control decision based on the current situation of the car and the suggested solution from the path planner.

6.4.4 The context analysis module

The first important task of the context module is to determine the immediate coming turn. There are two cases:

1. The current segment is straight. If the distance from the highest visible point to the origin (the car location) is greater or equal to 100 metres, the car is not approaching a turn. Otherwise, if there is only one visible highest point that does not belong to any edges (see Figure 6.14a), the incoming turn is unknown. If there



a) Approaching a turn . b) The turn is a right turn.

Figure 6.14: Determine direction of incoming turn from the current straight segment

are at least two points that do not belong to any edges (see Figure 6.14b) and the distance the two points is fairly close, the incoming turn can be estimated by

$$turn = signum((highestPoint.y - ph.y) * (highestPoint.x - ph.x)) \qquad (6.4.4.1)$$

where $highestPoint$, ph are the location of the two points. $signum(x)$ is the sign function of x. And the edge to which both $highestPoint$ and ph belong is $which = -turn$ ($which = -1$ for left edge, 1 for right edge, 0 for unknown).

2. The current segment is a turn. If there is only one visible highest point that does not belong to any edges, the incoming turn is indeterminable. In this case, the direction of the coming turn is the direction of the last estimated segment. Let $highestPoint$, ph be the location of the highest visible point at the current step and previous step, respectively. If the distance between $highestPoint$ and ph is fairly small, the coming turn can be estimated as

$$turn = signum((highestPoint.y - ph.y) * (highestPoint.x - ph.x)) \qquad (6.4.4.2)$$

The edge to which both $highestPoint$ and ph belong is $which = -turn$ ($which = -1$ for left edge, 1 for right edge, 0 for unknown).

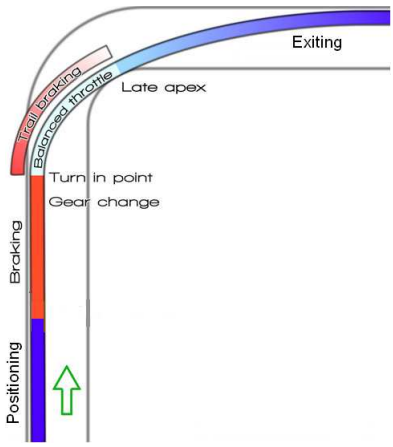


Figure 6.15: Context analysis

When the direction of the coming turn is determined, the car will prepare to take the turn. The corner taking process can be divided into four stages as shown in Figure 6.15. Depending on the current position and state of the car, the current phase of the car can be determined. The current phase is considered as context information of the car with respect to the current corner the car needs to take.

Determine current zone of the car

Once the coming turn is determined, the second task of the context analysis is to divide the coming corner into distinct zones (see Figure 6.15) and determine which zone the car currently belongs to.

- Positioning zone: prior to cornering.
- Braking zone: the car focuses on getting the right cornering speed.
- Gear change: optional. Normally, the car changes gear according to its current speed. However, in some situation, the gear can be changed manually so as to perform a special technique such as drifting, over-steering, or balancing.
- Turn-in point : point where the car starts making the turn.
- Trail braking/balanced throttle: the car is committing to the turn. In this zone, turning has more priority than braking. The aim is to navigate the car to the apex point.

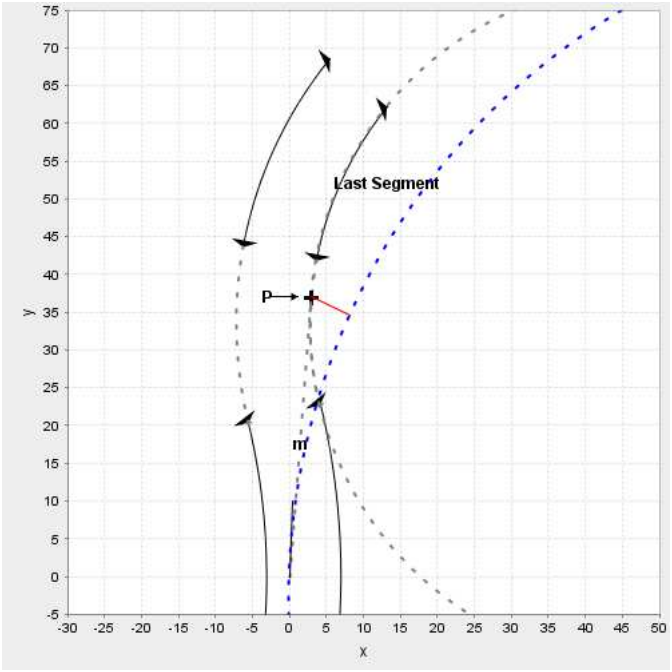


Figure 6.16: Context variables

- Apex zone: if turning at a faster speed than the safe speed of the corner, the car tries to stay at the apex zone for as long as possible. In this zone, braking is reduced to the minimum.
- Exiting zone: turning at a high speed will result in sliding of the car when exiting the corner. The focus of the car in this zone is to remain on the track, reduce speed if necessary to increase balance and stability, and resume to acceleration when possible.

Information of the distance of the car to the turn, the current heading and movement of the car with respect to the turn is required in order to differentiate between zones. The following context variables are created to hold information of the car states with respect to the current direction of the turn.

- $relativePosMovement$: the difference between current and last position of the car w.r.t to the direction of the last segment turn. $relativePosMovement > 0$ means that the car is moving towards the turn.

$$relativePosMovement = turn \times (curPos - lastPos) \tag{6.4.4.3}$$

- $relativeAngle$: angle between the car direction and the direction of the track axis w.r.t to the direction of the last segment turn. $relativeAngle > 0$ means that currently the car is heading towards the turn.

$$relativeAngle = turn \times angle \tag{6.4.4.4}$$

- $relativeAngleMovement$: the difference of $relativeAngle$ from the last and current time. $relativeAngleMovement > 0$ means that the head of the car is turning towards the curve.

$$relativeAngleMovement = turn \times (angle - lastAngle) \tag{6.4.4.5}$$

- $relativeSpeedY$: speed of the car along the transverse axis of the car w.r.t to the movement of the car. $relativeSpeedY > 0$ means that the car is turning normally. $relativeSpeedY < 0$ means that the car has just changed its direction and the car has to steer slowly in order to keep balance.

- Apex point P: is the intersection point of the inner circle C of the last segment and the tangent line from the origin to C. See figure 6.16

- m : distance from the car to the turn (the distance between the origin and P). See figure 6.16 for more details

  - $distToCircle$: distance from the target point P to the circle β that the car makes at the current speed. The radius of β is equal to $radiusAtSpeed(speedX)$. β is the smallest radius the car can make if it continues at current speed. If P lies outside of β, the car can turn to P ($distToCircle > 0$). If P is inside of β, the car cannot reach to P at current speed. $distToCircle$ allows us to adjust the speed so that the car can get to the target point P. In figure 6.16, β is the blue dotted curve (circle), and $distToCircle$ is represented as the red segment.

- $angleToP$: angle from the current track heading vector to P.

- $safeSpeed$: the safe speed to take the turn, i.e.,

$$safeSpeed = speedAtRadius(lastSeg.radius) \tag{6.4.4.6}$$

From these variables, the current zones of the car can be roughly divided as follows:

1. Positioning zone : m must be at least 20m. If $distToCircle > 0$ or $angleToP$ is too small, the car needs to position itself so as to have a better angle when turning into the corner. If $distToCircle < 0$ and the current car speed is not too fast compared to $safeSpeed$, the car is in the positioning zone if the $distToCircle$ is greater than a threshold. If $distToCircle < 0$ and the current speed is too fast compared to $safeSpeed$, the car is not in the positioning zone.

2. Exiting zone: when $relativePosMovement < 0$ and the car is turning towards the turn. The negative value of this variable indicates that the car is currently moving towards the outer edge of the track (moving outward of the turn). There are two occasions when $relativePosMovement < 0$. The first occasion is when the car is in the positioning zone and needs to move to the outer edge for a better angle to take the turn; in this case, the car deliberately steers to the outer edge. The second occasion is when the car exits a turn. Although the car continues to steer towards the turn, it still moves towards the outer edge of the track. So if a car is moving outward while still steering towards the turn and the value $distToCircle$ is decreasing, it is quite safe to assume that the car is in the exiting zone.

3. Turn-in point +trail braking/balanced throttle : if $m < 20$ or $distToCircle < 0$ and is decreasing. The car needs to turn into the corner. In this zone, turning towards the apex point is the main target and braking is reduced to a minimal.

4. Apex point: If $distToCircle \geq 0$ and $m < 10$, the car is approaching the apex point. If $distToCircle < 0$ and $m < 10$, there are situations in which the car cannot reach the apex point of the turn, e.g., turning at a too high speed. In this case, the value of $relativePosMovement$ is examined to detect when the car reaches the nearest possible to the apex point. If $relativePosMovement > 0$ but the value of

relativePosMovement is decreasing and is currently less than a specific threshold, the car reaches the nearest possible to the apex point of the turn.

6.4.5 The path planner module

The main purpose of the path planner module is to suggest a trajectory that the car should follow based on the current estimations of the local track and the target speed the car should achieve in order to follow the path. If the coming turn is unknown or is anticipated to be in the reverse direction of the last visible segment (the car has not yet seen the new segment), the path is found as if the coming turn is the same direction as the last segment.

