A Precalculated Adaptive Fuzzy Controller for Real-Life Environments

Christoph Kattmann

February 14, 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Conceptual Formulation

Robotics and Artificial Intelligence have been vibrant fields of researach in the past years and, on the way towards a truly intelligent machine, will keep on producing great results and even greater challenges. Some of the domains that humans are still much better at than robots is the ability to master complex control situations, adaptation and learning, and cooperation. A soccer player that follows orders of his coach, positions himself according to the opponents behaviour, kicks the ball in exactly the right manner for a pass or a shot and communicates and coordinates with his team performs a fantastically complex control task with thousands of variables and, astonishingly, does so without an exact model of physics or of the other players. This ability to quickly analyse and predict situations like the ones in a soccer match solely from intuition and experience is currently still unrivaled by computers.
There are a couple of theories that aim to mimic those capabilities. The problems that are faced are however mostly very special and individual, so no general, unified theory is available. In this thesis, an algorithm to adapt behaviour of a RoboCup robot is developed and introduced, as well as tested. The input is provided by sensors like the omnidirectional camera on top of the robot and by referee events delivered to all of the robots during the game. The output of the algorithm is a set of parameters which control the robots behaviour. So the whole system can be represented as a control system where the robot is the controlled process and the algorithm is the controller. However, common controller design methods fail here, the special difficulties of this problem lie in the incertainty of the input signal

and the highly non-deterministic environment. For the design of a controller for this kind of process, new methods and even controller components have to be developed.

1.2 The RoboCup

In order to test and rate the performance of any algorithm, test problems, called benchmarks, are needed. Sorting algorithms are evaluated using random or in some way presorted datasets to understand their performance in different scenarios. The same has to be done with algorithms in artificial intelligence. For algorithms which try to simulate human thinking in general, the most famous benchmark is the Turing test, in which the system has to fool a user who is asking questions to think the responding unit is in fact human. To date (2011) no system has come close to passing the test consistently. There are however other domains of AI research which can not be as easily tested as the capability of a system or algorithm to respond to questions like a human. These include cooperation, learning and sensor and actor technology. The RoboCup is a yearly competiton where exactly these domains are benchmarked in soccer games between teams of robots from universities from all around the world.
The rules of the contest are very close to the official soccer rules stated by the international soccer federation FIFA. Some adaptations are necessary to make participance in the competition affordable and to rid the game of elements which are unnecessary when robots are playing, like a dedicated referee who is present on the field. Other elements of the game like throw-ins and offside rules are slowly introduced into the competition due to their difficulty.
There are five different leagues with different rules. The university of Stuttgart takes part in the Middle-Size league with non-humanoid robots of a maximum of 50cm in diameter and a height of approx. 1m. They play in teams of five robots each with standard soccer balls on a field which is scaled to a length 18m, but otherwise nearly identical to the standard soccer field outlined by FIFA regulations. The main focus of this competition is on image understanding, cooperation and planning. Other leagues have other regulations and a different focus.
What makes the RoboCup scenario so valuable for research in Artificial Intelligence is its focus on a dynamic environment and incomplete information accessibility, that means the input of the sensors of the robots is never entirely certain. The principles and methods used to compete in the RoboCup are useful in other domains and scenarios.