Trajectory planner

Depending on the current zone of the car, there are different trajectories for each zone. At each time step, a point on the track for the car to follow is to be found. Figures 6.17a,b,c detail the target point when the car in the positioning zone, braking zone and turning zone. The idea is simple. When the car is in the positioning zone (far distance from the turn), the car positions itself to the outer edge so as to maximise the visibility of the incoming turn. When the car gets closer (the braking zone), the car starts turning but does not fully commit to the turn. Only when the car gets very close to the turn does it commit fully to the turn. As aforementioned, the trajectory planner only indicates a suggested path based on the geometric shape of the turn and some limited information of the car state. Therefore, the suggested path is not meant to be optimal in terms of performance. It only gives a possible trajectory to take the corner with hopefully the most visibility.

Speed planner

The main purpose of the speed planner is to compute a suggested target speed so that the car can reach the target point. The expected speed at the apex point is $lastSpeed + \delta$ where $lastSpeed = speedAtRadius(radius\ of\ last\ segment)$. The car does not know

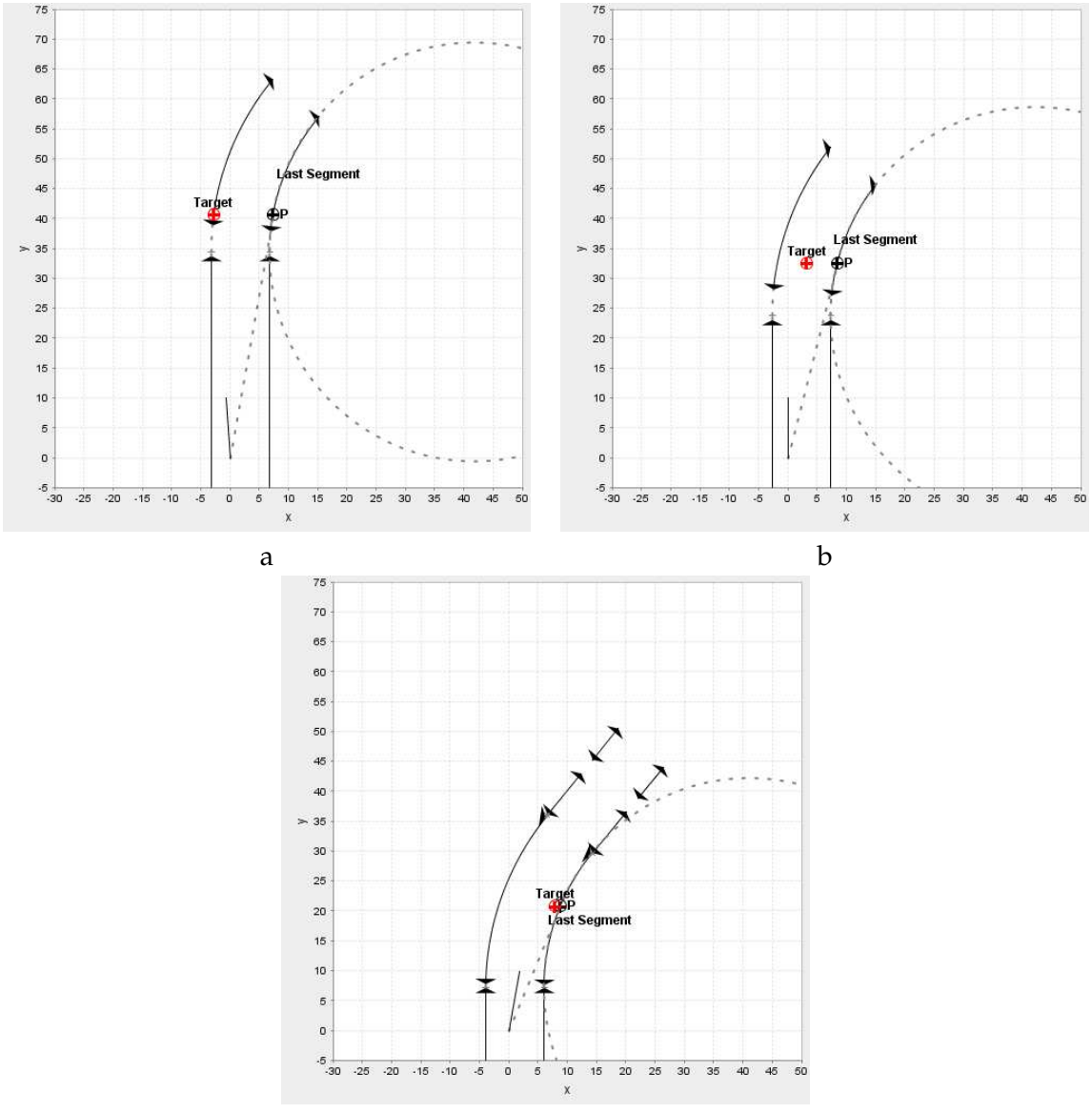


Figure 6.17: Target points. a) Car in positioning zone. b) Car in braking zone. c) Car in turning zone

what awaits after the last segment. The best it can do is to assume that the last segment will continue for some distance after the apex point. The last segment could be a hairpin turn in which the car has to slow down to around $lastSpeed$ to get through the corner, or it could be a 90 degree turn where the car can go at a much faster speed. It is assumed that most turns are less than 90 degrees so δ is chosen so that the car can go fast in most cases, and in the worst case of a hairpin turn, the car can slow down quickly enough to make the turn. δ is chosen as follows:

$$\delta = \begin{cases} 30 \; if \; lastSpeed > 100 \\ 15 \; if \; lastSpeed <= 100 \end{cases} \tag{6.4.5.1}$$

The $targetSpeed$ of the car should achieve at present is $targetSpeed = lastSpeed + \delta + m * \gamma$. The $targetSpeed$ indicates the maximum speed at the distance m to the target point so that it is possible to break down to the expected speed ($lastSpeed + \delta$) at the target point. γ is chosen as follows

$$\gamma = \begin{cases} 1.5 \; if \; m \geq 40 \\ 1 \; if \; m \geq 20 \\ 0.35 \; otherwise \end{cases} \tag{6.4.5.2}$$

6.5 The fuzzy-based executioner

The executioner consists of two components, the speed controller and the steering controller. The speed controller controls gas and brake commands; the steering controller, as the name suggest, controls the steering of the car. Although there is clutch control in the original TORCS, it is not included in SCRC. Both the speed controller and steering controller are designed as context-dependent fuzzy controllers whose membership functions are defined according to values of the context variables returned by the context analysis process. There are many input variables for the two controllers including all the variables mentioned in the context analysis, the target speed and target angle returned by the path planner modules, the $speedX$ of the car, and the rotation speed of each wheel, etc. For brevity's sake, only brief description of the rules and the strate-

gies implemented by the speed and steering controllers are provided. For a standard behaviour of the car (simply following the target point at specified target speed), the fuzzy rules of each component would look like:

- Brake rules (when $speedX > targetSpeed$):
  If $speedX$ is FAST then brake is HARD ($weight = W_1$)
  If $speedX$ is MODERATE then brake is MODERATE ($weight = W_2$)
  If $speedX$ is SLOW then brake is SLIGHT ($weight = W_3$)
  ,etc.
- Acceleration rules (when $speedX \leq targetSpeed$)
  If $diff$ is LARGE then accel is MAX ($weight = W_5$)
  If $diff$ is MODERATE then accel is MODERATE ($weight = W_6$)
  If $diff$ is SMALL then accel is SMALL ($weight = W_7$)
  ,etc.
- Turn rules:
  If $targetAngle$ is ZERO then steer is ZERO ($weight = W_1$)
  If $targetAngle$ is POSITIVE then steer is POSITIVE ($weight = W_2$)
  If $targetAngle$ is NEGATIVE then steer is NEGATIVE ($weight = W_3$)
  ,etc.

In context-dependent fuzzy controllers, the weight W_i of each rule is configurable. This allows the configuration of the contribution of each rule in the final decision. In the context analysis part, the current location of the car is divided into different zones. In each zone, there is a target point and target speed that the car should follow. It is not always possible to achieve both targets. Generally, there is a trade-off between getting the right speed and right steering. By changing the weights of the rules, the balance of the two targets can be controlled. The weight values of the corresponding rules can be adjusted according to our preference in each context (zone). If steering by reducing braking in the positioning zone is favoured, all the weights for any rule that controls brake can be divided by half for example. A set of rules on a specific context can also be activated/deactivated by assigning 1 or 0 to the corresponding weights.

6.5.1 Context-dependent membership functions

To control the speed of the racing car when taking a turn of radius r ($r = [0, \infty]$, where ∞ means a straight segment), linguistic variable called $speedX$ with three associated linguistic terms "SLOW", "MODERATE" and "FAST" is defined. The current distance from the car to the incoming turn is $d < 100$. The idea is to adjust the speed of the car around a "MODERATE" value when taking a turn. Naturally there is no specific range of speed that suits every turn. In fact, an appropriate turning speed must be dependent on the radius r and the distance d from the car to the turn. ($C = \{r, d\}$). The membership functions of "SLOW", "MODERATE" and "FAST" are defined as shown in Figure 6.18. The shapes and widths of the membership functions are fixed regardless of context. Only the position of the functions are changed. The parameters a, b, c, d and e completely define all three membership functions. Let $a - b = b - c = c - d = d - e = constant$, now only the parameter c is required to construct the membership functions of $Speed$ at each context. c is the appropriate speed the car should achieve in the current situation and c must be dependent on r and d. Note that when $d \to 0$, the car is getting closer to the turn. When $d = 0$, the value of c determines the fastest possible speed that the car can safely make the turn of radius r. c can be defined as

$$c = speedAtRadius(r) + \alpha \times d \tag{6.5.1.1}$$

From (6.5.1.1), c can be thought of as the highest speed given an available braking distance d so that the car can reduce its speed from c down to the target speed specified by $speedAtRadius(r)$. In practice, the coefficient α is dependent on many factors, e.g., average deceleration of the car, braking capacity, condition of the tyre, road surface, etc., therefore, it is generally impossible to work out the exact value of α. However, it is not difficult to give a rough estimation of α. If a car accelerates to a certain speed and then applies full brake so that the car slows down to a predefined speed, we can record the distance required for the car to slow down from the initial speed to the target speed. The ratio $\alpha' = \frac{dv}{d}$ is recorded. The process is repeated for different initial and target speed, and for different road surface. At the end, α is calculated as the average

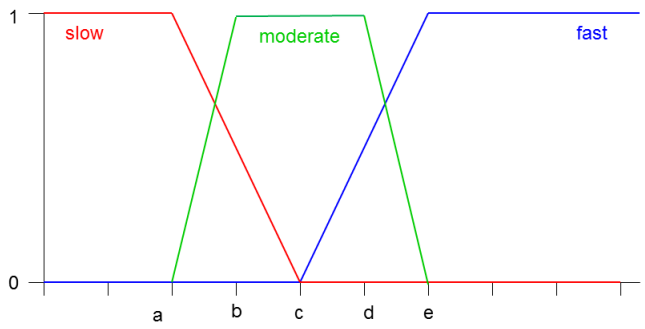


Figure 6.18: Representations of "SLOW", "MODERATE" and "FAST" speed

value of all α'.

$$\alpha = \frac{\sum_{i=1}^{n} \alpha_i'}{n} \tag{6.5.1.2}$$

The function $speedAtRadius(r)$ has been already defined in (6.4.1.6).

$$speedAtRadius(r) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + \frac{b_1}{x} + \frac{b_2}{x^2} + \frac{b_3}{x^3} \tag{6.5.1.3}$$

Finally, c can be calculated as in (6.5.1.1)

6.5.2 Driving behaviours

There are so many rules in the fuzzy systems for the speed and steering controllers that this chapter unable to go into all the details. However, brief descriptions of the driving strategies implemented by the fuzzy controllers will be touched upon. There are three behaviours of the car that can be implemented: drifting, balancing and standard driving behaviour.

Drifting

There are two occasions in which drifting is necessary.

1. The car is at an extremely high speed and is unable to reach the apex point of the turn.

2. The car is in the apex zone and travelling very fast. It may be desirable to keep the car at the apex point for as long as possible and reduce the speed to a reasonable level.

The car can drift from a sudden change in steering direction. After the car is off balance, it will quickly steer the car towards the corner. The car will start to drift through the corner and loose speed rapidly. When the speed is at a reasonable level, the car regains balance and drives normally. There is a set of rules that are specifically designed for handling drifting. When the car is drifting, only the relevant rules are activated, all other rules are deactivated.

Balancing

When the car is not drifting, the top priority of the speed and steering controllers are to maintain the car's balance and stability. The following situations all require special treatment:

- The car is flying;
- The car has landed after flying;
- The car suddenly looses balance; and
- The car is sliding too fast in one direction (either towards or away from the turn).

When one of these conditions occurs, the car will not follow the target waypoint and target speed, instead, it tries to regain balance first.

Standard driving behaviour

The car steers towards the target point and controls the speed according to the suggested target speed. In most situations, the CDFS gives smooth control of the car to

the target waypoint. However, there are some special situations in which a smooth control towards the target waypoint is not desired. Figure 6.14 shows an example of such a situation: the car is currently in a straight segment approaching a turn. The context analysis module could not determine the direction of the turn because there is only one point visible (the highest point) belonging to the coming turn (all other points belonging to the left and right edges). Now the path planner would suggest that the car carries straight on because the last segment is straight. If the car continues its direction ($steering = 0$ or very small), at the next step, it will only see the highest point getting closer and closer. The segment estimation algorithm cannot detect a new segment because the highest point does not belong to any edge and even if it does, from only one point, the algorithm cannot estimate a segment. The path planner therefore will continue to suggest that the car goes straight. The car will not see any other point on the coming turn segment until it gets much closer, which might be too late to make the turn. To handle such a situation, a small random perturbation is added to the final outputs of the controller such that the car still generally moves towards the target point. The random perturbation allows the exploration the area surrounding the target point and hence improves the visibility of the car.

6.6 Experiments

As mentioned previously, to best evaluate the usefulness of our modelling technique, the performance of our hybrid controller "Drifter" which was built upon the estimated track segments will be compared against other top drivers in TORCS and in SCRC. The car is car1-trb1 which was used in SCRC. The drivers in comparisons are Drifter, Simplix (one of the best controllers in original TORCS), berniw3, bt3, inferno3, lliaw3, tita3 olethros3 (car1-trb1 drivers in the original TORCS distribution). The tracks used are Street 1,Wheel 1, Wheel 2, Alpine 1, Alpine 2, E-track 2, E-track 3, E-track 4, E-track 5, E-track 6, E-Road, Forza, CG1, CG2, CG3, Michigan, F-Speed Way, E-track 5, E-speed way, C-speed way, B-speed way and A-speed way. Each bot will be driving solo on each track for 20 laps. To investigate the performance of our bots on an entirely

unknown environment, the results of each bot for the first-lap only will be separately presented in Table 6.4. Table 6.5 shows the performances of all bots in 20 lap races. All the tests were run on a three-year-old Windows XP SP1 laptop and in "Practice mode". The format of the recorded time is minute: second: centisecond. It is noticed that three controllers; inferno3, lliaw3 and tita3 in the TORCS distribution had exactly the same results for every track, so we grouped the results of all these three into one column and referred to the controller as inferno3. The name of the fastest driver for each track is also listed.

In the first-lap only races, the best driver is Simplix with the fastest lap times in nine out of 22 tracks. Our controller Drifter comes second with fastest lap times in seven tracks. However, in terms of total time, Drifter comes first at three seconds faster than Simplix. All the bots are very good and their performances are close to each other. In 20 lap races, Drifter comes top with fastest race times in fourteen tracks. Simplix comes second with fastest performances in four tracks. In terms of total time, Drifter is also top of the table with two minutes clear of runner up bot Simplix. Results of human player performance on some of the tracks can be found in [93]. The results represented here do not imply that Drifter is superior to those in TORCS, it just shows that a potentially very good controller based on the estimation techniques described above can be built.

6.7 Discussion & future work

In this chapter, we have discussed the methods for learning track models as well as the design of the context-dependent fuzzy controller that was built upon the learned track models. A comparison of our bot with the best drivers in the original TORCS has been made. Although in theory, a bot which only operates on a partially known environment cannot surpass those that have global knowledge, the fact that our bot has outperformed the best regular TORCS controllers on the majority of standard tracks is clear evidence that there is still room for further improvement for drivers in the original

Maps	berniw3	inferno3	olethros3	Simplix	Drifter	Best driver
Street 1	01:26:22	**01:25:47**	01:35:32	1:25:92	1:25:94	inferno3
Wheel 1	01:26:69	01:25:87	01:35:00	**1:24:07**	1:25:09	Simplix
Wheel 2	2:00:01	1:59:03	1:59:00	**1:56:09**	1:59:02	Simplix
Alpine 1	02:16:88	02:16:17	02:16:42	**2:13:96**	2:15:84	Simplix
Alpine 2	01:46:76	01:46:00	01:44:86	1:46:58	**1:37:20**	Drifter
E-track 2	01:22:55	01:21:70	01:27:90	1:21:98	**1:20:20**	Drifter
E-track 3	01:36:43	01:34:72	01:35:08	**1:32:10**	1:37:36	Simplix
E-track 4	01:53:79	01:52:87	01:55:03	**1:51:62**	1:54:72	Simplix
E-track 5	00:37:15	00:36:43	00:38:67	**00:33:31**	00:33:68	Simplix
E-track 6	01:33:85	01:33:13	01:32:29	**1:29:37**	1:31:38	Simplix
E-Road	01:14:03	01:13:17	01:14:26	1:10:86	**1:10:05**	Drifter
Forza	01:39:21	01:38:44	01:37:13	**1:34:79**	1:38:85	Simplix
CG1	00:46:82	00:45:62	00:45:48	00:46:85	**00:44:56**	Drifter
CG2	01:00:33	00:59:67	01:04:56	**00:58:15**	00:59:81	Simplix
CG3	01:10:30	01:09:59	01:08:31	1:09:46	**1:05:03**	Drifter
Michigan	00:43:73	00:43:07	00:44:52	00:42:73	**00:41:70**	Drifter
F-Speed Way	00:53:78	00:53:10	**00:52:80**	00:55:51	00:53:90	olethros3
E-track 5	00:37:15	00:36:43	00:38:67	00:33:31	**00:33:21**	Drifter
E-speed way	00:58:41	**00:57:72**	00:57:80	00:59:66	00:58:71	inferno3
C-speed way	00:48:68	**00:48:01**	00:48:15	00:50:22	00:49:07	inferno3
B-speed way	00:56:73	**00:56:46**	00:56:52	00:58:28	00:57:50	inferno3
A-speed way	00:36:32	**00:35:06**	00:37:14	00:35:35	00:35:52	inferno3
Total	27:16:48	27:00:31	27:37:29	26:42:40	**26:39:46**	