1.3 Related Work

The analysis and control of non-deterministic processes has been subject to research since the dawn of modern control theory in the 1950s. In mathematics, researchers have usually tried to isolate the difficulties of those systems by limiting the chaotic behaviour to a clearly defined, theoretic bounds.
H. Tong et al. ([TCC$^+$95], [CT92]) , for instance, have explored non-deterministic time-series relevant to his field of economics and statistics. They use very theoretic approach to predict and control chaotic systems.
A popular system for the analysis of non-deterministic behaviour is the Markov chain. The estimation and control of Markov chains has been explored by Mandl [Man74] and Aicardy et al. [ADM87] amongst others. Borkar and Varaiya [BV79] even proposed an adaptive approach. Their work is very interesting for the mathematical understanding of chaotic processes, but it is limited to precisely defined time-series and is therefore unapplicable to less clearly defined systems. The methods could however be applied to subproblems in the RoboCup.
More similar problems have come up in computer science and robotics, and the domain of *soft computing* has emerged as a field of research that deals with the identification and control of highly complex systems.
One approach that has been applied in artificial intelligence is rough set theory by Z. Pawlak ([Paw82] [PS94]). As a way to represent uncertainty, it is related to fuzzy systems, but its general notion is even more universal.
Fuzzy Systems themselves have been subject to research ever since their introduction by Zadeh [Zad65]. Fuzzy controllers [Lee90] have proven to be effective and performant tools for the control of complex systems. They have been applied to chemical process engineering [AB05], electrical engineering [BPC00], [ACR99] and robotics [RPW$^+$03].
However, there is, to our knowledge, no published work which applies a fuzzy controller to real-time system as complex and non-deterministic as this one. The existing applications are mostly limited to systems with clear and reliable sensor inputs and a limited complexity. There are however many approaches on adaptive fuzzy systems, like [LT03] and [Wan93]. These however focus on theoretic concepts, they are not applicable in a time-critical environment like robocup.
In the end, there is no approach that can be easily adopted for the problem at hand. Although many insights of the investigated approaches can be used, a very specific system like the robocup calls for an individually tailored solution.

1.4 Overview

In this thesis, at first the exact problem and the theoretical foundations of the solution method are introduced, before outlining the devised system itself.
Chapter 2 further describes the difficultes that occur and introduces the exact role and characteristics of both the controlled process and the controller. Then, two possible design methods, artificial neural networks and fuzzy systems are presented and compared. The decision for fuzzy systems is explained and an early outlook on the system is made.
In chapter 3, the theory of fuzzy systems is introduced as detailed as necessary. After the foundations of fuzzy logic, the Mamdani and Takagi-Sugeno controller are explained as a basis to the controller that is devised later. Also, an example that is used throughout the thesis is established.
Then, in chapter 4, the actual fuzzy system is presented. All underlying principles and ideas are introduced. The three modules construction, fuzzy system and adaptation are presented and details on the implementation are given. At last, the adaptation mechanism is introduced in detail.
Chapter 5 contains the testing of the system for a specific problem, the adaption of the shooting mechanism. The properties of this special problem are introduced, and the complete setup of the system is outlined. At last, the results are presented.
Finally, chapter 6 presents an outlook on further features that have not been covered in this thesis, but are worth researching. They include an extension of the system to a more extensive environment, the use of evolutionary algorithms for the adaptation and a cooperation between the team of robots.

Chapter 2

Problem Statement and Approaches

2.1 General Problem Description

As introduced in chapter 1.1, the problem of controlling the robots behaviour can be stated as a control problem. It is however important to be clear about the bounds of the systems that are used. A system in the sense of control theory is, simply put, any entity with a defined input and output. The first system of the control loop that is going to be introduced later is the controlled process, also called plant.
Here, the plant is the robot and its environment, so in a sense the entire game. Its input are the parameters that are used to steer behaviour of the robot. They are used by the robots internal software to operate and regulate the actuating elements, in the example of a RoboCup player the shooting and driving mechanism. These actions are visible on the field and lead to reactions by the opponents and the own team. The reactions are picked up by the sensor the robot possesses. Each robot of the Stuttgart team has an omnidirectional camera to constantly observe its surrounding within a certain radius. Using Image Processing and Understanding technologies, the position of the robot itself, of other robots and of the ball can be calculated. However, the results are always inflicted with uncertainty, even after data fusion of the whole teams observations. Sensor technology and sensor data processing are a big area of research and progress in the RoboCup, but for now, the steering systems have to deal with a certain degree of uncertainty regarding the current state of the game.
It is also nearly impossible to predict the state of the game except for very

simple situations and very short periods of time. This is an important factor that makes this plant very hard to deal with using standard control theory methods. Unlike classical controller design problems where the plant can be simulated and its output to a certain input can be calculated, the plant in this problem behaves in a way that can be called chaotic, at least non-deterministic.