Table 6.4: First lap performances of all controllers

Maps	berniw3	inferno3	olethros3	Simplix	Drifter	Best driver
Street 1	26:59:00	26:58:00	27:15:00	26:53:00	**26:25:00**	Drifter
Wheel 1	27:08:00	27:06:00	29:22:00	26:26:00	**26:03:00**	Drifter
Wheel 2	38:35:00	38:34:00	37:44:00	**32:25:00**	38:15:00	Simplix
Alpine 1	22:15:00	22:12:00	22:11:00	**21:49:00**	22:05:00	Simplix
Alpine 2	33:50:00	33:45:00	33:20:00	33:56:00	**31:03:00**	Drifter
E-track 2	26:24:00	26:23:00	26:06:00	26:41:00	**25:13:00**	Drifter
E-track 3	30:01:00	29:45:00	29:35:00	**28:55:00**	29:02:00	Simplix
E-track 4	36:00:00	35:57:00	36:08:00	35:44:00	**35:32:00**	Drifter
E-track 5	10:37:00	10:36:00	10:33:00	9:30:00	**9:07:00**	Drifter
E-track 6	29:19:00	29:18:00	28:26:00	28:02:00	**27:54:00**	Drifter
E-Road	22:36:00	22:35:00	22:06:00	21:35:00	**21:10:00**	Drifter
Forza	30:39:00	30:38:00	30:06:00	**27:43:00**	29:50:00	Simplix
CG1	13:48:00	13:46:00	13:17:00	13:09:00	**12:58:00**	Drifter
CG2	18:05:00	18:04:00	18:04:00	17:29:00	**17:24:00**	Drifter
CG3	21:46:00	21:44:00	21:26:00	21:16:00	**19:44:00**	Drifter
Michigan	12:18:00	12:17:00	12:23:00	12:11:00	**11:24:00**	Drifter
F-Speed Way	14:58:00	**14:57:00**	14:58:00	14:59:00	14:59:00	inferno 3
E-track 5	10:37:00	10:36:00	10:33:00	9:30:00	**9:07:00**	Drifter
E-speed way	16:30:00	**16:29:00**	16:35:00	16:35:00	16:34:00	inferno 3
C-speed way	13:17:00	**13:16:00**	13:22:00	13:51:00	13:23:00	inferno 3
B-speed way	16:06:00	**16:05:00**	16:11:00	16:34:00	16:11:00	inferno 3
A-speed way	9:46:00	9:44:00	9:57:00	9:40:00	**9:32:00**	Drifter
Total	481:34:00	480:45:00	479:38:00	464:53:00	**462:55:00**	

Table 6.5: Performances of all controllers in 20 lap races

TORCS as well as SCRC. Another point to note is that our bot was developed based on the rules of the first SCRC (2008) which neglects clutch control. Clutch control allows an efficient gear shifting mechanism which helps the car accelerate more quickly especially from a standstill position. There are several limitations in our implementation bot which could be further improved.

- Poor performance on dirt track: the bot can perform very well on many different types of road tracks but not on dirt track. Due to the low grip surface of dirt tracks, the car needs to travel at significant slower speeds than usual and therefore requires that the fuzzy sets of the linguistic variable Speed to be moved to other suitable locations. This feature could be added in the future to specifically deal with low grip tracks.

- No second lap improvement: the bot always reacts to what it sees and does not maintain a memory of the features of the track it has passed. As a result, the same behaviours and mistakes are repeated over and over again. Second lap improvement is another feature that should be added in the future.

- No car overtaking and car blocking strategies have been included at the moment.

When all of the above features are implemented, we believe that SCRC bots can compete fairly with TORCS bots and potentially outperform them in any of the tracks.

CHAPTER 7

Conclusions and Future Work

In this chapter, we summarize the main contributions of this thesis both in terms of theory and applications. A final section will discuss limitations and suggest directions for future research.

7.1 Contribution of the thesis

7.1.1 Theoretical contribution

In this research, we have developed a novel approach to solve the issue of generality of linguistic terms by incorporating contextual information into the representation of fuzzy sets. This approach ensures the improvement accuracy while still preserving the interpretability of FRBSs. It is well known that traditional fuzzy techniques, despite having many advantages due to their simplicity, are not sufficiently flexible to model the influence of context on the meaning of linguistic terms in an accurate and concise manner. Research in the past mainly focused on modelling changes made to the data base while largely ignoring the definitions of context models. There are two main approaches to model the effects of context on the meanings of fuzzy terms depending on whether those fuzzy terms were defined and maintained by experts or computers. The first is by manual expert adjustments of the default fuzzy sets to the context. The second is to represent context as some form of filtering functions applied on the normalised membership functions. The process of finding suitable filtering functions is

called the context adaptation process. Each of the approaches is only suitable in the field of applications for which it is designed. What distinguishes this research from earlier fuzzy context modelling work is that our approach, based on a well-defined semantics and typing context model, is amenable to both expert design and automatic generation from learning methods. Contexts are constructed from a series of components which could be any kind of variable, e.g., numerical or linguistic variables. A context change is signalled by a change in value of any of the context components.

The research also looked into other approaches which is not designed for context modelling but possess the capability of being a good one, i.e., HFS. Despite sharing many similarities, it has been shown that our approach is different. The comparison between the two has been made both theoretically (in chapter 3) and practically (in chapter 5). In the well-known Mackey-Glass time-series benchmark, CDFS outperform HFS by a fair margin. However, we cannot conclude that whether our CDFS approach is better than HFS both in terms of transparency as well as interpretation. To sum up, the main features and benefits of context-dependent fuzzy sets in fuzzy expert and fuzzy modelling applications are as follows:

- In fuzzy expert systems, experts can hard code membership functions in the same way as a standard fuzzy approach, or provide rules or mathematical functions to automatically create MFs from the context components. In other words, the context-dependent fuzzy approach allows experts to automate the MFs creation process. This simultaneously improves the modelling accuracy and significantly reduces the number of rules required, and thus increases the interpretability of the system. This flexibility allows complete capture of the dependence in meanings between input variables. Our approach provides an alternative method other than HFS to solve the "curse of dimenionality" problems when modelling the inter-relationships among input variables.

- In fuzzy modelling, our proposed context-dependent approach can be used in collaboration with any existing learning method to help achieve enhanced performance. The key difference between our context-dependent fuzzy modelling method and other fuzzy identification approaches is that the final optimised FRBSs

of other approaches are always fixed, while our proposed tuned context-dependent FRBSs have different fuzzy sets for different inputs. Depending on the extracted features of inputs, the MFs of the optimised FRBS are tuned accordingly. Our approach allows a true "context-dependent" fuzzy system, i.e. a changing fuzzy system, to be tuned. The details of the proposed modelling method were given in chapter 5 of this thesis.

7.1.2 Applications

The effectiveness and role of the proposed context dependent fuzzy model is investigated and compared to other approaches through a series of applications in several research areas. In particular, the proposed fuzzy model and its capabilities can be applied in the applications as described below.

1. To investigate the expressive capability of our approach as a stand-alone method compared to conventional fuzzy approaches, the proposed fuzzy model is applied to develop car racing bots for the FUZZ-IEEE 2007 [57] and the 2007 IEEE CEC Simulated Car Racing Competition [128] in chapter 4.

    - The FUZZ-IEEE 2007 car racing competition [57]: a simple point-to-point car racing game in which multiple car agents compete with each other for waypoints in a racing field. The racing problem is fairly simple and ignores skidding or collision. To make a fair comparison and to ensure that the difference in performance between the controllers are not due to exogenous factors such as rule base, inference process and so on, a standard type-1 FRBS is developed with identical structure and rule base. The only difference lies in the representation of one single MF that is defined using context dependent fuzzy set in the context-dependent FRBS.
    - The 2007 IEEE CEC Simulated Car Racing Competition [128]: a similar but more sophisticated point-to-point car racing competition which was held as part of the CIG 2007 and CEC 2007 conferences. With the introduction of skidding and car-to-car collisions as well as much more sophisticated un-

derlying physics, the car controller needs not only to get as quickly as possible to the next waypoint, but also to anticipate other opponents' actions and plan a good response strategy. In this application, our controller competed against other strong competitors from both the fuzzy and computational intelligence communities.

Our hand-tuned context-dependent fuzzy controllers won first prize in both competitions in solo and dual races, in both noisy and noiseless tracks. Moreover, in the FuzzIEEE competition, our proposed fuzzy controller also outperformed the nearly identical standard type-1 fuzzy controller despite our best effort to tune its MFs. This has proven the modelling capability of context-dependent fuzzy sets. The two applications have also illustrated how experts can automate the creation of MFs using the context components to represent an unlimited number of contexts.