To make matters worse, the plant changes its behaviour completely from time to time, at the latest when the opponents change. Parameters that consistently yielded good results with one opponent may be useless against the next one.

Another factor that makes this process unusually difficult is the nature of the output. Outputs of a plant are only relevant if they can be measured. In theory, the positions, directions and speed of all the players are visible and measurable, but, as mentioned earlier, the robots can detect only part of those with a limited certainty. The most available and reliable information that the sensors of the robot deliver is its own state, i.e. its position and speed. Together with the posistion of the ball and the players of the opposing team this is the most basic and raw data that the robots sensor data processor provides. In theory, the actual image stream from the camera is an even more basic data source, but it is not practically useable and delivers no data without heavy processing.

The referee events are another sort of information that can be understood as an output of the plant. During RoboCup matches, the referee is a human. Whenever an intervention is needed, he broadcasts a signal to all robots. The necessary interface and behaviour is built into all participating robots. The transmitted information are events like 'goal for team 1', 'corner kick' and 'throw-in'. Following for instance a shot on the goal, these events are the only way to evaluate the succes of the action. The omnidirectional camera is not yet capable to track a shot and observe if it hits the goal, gets deflected or misses altogether.

Figure 2.1 shows a the simplest representation of the plant with input and output. The input is, as discussed, a set of parameters which influence the robots behaviour, the output is the robots current state and discrete referee events. The plant hence contains the part of the robot control system that processes and interpretes the given input parameters, the actuators of the robot, the whole game environment including the ball and enemy behaviour, the sensors detecting raw data and the processing of the sensor data to produce the discrete observations that are the output of the plant. Thus, the plant represents not only the robot, but the game. Figure 2.2 outlines the inside of the plant.

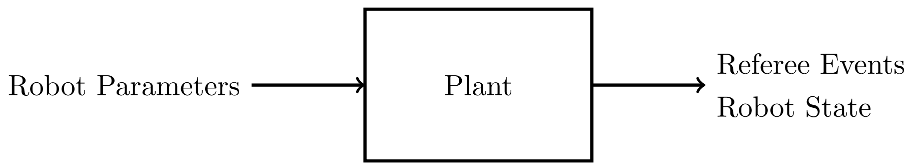

Figure 2.1: Graphic representation of the controlled process with a set of parameters controlling the robots behaviour as the input and the generated referee events and sensor input on the state of the robot as output

In figure 2.2, the block called 'environment' also contains the referee. By

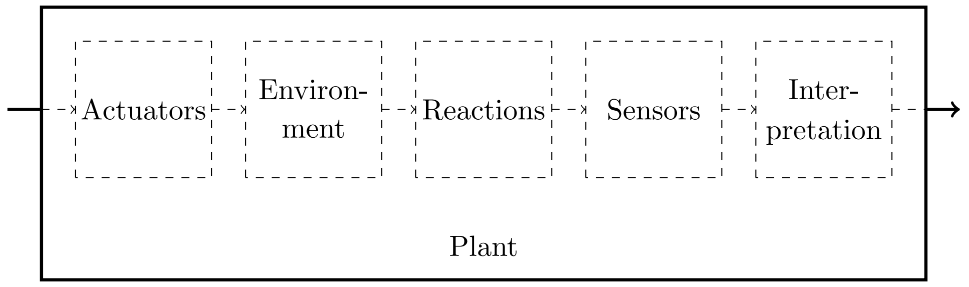

Figure 2.2: Isolated elements of the controlled process as subsystems; the plant and the inputs and outputs are the same as above

seeing him as a part of the game and the broadcast receiver as a part of the robots sensors, the structure is valid for both types of output.
The task in this thesis is to devise a controller, as seen in figure 2.3, that influences the plant and therefore the game in a desired way. It takes the plant output as input and produces a set of parameters that are in turn the input for the plant. This control loop is not, as in classical control problems, executed continuously, but only when needed. The desired way in which we want to influence the plant is also not as clearly available as in classical problems. Instead of a reference value, which can be compared to the plant output to obtain the control error, the events only allow for an unprecise, symbolic desired outcome. It is however generally possible to establish a

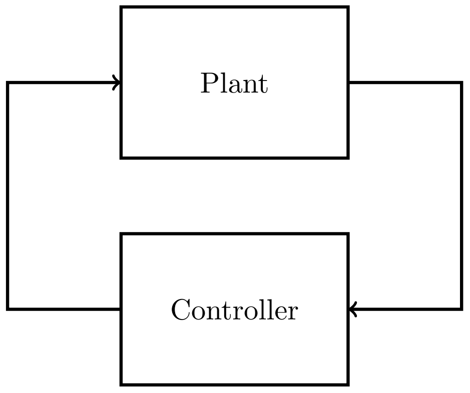

Figure 2.3: The plant with a controller that takes the plants output as input and produces output that goes into the plant as input. The goal is to design the controller in such a way that the output of the plant behaves in a desired way.

certain hierarchy between the different observations and to specify the best-case scenario. Scoring a goal is better than missing, completing a pass is better than losing the ball and it would be perfect if every pass reached its destination and every shot would be a goal. This is what is gained from strong processing of the robots sensor input. If the controller used more of the available raw data as input, the desired value would be less clear and the interpretation would be another source of error.
It is clear that the controller can not continuously influence the parameters of the robot. There is no way to obtain the complete, optimal parameter set in any given moment. The parameters can only be reasonably changed in situations where some parameters have to be recalculated, like a pass or a shot on the goal, or when the available information is precise enough to evaluate the performance of the robot in certain domains.
In order to control this plant the controller needs to be radically different from a classical PID-controller that is so useful in many other situations. The symbolic and uncertain output of the plant defies any efforts to calculate a control error. To deal with the general uncertainties, the controller needs to be exceptionally robust, i.e. it needs to produce acceptable values with defective and error-ridden input. As it is impossible to compute a model of the plant (which includes the non-deterministic behaviour of enemy players), the controller also needs to be adaptive. Over a period of time,

the quality of the results produced by the controller should increase. This is necessary because it is impossible to determine the optimal configuration for the controller by analyising the plant. Also, this mechanism should enable the controller to adapt to changes in the environment, for instance changing opponents.
The discrete, event-like output of the plant which is provided by the referee

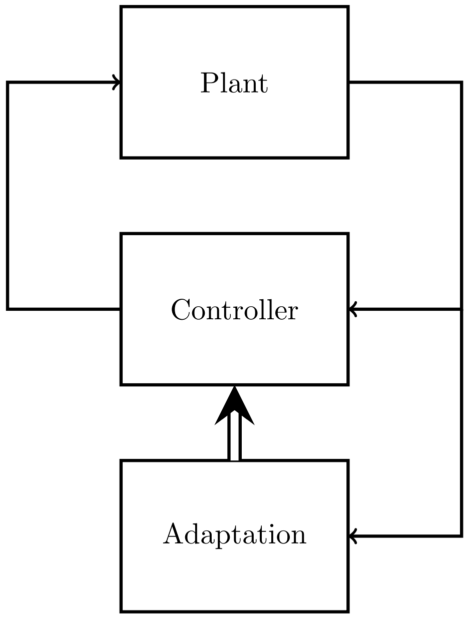

Figure 2.4: The control look with the adaption mechanism. The adaption module does not influence the plant, but only changes the controller based on the outputs of the plant which allow an evaluation of the controllers performance. Generally, this holds true for the referee events, but other measured values may provide insight on flawed controller behaviour.