2. To study the role and benefits of the context-dependent fuzzy model in fuzzy modelling tasks, the proposed method is applied for the well-known Mackey-Glass time-series prediction benchmark in chapter 5. In particular, a combination of methods, i.e., an IMPGA, a space search MA and an improved QR householder LSM, is used to create a standard type-1 TSK fuzzy system for the problem as the first step. Then, in the next step, input variables are used as context components and K-means algorithm is utilised to form the context by clustering the input space into several context groups. In each context group, the MFs of the type-1 FRBS will be scaled by different values. A context adaptation algorithm is introduced to optimise the transformation matrix which contains scale values in each context. The resulting fuzzy system is unique in the sense that the MFs are not fixed during the inference process but vary depending on the inputs. The benefit of the proposed context adaptation is that it collaborates with the standard fuzzy learning methods to further improve the performance of the well-optimised FRBS. To the best of our knowledge, the techniques proposed to tune the type-1 FRBS have already achieved the best performance for the Mackey-Glass benchmark to date. With the added context adaptation, enhanced perfor-

mance has been record with 50% improvement in accuracy.

3. To explore the advantages when combining the proposed fuzzy modelling with other techniques, a multi-agent car racing bot which utilises context-dependent fuzzy sets has been developed for the TORCS-based Simulated Car Racing Championship (SCRC). The SCRC is a complicated, close to real-world, car racing competition based on The Open Racing Car Simulator (TORCS). The controller is designed to race in solo on unknown tracks with very limited information about the surrounding environment. There are two important steps involved in controlling an SCRC bot. The first is track modelling which reconstructs the track segments as close to the original as possible from the current sensory data and previous observations of the track. The second is to use the track models to navigate around the track. The core component is a context-dependent fuzzy system which contains all the general rules that define the driving behaviours of the bot. These rules are fixed regardless of the situation. However, the actual representations of the linguistic terms are defined according to shapes of the visible parts of the track so that the same rules can be used to take any kind of corners. Our approach is very similar to that which a human driver has to process in the real-world, i.e., observe the track, plan a strategy and adapt the driving behaviour to the situations. The methods to learn both the track models and the context-dependent fuzzy system are novel and efficient enough to be applied real-time in the races. The performance of the our controller is very competitive against the best drivers in regular TORCS, which have access to global, accurate information of the track environment prior to and during the race.

In every application, our fuzzy approach, whether as a stand-alone or in combination with other methods, has shown superior performance compared to any other technique for the same applications to the best of our knowledge. The use of context and/or context adaptation had not been previous carried out by any of the applications; and the fact that our context-dependent fuzzy approach produced the best performance in all four applications has proven its capability as a modelling method.

7.2 Limitations & future work

Although the research has achieved its initially intended aims, there remain some limitations which could be addressed in future work. First, because of time limitations, only a small number of applications of the proposed approach have been carried out. While the results attained were significant, the fact that three out of four applications involved car racing games makes it hard to draw a strong conclusion of the applicability of the approach in other research areas. Second, one of the objectives of the proposed context fuzzy approach is to model the inter-relationships between input variables, which could be linguistic variables. Although the use of linguistic variables as context components with the help of fuzzy logic to generate the context-dependent fuzzy sets have been briefly described in chapter 3, a practical example of the methodology is lacking. Finally, although the applicability of context-dependent fuzzy sets in complicated, close to real-world problems has been investigated in chapter 6, the fact that a number of other different techniques were also involved in the development of the system makes it very difficult to quantitatively determine the contribution of the approach in the overall solution. Therefore, despite the impressive results, no decisive conclusion regarding the effectiveness of our proposed fuzzy approach can be drawn. However, when one of the best performing solutions has been achieved for a problem generally considered not suitable for fuzzy-based approaches, we can assert with confidence the improved accuracy in modelling attributable to our fuzzy approach must play an important role.

The limitations can lay the groundwork for future academic research. The cooperation of context models and other types of fuzzy systems, e.g., type-2 fuzzy systems or non-stationary fuzzy systems, could be an interesting research topic. The use of linguistic variables in context components is also an interesting area that should be further explored. Last but not least, in order to validate the usefulness of the approach under more general sets of circumstances, it has to be supported by a fair amount of experiments in a variety of research areas. This of course is reserved for future research.

APPENDICES

APPENDIX A

Standard QR Householder transformation

For reference, we provide details of the specific QR Householder transformation used in this paper (adapted from Bates, 2008 [7]). A Householder transformation is a transformation that takes a vector and reflects it about some plane or hyperplane. We can use this operation to calculate the QR factorization of an $m \times n$ matrix A with $m \geq n$. Q can be used to reflect a vector in such a way that all coordinates but one disappear. Let $\mathbf{x}$ be an arbitrary real m-dimensional column vector of A such that $||\mathbf{x}|| = |\alpha|$ for a scalar α , e_1 is the vector $(1, 0, ..., 0)^T$,$\|.\|$ is the Euclidean norm and I is an $m \times m$ identity matrix. We can set $u = \mathbf{x} + \alpha e_1; v = \frac{u}{\|u\|}; Q = I - 2vv^T$. Q is an $m \times m$ Householder matrix and $Q\mathbf{x} = (\alpha, 0, ..., 0)^T$. Now, the $m \times n$ matrix A is transformed to upper triangular form. Let Q_1 be the Householder matrix obtained when the first matrix column $\mathbf{x}$ is chosen. If we multiply A with Q_1, the resulting matrix $Q_1 A$ is filled with zeros in the left column (except for the first row).

$$Q_1 A = \begin{bmatrix} \alpha_1 & \star & \dots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{bmatrix} \tag{A.0.0.1}$$

Repeat the process for the matrix A', which is obtained from $Q_1 A$ by deleting the first row and first column. This results in a Householder matrix Q'_2 . Note that Q'_2 is

smaller than Q_1. Since we want to operate on $Q_1 A$ instead of A' we need to expand Q_2' to the upper left, filling in a 1, or in general $Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & Q_k' \end{pmatrix}$. After t iterations of this process, $t = \min(m-1, n)$, we end up with $R = Q_t \cdots Q_2 Q_1 A$, which is an upper triangular matrix and $Q = Q_1^T Q_2^T \cdots Q_t^T$. With the matrix Q and R, we can easily work out the solution of the least square problem by using equation (4.2.4.5). See [7] for a more comprehensive explanation of the method.

APPENDIX B

Best fit curve through two points

In this section, we present the method that were used in chapter 6 for estimating a curve (line or circle) based on the knowledge of adjacent segment and two data points. Let's p be the segment from which we use to estimate the new segment through two points A and B.

1. If p is a straight segment, we can find the circle passes through point A and B at touches the line specified by p

   - Find the intersection O of the line created by points A,B and the line created by *start* and *end*.
   - $r = \sqrt{OA \times OB}$
   - Find C on the line created by the point O and the point *end* so that $OC = r$.
   - Find the circle through A, B, C (see figure)

2. If p is a turn segment, let's (ox, oy) and r_0 be the centre and radius of p. Let's (Ax, Ay) and (Bx, By) be the location of points A and B respectively. The circle (x, y, r) passes through A and B and touches the circle represented by (ox, oy, r_0) can be found by solving the follow three equations

$$\begin{cases} (x - Ax)^2 + (y - Ay)^2 = r^2 \\ (x - Bx)^2 + (y - By)^2 = r^2 \\ (x - ox)^2 + (y - oy)^2 = (r \pm r_0)^2 \end{cases} \tag{B.0.0.1}$$

where the $\pm$ operator at the third equation indicates that the circle we are finding touches the circle (ox, oy, r_0) externally or internally. Although the solution to B.0.0.1 can be found without much difficulty using elementary mathematics, the length of the solutions as well as the calculations are too long and complicated. For brevity's sake, we will omit both in this work.

There could be 0 or multiple solution of the problem described above. If there is at least a solution, we need to verify the correctness of it by the followings steps:

- If the radius r is too large (>1000), it is likely that the segment is in fact a straight segment. We verify that if the line through A and B touches the circle specified by p. If it does than the new segment is a straight segment through A and B
- If the radius r is too small, we reject the solution
- We find the touching point P of the segment p and the new found circle. P is in fact the estimated starting point (or ending point if p is at higher location than both A and B) of the new segment, therefore, it must be the lowest (or highest point) in p. If P is supposed to be the lowest point and $P.y > max(Ay, By)$ or $P.y < min(start.y, end.y)$ we reject the solution. Similarly, if P is supposed to be the highest point, we reject the solution if $P.y < min(Ay, By)$ or $P.y > max(start.y, end.y)$.

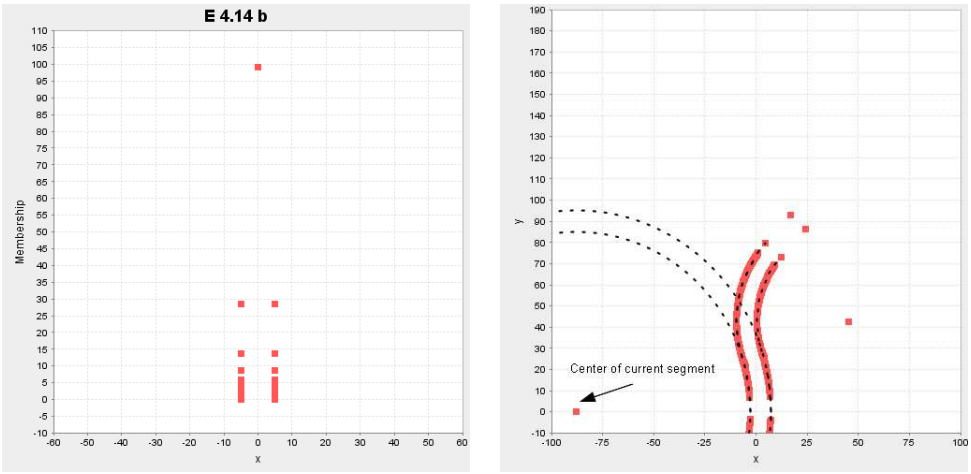
APPENDIX C

Finding the first segment

In this section, we present the method that was used in chapter 6 to identify the current segment. The current segment or the first segment is a special segment that has a very specific signature that could easily be identified. If the segment is straight, its start and end point must be in the same straight up line. In other words, $abs(end.x - start.x)$ must be very very small. If the segment is a turn, its centre must lie on the track horizontal axis. In other words, $center.y$ must be very close to 0. (see Figure C.1). If the first segment has radius r, then the distance from the centre of the turn to the middle of the current track segment is also r. The middle of the current track segment always has the location of $(toMiddle, 0)$ which is independent of r. Therefore, if we know the radius r and the direction of the turn, then we can estimate:

- centre of the turn : $(toMiddle + type * r, 0)$
- radius of the side segment: $r - type * which * trackWidth * 0.5$

where $type \in \{-1, 0, 1\}$ and $which = -1$ for left edge, 1 for right edge.