is used by the adaption mechanism to evaluate the performance of the controller, while the controller uses mainly the information about the state of the robot to calculate the parameters that are the input of the plant. This seperation of tasks is visible in figure 2.4. Still, the same input is available to both entities.
In the case of the robots of the team CopS of the University of Stutttgart there already exists a mechanism to dynamically calculate some parameters during the game, while others are set by hand and only changed manually

during matches , if ever. This mechanism should not be replaced, but extended and equipped with the capabilities of a learning system. To meet this requirement, the new controller does not replace the old system, but rather exists beside it. The outputs of both systems are added up to obtain the final parameters. The new system therefore acts as a corrector to the old system. Figure 2.5 shows the complete control loop including the original, 'old' controller.
To summarize, the task is to devise the controller and the adaption mechanism so that the performance of the plant (i.e. the success of the robot soccer team) is enhanced and to add learning capabilities to the robots control system.
The plant has the following properties:

- The input is a set of parameters,
- Its inner workings are hidden (black-box behaviour),
- The enviroment and therefore the plant change over time,
- The outcome is hence highly non-deterministic,
- The output is inflicted with incertainty,
- The output is, in part, a series of symbolic events.

These characteristics call for a controller with the following properties:

- It has to deal with uncertain input,
- It has to be robust against uncertainties in the controller design,
- It needs to be adaptive,
- It needs to fit into the existing infrastructure.

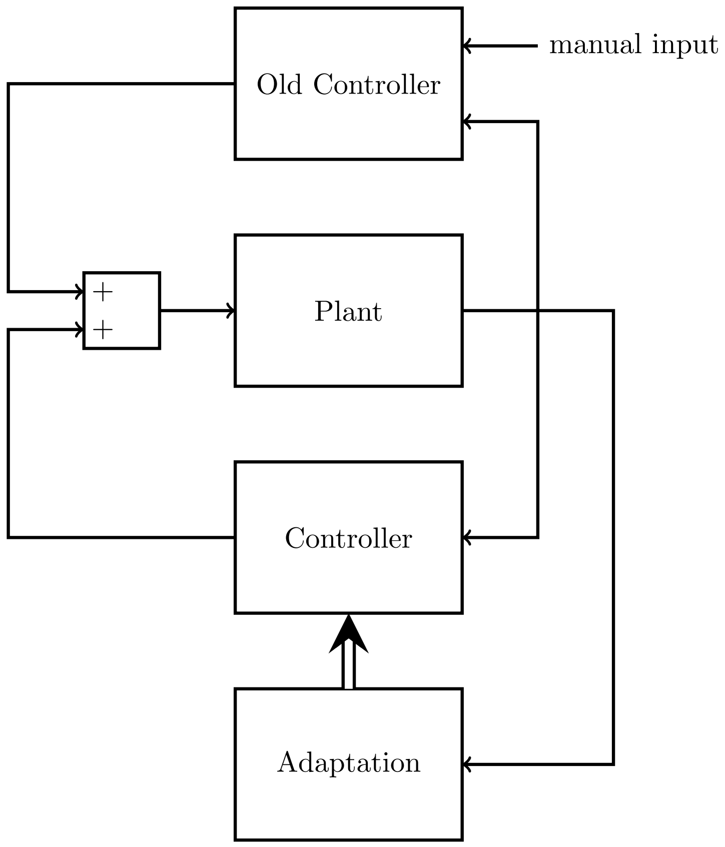

Figure 2.5: Complete control loop including the 'old' controller that also influences the input parameters of the plant. The outputs of both controllers are added up, so the 'new', adaptive controller acts as a corrector to the original system.

2.2 General Approaches

The desired properties of the controller under the given circumstances don't allow for a classical controller design. With a plant as chaotic as this one, engineers often resort to soft-computing methods like neural nets or fuzzy systems. Both have different advantages and disadvantages, and have proven to be effective in real-world applications.