Straight Turn

Figure C.1: First segment

Bibliography

[1] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 2006 International Conference on Machine Learning (ICML '06)*, pages 1–8, Pittsburgh, Pennsylvania, USA, June 25-29, 2006.

[2] A. Agapitos, M. O'Neill, A. Brabazon, and T. Theodoridis. Learning environment models in car racing using stateful genetic programming. In *Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games (CIG '11)*, pages 219–226, Seoul, South Korea, August 31 - Semtember 3, 2011.

[3] A. Agapitos, J. Togelius, and S. M. Lucas. Evolving controllers for simulated car racing using object oriented genetic programming. In *Proceedings of the 2007 Genetic and Evolutionary Computation Conference (GECCO '07)*, pages 1543–1550, London, England, July 7-11, 2007.

[4] M. Almaraashi, R. John, S. Coupland, and A. Hopgood. Time series forecasting using a TSK fuzzy system tuned with simulated annealing. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '10)*, pages 1–6, Barcelona, Spain, July 18-23, 2010.

[5] M. Awad, H. Pomares, I. R. Ruiz, O. Salameh, and M. Hamdon. Prediction of time series using RBF neural networks: A new approach of clustering. *International Arab Journal of Information Technology*, 6(2):138–143, 2009.

[6] A. Bastian. How to handle the flexibility of linguistic variables with applications. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(4): 463–484, 1994.

[7] D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*, chapter Appendix 2. QR Decompositions Using Householder Transformations, pages 286–289. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2008.

[8] W.-D. Beelitz. The SIMPLy mIXed best practice TORCS robot, 2009. URL http://www.wdbee.gotdns.org:8086/SIMPLIX/SimplixDefault.aspx.

[9] J. C. Bezdek, M. R. Pal, J. Keller, and R. Krisnapuram. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, volume 4 of *The Handbooks of Fuzzy Sets*. Kluwer Academic Publishers, Boston, USA, 1999.

[10] T. Bohlin. *Interactive system identification: prospects and pitfalls*. Communications and control engineering series. Springer Berlin Heidelberg, New York, USA, 1991.

[11] P. Bonissone, P. S. Khedkar, and Y. Chen. Genetic algorithms for automated tuning of fuzzy controllers: A transportation application. In *Proceedings of the 1996 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '96)*, volume 1, pages 674–680, New Orleans, LA, USA, September 8-11, 1996.

[12] A. Botta, B. Lazzerini, F. Marcelloni, and D. Stefanescu. Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13(5):437–449, 2009.

[13] B. Bouchon-Meunier and Y. Jia. Linguistic modifiers and imprecise categories. *International Journal of Intelligent Systems*, 7(1):25–36, 1992.

[14] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2(1):321–355, 1988.

[15] R. L. Cannon, J. V. Dave, and J. C. Bezdek. Efficient implementation of the fuzzy c-means clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):248–255, 1986.

[16] L. Cardamone, D. Loiacono, and P. L. Lanzi. Learning to drive in the Open Racing Car Simulator using online neuroevolution. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(3):176 –190, 2010.

[17] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli. Searching for the optimal racing line using genetic algorithms. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games (CIG '10)*, pages 388–394, Copenhagen, Denmark, August 18-21, 2010.

[18] L. Cardamone, D. Loiacono, and P. L. Lanzi. Applying cooperative coevolution to compete in the 2009 TORCS Endurance World Championship. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC '10)*, pages 1–8, Barcelona, Spain, July 18-23, 2010.

[19] L. Cardamone, D. Loiacono, and P. L. Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO '09)*, pages 1179–1186, Montreal, Québec, Canada, July 8-12, 2009.

[20] J. Casillas, O. Cordón, and F. Herrera. COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32(4):526–537, 2002.

[21] J. Casillas, O. Cordón, F. Herrera, and J. J. M. Guervós. Cooperative coevolution for learning fuzzy rule-based systems. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, volume 2310 of *Lecture Notes in Computer Science*, pages 311–322. Springer Berlin Heidelberg, 2002.

[22] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena. Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: An overview. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 3–22. Springer Berlin Heidelberg, 2003.

[23] J. Casillas, F. Herrera, O. Cordón, and L. Magdalena. *Accuracy Improvements in Linguistic Fuzzy Modeling*, volume 129 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, New York, USA, 2003.

[24] J. Casillas, O. Cordón, and F. Herrera. Learning fuzzy rules using ant colony optimization algorithms. pages 13–21, Brussels, Belgium, September 8-9, 2000.

[25] B. Chaperot and C. Fyfe. Improving artificial intelligence in a motocross game. In *Proceedings of the 2006 IEEE Conference on Computational Intelligence and Games (CIG '06)*, pages 181–186, Reno, USA, May 22-26, 2006.

[26] Y. Chen and A. Abraham. Hierarchical fuzzy systems. In *Tree-Structure based Hybrid Computational Intelligence*, volume 2 of *Intelligent Systems Reference Library*, pages 129–147. Springer Berlin Heidelberg, 2010.

[27] Y. Chen, B. Yang, A. Abraham, and L. Peng. Automatic design of hierarchical Takagi-Sugeno type fuzzy systems using evolutionary algorithms. *IEEE Transactions on Fuzzy Systems*, 15(3):385–397, 2007.

[28] Y. Chen, L. Peng, and A. Abraham. Programming hierarchical TS fuzzy systems. In *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems (EFS '06)*, pages 157–162, Ambleside, UK, September 7-9, 2006.

[29] J.-S. Cho and D.-J. Park. Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network. *Journal of Intelligent and Fuzzy Systems*, 8(2):99–110, 2000.

[30] J.-N. Choi, Y.-I. Lee, and S.-K. Oh. Fuzzy radial basis function neural networks with information granulation and its genetic optimization. In W. Yu, H. He, and N. Zhang, editors, *Advances in Neural Networks - ISNN 2009*, volume 5552 of *Lecture Notes in Computer Science*, pages 127–134. Springer Berlin Heidelberg, 2009.

[31] J.-N. Choi, S.-K. Oh, and H.-K. Kim. Hybrid optimization of information granulation-based fuzzy radial basis function neural networks. *International Journal of Intelligent Computing and Cybernetics*, 3(4):593–610, 2010.

[32] O. Cordón and F. Herrera. A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples. *International Journal of Approximate Reasoning*, 17(4):369–407, 1997.

[33] O. Cordón and F. Herrera. A two-stage evolutionary process for designing TSK fuzzy rule-based systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 29 (6):703–715, 1999.

[34] O. Cordón, F. Herrera, L. Magdalena, and P. Villar. A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base. *Information Sciences*, 136(1-4):85–107, 2001.

[35] O. Cordón, F. Herrera, and I. Zwir. Linguistic modeling by hierarchical systems of linguistic rules. *IEEE Transactions on Fuzzy Systems*, 10(1):2–20, 2002.

[36] R. Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2002.

[37] C. G. Coy and D. Kaur. Improving evolutionary training for Sugeno fuzzy inference systems using a mutable rule base. In *Proceedings of the 2010 North American Fuzzy Information Processing Society Annual Conference(NAFIPS '10)*, pages 139–144, Toronto, Canada, July 12-14, 2010.

[38] F. Dellaert, D. Pomerleau, and C. Thorpe. Model-based car tracking integrated with a road-follower. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA '98)*, volume 3, pages 1889–1894, Leuven, Belgium, May 16-20, 1998.

[39] A. Devillez, P. Billaudel, and G. V. Lecolier. A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition. *Fuzzy Sets and Systems*, 128(3):323–338, 2002.

[40] Y. El-Sonbaty and M. A. Ismail. Fuzzy clustering for symbolic data. *IEEE Transactions on Fuzzy Systems*, 6(2):195–204, 1998.

[41] E. Espié, C. Guionneau, and B. Wymann. The Open Racing Car Simulator, 1997. URL http://torcs.sourceforge.net/.

[42] X. Fan, N. Zhang, and S. Teng. Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme. *Fuzzy Sets and Systems*, 144(2): 297–312, 2004.

[43] D. Floreano, T. Kato, D. Marocco, and E. Sauser. Coevolution of active vision and feature selection. *Biological Cybernetics*, 90(3):218–228, 2004.

[44] T. Fraichard and P. Garnier. Fuzzy control to drive car-like vehicles. *Robotics and Autonomous Systems*, 34(1):1–22, 2001.