2.2.1 Artificial Neural Nets

Artificial Neural Nets (ANN), first developed by McCulloch and Pitts in 1943 in [MP43] try to mimic the functionality of human brain cells in order to approximate highly nonlinear and multidimensional functions. They are a vibrant topic of research, and introductions are available and numerous.
The basic component of an ANN is the *neuron*, which is often modeled as simple entity which sums up the inputs from other neurons or from an input interface over time. When the accumulated input reaches a certain *activation level*, the neuron, as its called, *fires* and delivers an output to all neurons in the net which are connected to the output.
There are many propositions on how to systematically construct an Artificial Neural Net in order to limit the implicit complexity without crippling the capabilities of large nets with a high number of neurons. The most successful and famous attempt is probably the *perceptron* by Rosenblatt [Ros58], which was developped into the *multilayer perceptron* and successfully used in a number of instances in robotics, like [LYL96] and [LH97]. An overview over the usage of ANN in robotics is given in [HJV90].
Areas where Artificial Neural Nets show great capabilities are classification and function regression, including prediction. The usage of a neural net makes sense for highly complex systems like the plant in this thesis because they require no prior knowledge about the properties of the controlled system at all. When initialized, the ANN contains no specific information, the necessary behaviour has to be trained using one of the available training algorithms like backtracking. A number of training samples have to be provided to the learning algorithm and in some sense replace the prior knowledge. Even after the learning is completed and the ANN delivers correct results, it is impossible to derive any rules of the parameters of the net, it shows a black box behaviour at all times. The learning process can and should go on when the net is already in use, ANNs are capable of adapting to a changing system.

2.2.2 Fuzzy Systems

Fuzzy Systems, on the other hand, use different principles. The theoretical foundation, Fuzzy Logic was introduced by Lofti A. Zadeh in 1965 [Zad65]. Fuzzy Logic is different from normal, boolean logic in that there are more states than just true and false, it establishes a mathematical possibility to grasp statements like nearly true, barely true, and halfway true. This is used in Fuzzy Systems to deal with uncertain information and verbally given rules. These rules can be vague statements like "If x is low, make y high", they are coded into one or more Fuzzy Systems and incrementally control the plant towards a hopefully stable output. Because of the these rules, which must be given by a human who has at least some knowledge of the plant, these systems are often called expert systems.

2.2.3 Comparison

Both Artificial Neural Nets and Fuzzy Systems do not require a mathematical model of the controlled system. They are however different in other domains as shown in table 2.1.

Artificial Neural Nets	**Fuzzy Systems**
no mathematical model necessary	no mathematical model necessary
no prior knowledge necessary	prior knowledge essential
built-in learning algorithm	nontrivial learning algorithms
black box behaviour	interpretation possible

Table 2.1: Comparison between ANN and Fuzzy Systems

There are proposals to combine the strenghts of both approaches. Often called Neuro-Fuzzy Methods, these are a vibrant topic of resarch, but a general consensus on an ideal combination has yet to be found. Notable papers on the topic include [MH00], [BH94] and [BY95].
The desired properties of the controller cannot be reached by either system in their purest forms. Artificial Neural Networks need many learning cycles to be entirely functional, Fuzzy Systems on the other hand do not come with a learning mechanism. The weakness of the ANN, however, is critical and

practically rules it out. Training a Neural Net to deliver consistently good results would take too long in the RoboCup. That does not at all mean that they are generally useless in robotics, when the plant is readily available they can deliver excellent results.

This weakness of Artificial Neural Nets is the strength of Fuzzy Systems. If the prior knowledge is correct and sufficiently exact, the controller operates perfectly from the start.