[45] J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation, Parts I and II. *The Computer Journal*, 4(3):265–271, 1961.

[46] M. Gao and S. He. Self-adapting fuzzy-PID control of variable universe in the non-linear system. In *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA '08)*, volume 1, pages 473–478, Hunan, China, October 20-22, 2008.

[47] J. M. Garibaldi and T. Ozen. Nondeterministic fuzzy reasoning. *IEEE Transactions on Fuzzy Systems*, 15(1):1–15, 2007.

[48] J. M. Garibaldi, M. Jaroszewski, and S. Musikasuwan. Non-stationary fuzzy sets. *IEEE Transactions on Fuzzy Systems*, 16(4):1072–1086, 2006.

[49] J. Gebhardt and R. Kruse. The context model: An integrating view of vagueness and uncertainty. *International Journal of Approximate Reasoning*, 9(3):283–314, 1993.

[50] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, USA, 3rd edition, 1996.

[51] A. F. Gómez-Skarmeta and F. Jiménez. Fuzzy modeling with hybrid systems. *Fuzzy Sets and Systems*, 104(2):199–208, 1999.

[52] S. Guadarrama. Computing with actions: the case of driving a car in a simulated car race. In *Proceedings of the Joint 2009 International Fuzzy Systems Association*

World Congress and 2009 European Society of Fuzzy Logic and Technology Conference (IFSA/EUSFLAT'09), pages 1410–1415, Lisbon, Portugal, July 20-24, 2009.

[53] R. Gudwin and F. Gomide. Context adaptation in fuzzy processing. In *Proceedings of the First Brazil-Japan Joint Symposium in Fuzzy Systems*, pages 15–20, Campinas, SP, Brazil, 1994.

[54] R. Gudwin, F. Gomide, and W. Pedrycz. Context adaptation in fuzzy processing and genetic algorithms. *International Journal of Intelligent Systems*, 13(10-11):929–948, 1998.

[55] M. Guezouri and L. Mesbahi. A new approach using temporal radial basis function in chronological series. *The International Arab Journal of Information Technology*, 6(1):85–90, 2009.

[56] F. Herrera, M. Lozano, and J. L. Verdegay. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12(3-4):299–315, 1995.

[57] D. T. Ho and J. M. Garibaldi. A novel fuzzy inferencing methodology for simulated car racing. In *Proceedings of the 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '08)*, pages 1907–1914, Hong Kong, China, June 1-6, 2008.

[58] A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2):129–139, 1995.

[59] W. Huang, L. Ding, and S.-K. Oh. A comparative study of space search algorithm and particle swarm optimization in the design of ANFIS-based fuzzy models. *Majlesi Journal of Electrical Engineering*, 5(1):50–59, 2011.

[60] W. Huang, S.-K. Oh, and J.-T. Kim. Design of information granulation-based fuzzy models with the aid of multi-objective optimization and successive tuning method. In D. Liu, H. Zhang, M. Polycarpou, C. Alippi, and H. He, editors, *Advances in Neural Networks*, volume 6677 of *Lecture Notes in Computer Science*, pages 256–263. Springer Berlin Heidelberg, 2011.

[61] V. N. Huynh and Y. Nakamori. A context-based interpretation of linguistic variables. In *Proceedings of the 2004 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '04)*, volume 1, pages 257–262, Budapest, Hungary, July 25-29, 2004.

[62] V. N. Huynh, Y. Nakamori, T. B. Ho, and G. Resconi. A context model for constructing membership functions of fuzzy concepts based on modal logic. In T. Eiter and K.-D. Schewe, editors, *Foundations of Information and Knowledge Systems*, volume 2284 of *Lecture Notes in Computer Science*, pages 93–104. Springer Berlin Heidelberg, 2002.

[63] V. N. Huynh, M. Ryoke, Y. Nakamori, and T. B. Ho. Fuzziness and uncertainty within the framework of context model. In T. Bilgiç, B. Baets, and O. Kaynak, editors, *Fuzzy Sets and Systems - IFSA 2003*, volume 2715 of *Lecture Notes in Computer Science*, pages 219–228. Springer Berlin Heidelberg, 2003.

[64] V. N. Huynh, Y. Nakamori, T. B. Ho, and G. Resconi. A context model for fuzzy concept analysis based upon modal logic. *Information Sciences-Informatics and Computer Science*, 160(1-4):111–129, 2004.

[65] H. Ishibuchi and T. Nakashima. Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 9(4):506–515, 2001.

[66] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka. Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(3):260–270, 1995.

[67] R. Jager. *Fuzzy Logic in Control*. PhD thesis, Technical University of Delft, Delft, Holland, 1995.

[68] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, 1993.

[69] J.-S. R. Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4 (1):156–159, 1993.

[70] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *The Proceedings of the IEEE*, 83(3):378–406, 1995.

[71] P. E. Johnson. *A history of set theory*, volume 16 of *Prindle, Weber & Schmidt complementary series in mathematics*. Prindle, Weber & Schmidt, Boston, Massachusetts, USA, 1972.

[72] S. B. Jørgensen and K. M. Hangos. Grey box modelling for control: Qualitative models as a unifying framework. *International Journal of Adaptive Control and Signal Processing*, 9(6):547–562, 1995.

[73] M.-S. Kim, C.-H. Kim, and J. jang Lee. Evolving compact and interpretable Takagi-Sugeno fuzzy models with a new encoding scheme. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(5):1006–1023, 2006.

[74] F. Klawonn. Reducing the number of parameters of a fuzzy system using scaling functions. *Soft Computing*, 10(9):749–756, 2006.

[75] S. Lin, B. D. Schutter, Y. Xi, and H. Hellendoorn. An efficient model-based method for coordinated control of urban traffic networks. In *Proceedings of the 2010 IEEE International Conference on Networking, Sensing and Control (ICNSC '10)*, pages 8–13, Chicago, USA, April 11-13, 2010.

[76] Y. Lin. A new approach to fuzzy-neural system modeling. *IEEE Transactions on Fuzzy Systems*, 3(2):190–198, 1995.

[77] P. Lindskog. Fuzzy identification from a grey box modeling point of view. In H. Hellendoorn and D. Driankov, editors, *Fuzzy Model Identification*, pages 3–50. Springer Berlin Heidelberg, 1997.

[78] D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lönneker, L. Cardamone, D. Perez, Y. Saéz, M. Preuss, and J. Quadflieg. The 2009 Simulated Car Racing Championship. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):131–147, 2010.

[79] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez. The WCCI 2008 simulated car

racing competition. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG '08)*, pages 119–126, Perth, Australia, December 15-18, 2008.

[80] Z. Long, X. Liang, and L. Yang. Some approximation properties of adaptive fuzzy systems with variable universe of discourse. *Information Sciences*, 180(16):2991–3005, 2010.

[81] S. M. Lucas. *FUZZ-IEEE 2007 Simulated Car Racing Competition*, 2007. URL http://cswww.essex.ac.uk/staff/lucas/fuzzieee/FuzzyRace.html.

[82] S. M. Lucas and J. Togelius. Point-to-point car racing: an initial study of evolution versus temporal difference learning. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG '07)*, pages 260–267, Honolulu, Hawaii, April 1-5, 2007.

[83] M. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[84] J. B. Macqueen. Some methods for classification and analysis of multivariate observations. In *Procedings of the 1967 Berkeley Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, Berkeley, USA, 1967.

[85] L. Magdalena. Adapting the gain of an FLC with genetic algorithms. *International Journal of Approximate Reasoning*, 17(4):327–349, 1994.

[86] L. Magdalena. On the role of context in hierarchical fuzzy controllers. *International Journal of Intelligent Systems*, 17(5):471–493, 2002.

[87] L. Magdalena. Adapting gain and sensibility of FLCs with genetic algorithms. In *Proceedings of the 1996 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '96)*, volume 2, pages 739–744, Granada, Spain, July 1-5, 1996.

[88] L. P. Maguire, B. Roche, T. M. McGinnity, and L. J. McDaid. Predicting a chaotic time series using a fuzzy neural network. *Information Sciences*, 112(1-4):125–136, 1998.

[89] J. M. Mendel and R. I. John. Type-2 fuzzy sets made simple. *IEEE Transactions on Fuzzy Systems*, 10(2):117–127, 2002.

[90] S. Mitra and J. Basak. FRBF: A fuzzy radial basis function network. *Neural Computing & Applications*, 10(3):244–252, 2001.

[91] E. H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:394–395, 1920.

[92] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[93] J. Muñoz, G. Gutierrez, and A. Sanchis. A human-like TORCS controller for the Simulated Car Racing Championship. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games (CIG '10)*, pages 473–480, Copenhagen, Denmark, August 18-21, 2010.

[94] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Proceedings of the 1999 International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES '99)*, pages 88–92, Adelaide, Australia, August 31 - September 1, 1999.

[95] S.-K. Oh, W. Pedrycz, and K.-J. Park. Identification of fuzzy systems by means of genetic optimization and data granulation. *Journal of Intelligent and Fuzzy Systems*, 18(1):31–41, 2007.

[96] S.-Y. Oh and D.-J. Park. Self-tuning fuzzy controller with variable universe of discourse. In *Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics (SMC '95)*, volume 3, pages 2628–2632, British Columbia, Canada, October 22-25, 1995.