One important task in this thesis will be the development of an adaption mechanism for Fuzzy Systems. Sadly, there is no known approach to mimic the ingenious learning algorithms that exist for Artificial Neural Nets. However, by interpreting the adaptation mechanism as an optimization problem, all the reliable tools of parameter optimization are available. A Fuzzy System is exactly defined by a set of parameters, and all possible sets form an optimization space, whose dimension is equal to the dimension of the set. The optimization is now a search in this optimization space for the perfect set of parameters. There are many effective algorithms for optimizing a set of parameters, like Hillclimbing and evolutionary algorithms. This topic will be further discussed in section 4.3.

Chapter 3

Theory of Fuzzy Systems

3.1 Capabilities of Fuzzy Systems

Fuzzy Systems are based on Fuzzy Logic. This theoretic concept is a generalization of the classical, Boolean Logic in that it allows more states than just 1 and 0. In Boolean Logic a statement is either true or false, there is no middle ground. In Fuzzy Logic a statement can also be partially true, which is represented by any value between 0 and 1. This concept makes it possible to represent uncertainty and doubt in statements. While not particularly useful in classical logic, the principle of Fuzzy Logic is a very powerful tool to construct controllers which deal with uncertain values.
Fuzzy Logic was conceived following the observation that humans do not require precise information, and are still able to master highly complex control tasks. Some of these tasks have not even remotely been mastered by machines. Robots are just now learning to walk smoothly, a task humans have mastered ages ago without precise numerical input and even without any knowledge about the physical laws of gravitation and friction. The mechanisms we humans use are robust and adaptive, and researchers use Fuzzy Logic in an attempt to mimic these capabilities.
There are actually two different sorts of uncertainty that Fuzzy Logic can handle. *Lexical Uncertainty* is present if something is not described using a sharp value, but a word or phrase that only describes the region the value is in. Saying 'It is warm outside' instead of 'It is $25°C$ outside' is an example of Lexical Uncertainty. This vagueness of information often occurs when the information is provided by humans. If the value itself is imprecise and doubtful, it is called *Stochastical Uncertainty.* A temperature sensor which measures an outside temperature of $25°C$ but is known to have a $\pm 1°C$

margin of error is inflicted with Stochastical Uncertainty.
Fuzzy Logic is a great tool to elegantly deal with both types of uncertainty. The tolerance of Lexical Uncertainty is a unique strength of Fuzzy Logic. This feature is useful because it allows the construction of a controller from simple, verbally given rules.
A controller is in every case just a function that maps a set of input values to a set of output values. This also holds true for a controller based on Fuzzy Logic. A Fuzzy Controller represents a possibly nonlinear and highly complex function, but the function does not have to be provided in detail, it is implicitly hidden behind a set of *rules* that are often intuitive.
A problem that occurs in the RoboCup and that a Fuzzy Controller could solve is the regulation of shoot strength. One can imagine that the distance of the robot to the goal is an important factor in the calculation of the optimal shoot strength. However, the exact, optimal relationship between the variables 'distance' and 'shoot strength' is mostly unknown, there can only be given the general nature of this relationship, formulated as rules. The basic rules that a coach would give (albeit he probably would not need to) to a player could be phrased as follows:

- If the distance to the goal is big, shoot strength should be high.
- If the distance to the goal is small, shoot strength should be low.

Note that these rules do not give exact instructions. They do however outline the general behavior that is expected from the player. A Fuzzy Controller is capable to overcome the Lexical Uncertainties and to use these kind of rules to replace a classical controller that for instance uses a map that has to be manually filled with correct values. In order to construct a Fuzzy Controller for this example problem, a bit of theory in Fuzzy Logic is necessary.

3.2 Basics of Fuzzy Logic

In 1965, Lofti A. Zadeh founded Fuzzy Logic by introducing the notion of Fuzzy Sets.

Definition 1. *Fuzzy Set*
A Fuzzy Set is a pair (X, μ) where X is a set and

$$\mu : X \to [0, 1] \tag{3.1}$$

is the Membership Function that assigns every $x \in X$ a grade of membership.