[97] E. Onieva, D. A. Pelta, J. Alonso, V. Milanés, and J. Pérez. A modular parametric architecture for the TORCS racing engine. In *Proceedings of the 2009 IEEE Conference on Computational Intelligence and Games (CIG '09)*, pages 256–262, Milano, Italy, September 7-10, 2009.

[98] T. Ozen, J. M. Garibaldi, and S. Musikasuwan. Preliminary investigations into modelling the variation in human decision making. In *Proceedings of the 2004 International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU '04)*, pages 641–648, Perugia, Italy, July 4-7, 2004.

[99] T. Ozen, J. M. Garibaldi, and S. Musikasuwan. Modelling the variation in human decision making. In *Proceedings of the 2004 North American Fuzzy Information Processing Society Annual Conference (NAFIPS '04)*, volume 2, pages 617–622, Banff, Canada, June 27-30, 2004.

[100] N. R. Pal and K. Pal. Handling of inconsistent rules with an extended model of fuzzy reasoning. *Journal of Intelligent and Fuzzy Systems*, 7(1):55–73, 1999.

[101] W. Pedrycz. *Fuzzy control and fuzzy systems*. Research Studies Press, New York, USA, 1st edition, 1989.

[102] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*. Research Studies Press, LTD., John Wiley & Sons, Inc, New York, USA, 2nd extended edition, 1993.

[103] W. Pedrycz. Fuzzy equalization in the construction of fuzzy sets. *Fuzzy Sets and Systems*, 119(2):329–335, 2001.

[104] W. Pedrycz and L. A. Zadeh. *Fuzzy Sets Engineering*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1995.

[105] W. Pedrycz, R. Gudwin, and F. Gomide. Nonlinear context adaptation in the calibration of fuzzy sets. *Fuzzy Sets and Systems*, 88(1):91–97, 1997.

[106] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(03):406–413, 1955.

[107] D. Perez, G. Recio, Y. Saez, and P. Isasi. Evolving a fuzzy controller for a car racing competition. In *Proceedings of the 2009 IEEE Conference on Computational Intelligence and Games (CIG '09)*, pages 263–270, Milano, Italy, September 7-10, 2009.

[108] U. Priber and W. Kretzschmar. Inspection and supervision by means of hierarchical fuzzy classifiers. *Fuzzy Sets and Systems*, 85(2):263–274, 1997.

[109] A. Priyono and M. Ridwan. Generation of fuzzy rules with subtractive clustering. *Jurnal Teknologi*, 43(D):143–153, 2005.

[110] J. Quadflieg, M. Preuss, O. Kramer, and G. Rudolph. Learning the track and planning ahead in a car racing controller. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games (CIG '10)*, pages 395–402, Copenhagen, Denmark, August 18-21, 2010.

[111] G. V. S. Raju and J. Zhou. Adaptive hierarchical fuzzy controller. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):973–980, 1993.

[112] G. Resconi, G. J. Klir, and U. S. Clair. Hierarchical uncertainty metatheory based upon modal logic. *International Journal of General Systems*, 21(1):23–50, 1992.

[113] B. Reusch and H. Thiele. On modal fuzzy approximate reasoning. In *Proceedings of the Joint 9th International Fuzzy Systems Association World Congress and 20th North American Fuzzy Information Processing Society International Conference (IFSA/NAFIPS)*, volume 4, pages 1930–1934, Vancouver, Canada, July 25-28, 2001.

[114] M. Russo. FuGeNeSys-a fuzzy genetic neural system for fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 6(3):373–388, 1998.

[115] M. Russo. Genetic fuzzy learning. *IEEE Transactions on Evolutionary Computation*, 4(3):259–273, 2000.

[116] L. Sánchez, J. Casillas, O. Cordón, and M. J. D. Jesús. Some relationships between fuzzy and random set-based classifiers and models. *International Journal of Approximate Reasoning*, 29(2):175–213, 2002.

[117] M. Setnes, R. Babuska, and H. B. Verbruggen. Rule-based modeling: precision and transparency. *IEEE Signal Processing Magazine*, 28(1):165–169, 1998.

[118] W. Shan, D. Jin, W. Jin, and Z. Xu. VLSI implementation of a self-tuning fuzzy controller based on variable universe of discourse. In L. Wang and Y. Jin, editors,

Fuzzy Systems and Knowledge Discovery, volume 3613 of *Lecture Notes in Computer Science*, pages 1044–1052. Springer Berlin Heidelberg, 2005.

[119] Q. Song and N. Kasabov. WDN-RBF: weighted data normalization for radial basic function type neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, volume 3, pages 2095–2098, Budapest, Hungary, July 25-29, 2004.

[120] P. S. Szczepaniak, P. J. G. Lisboa, and J. Kacprzyk. *Fuzzy systems in medicine*, volume 41 of *Studies in fuzziness and soft computing*. Physica Verlag Heidelberg, New York, USA, 1st edition, 2000.

[121] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1): 116–132, 1985.

[122] J. Talaq and F. Al-Basri. Adaptive fuzzy gain scheduling for load frequency control. *IEEE Transactions on Power Systems*, 14(1):145–150, 1999.

[123] H. Thiele. On approximate reasoning with context-dependent fuzzy-sets. In *Proceedings of the Fourth Biannual World Automation Congress (WAC '2000)*, Wailea, Maui, Hawaii, USA, June 11-16, 2000.

[124] H. Thiele. On the concept of qualitative fuzzy set. In *Proceedings of the 29th IEEE International Symposium on Multiple-Valued Logic (ISMVL '99)*, pages 282–287, Freiburg im Breisgau, Germany, May 20-22, 1999.

[125] J. Togelius and S. M. Lucas. Arms races and car races. In T. Runarsson, H.-G. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 613–622. Springer-Verlag, 2006.

[126] J. Togelius and S. M. Lucas. Evolving robust and specialized car racing skills. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC '06)*, pages 1187–1194, Vancouver, Canada, July 16-21, 2006.

[127] J. Togelius and S. M. Lucas. Evolving controllers for simulated car racing. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC '05)*, pages 1906–1913, München, Germany, July 19-22, 2005.

[128] J. Togelius, S. M. Lucas, D. T. Ho, J. M. Garibaldi, T. Nakashima, C. H. Tan, I. Elhanany, S. Berant, P. Hingston, R. M. Maccallum, T. Haferlach, A. Gowrisankar, and P. Burrow. The 2007 IEEE CEC simulated car racing competition. *Genetic Programming and Evolvable Machines*, 9(4):295–329, 2008.

[129] A. P. Topchy, E. P. Topchy, O. A. Lebedko, and V. V. Miagkikh. Fast learning in multilayered neural networks by means of hybrid evolutionary and gradient algorithms. In *Proceedings of the 1996 International Conference on Evolutionary Computation and Its Applications (EvCA '96)*, pages 390–399, Moscow, Russia, June 1-4, 1996.

[130] G. Tsekouras, H. Sarimveis, E. Kavakli, and G. Bafas. A hierarchical fuzzy-clustering approach to fuzzy modeling. *Fuzzy Sets and Systems*, 150(2):245–266, 2005.

[131] G. E. Tsekouras. On the use of the weighted fuzzy c-means in fuzzy modeling. *Advances in Engineering Software*, 36(5):287–300, 2005.

[132] R. M. Turner. Determining the context-dependent meaning of fuzzy subsets. In *Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT '97)*, pages 233–242, Rio de Janeiro, Brazil, February 4-6, 1997.

[133] D. Wang, X.-J. Zeng, and J. A. Keane. A survey of hierarchical fuzzy systems. *International Journal of Computational Cognition*, 4(1):18–29, 2006.

[134] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.

[135] M. Widjaja and L. F. Sugianto. Context-based fuzzy system for optimization. In

Proceedings of the 2002 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '02), volume 1, pages 104–109, Honolulu, USA, May 12-17, 2002.

[136] Wikipedia. Smooth function — Wikipedia, the free encyclopedia, 2011. URL http://en.wikipedia.org/wiki/Smooth_function.

[137] Wikipedia. Selection (genetic algorithm) — Wikipedia, the free encyclopedia, 2012. URL http://en.wikipedia.org/wiki/Selection_%28genetic_algorithm%29.

[138] Wikipedia. DARPA — Wikipedia the free encyclopedia, 2012. URL http://en.wikipedia.org/wiki/DARPA.

[139] B. Wymann. TORCS robot tutorial, 2005. URL http://www.berniw.org/.

[140] R. R. Yager and D. P. Filev. Generation of fuzzy rules by mountain clustering. *Journal of Intelligent and Fuzzy Systems*, 2(3):209–219, 1994.

[141] J. Yen and L. Wang. Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 29(1): 13–24, 1999.

[142] L. Zadeh. A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, 2(3):4–34, 1987.

[143] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[144] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(1):28–44, 1973.

[145] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning - I. *Information Sciences*, 8(3):199–249, 1975.

[146] L. A. Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*, 90(2):111–127, 1997.

[147] W. Zhang, C. T. Yu, B. Reagan, and H. Nakajima. Context-dependent interpretations of linguistic terms in fuzzy relational databases. In *Proceedings of the 1995 International Conference on Data Engineering (ICDE '95)*, pages 139–146, Taipei, Taiwan, March 6-10, 1995.

[148] M. Zirkohi, M. Fateh, and A. Akbarzade. Design of radial basis function network using adaptive particle swarm optimization and orthogonal least squares. *Journal of Software Engineering and Applications*, 3(7):704–708, 2010.