Using the example introduced above, $x \in X \subset \mathbb{R}^+$ could be the distance from the robot to the goal. A Fuzzy Set is now used to map the distance to a *Linguistic Value* and ultimately convert the numerical variable 'distance' to a *Linguistic Variable.*

Definition 2. *Linguistic Variable*
A Linguistic Variable A is a non-numeric variable whose values a, called Linguistic Values are symbolic terms and not exact numbers.

Again using the example, the conversion from numerical to linguistic variable in the case of 'distance' means mapping every numerical distance to terms like 'far' and 'near'. It is of course not possible to clearly state when the distance to the goal is far or near. However, it is inherently obvious that 10m, more than half of the length of the field, is far, and 1m is not far, but near. To grasp the expression of for instance 'far' in a classical, boolean logic, it would be necessary to define a threshold distance. One could decide that every distance from 6m onwards is 'far'. Figure 3.1 shows the logical mapping of the value 'far' by this definition using boolean logic.
This logical mapping can be described by a *Crisp Set.* It is different from a Fuzzy Set in that it uses the mapping

$$\mu : X \to \{0, 1\}. \tag{3.2}$$

This leads to a discontinuous membership function like the one in figure 3.1. Crisp Membership Functions are arbitrary definitions that don't mimic human thinking and may also cause problems because of the sudden change in state that occurs when the threshold is crossed.

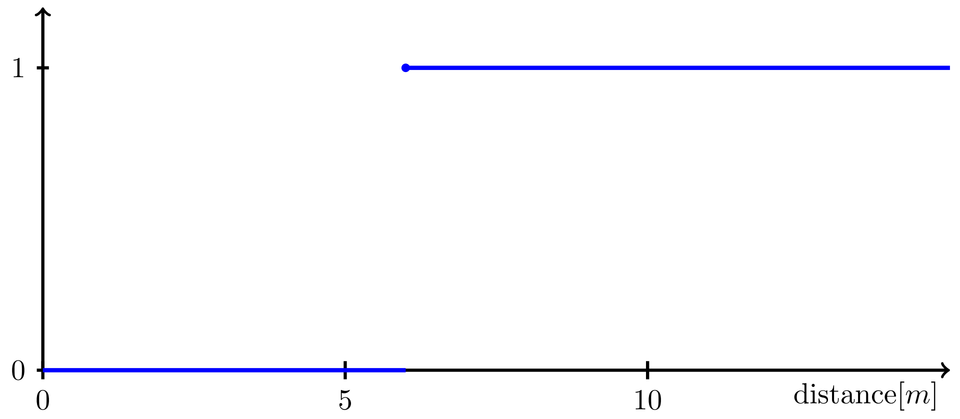

Figure 3.1: Boolean Mapping of the Linguistic Variable 'far' with a threshold value of 6

Using Fuzzy Logic it is possible to define a fluid, continuous Membership Function of a Linguistic Value. In the case of the phrase 'far', the Membership function could look like outlined in figure 3.2.
A fuzzy Membership Function μ can take all values between 0 and 1 and maps all distances to their corresponding grade of membership to the term 'far'. In this case, 2m is not far, 5m is a bit far, 7m is fairly far and every distance from 8m onwards is definetly far. The grade of membership can be exactly stated as a *truth value* t.

Definition 3. *Truth Value*
The truth value t is defined by

$$t = \mu(x), \tag{3.3}$$
$$t \in [0,1] \tag{3.4}$$

and denotes the grade of membership of the numerical input x to the Linguistic Value which is expressed by the Membership Function μ.

For 2m, the truth value is 0, for 5m and 7m it is 0.25 and 0.75, respectively, and for 8m, the truth value is 1.

This Membership Function is only valid for the specific situation. In a real football game, a distance of 10m to the goal is generally not perceived as far, the function would in this case need to be changed.
Of course, the Membership Function of the Linguistic Value 'near' looks

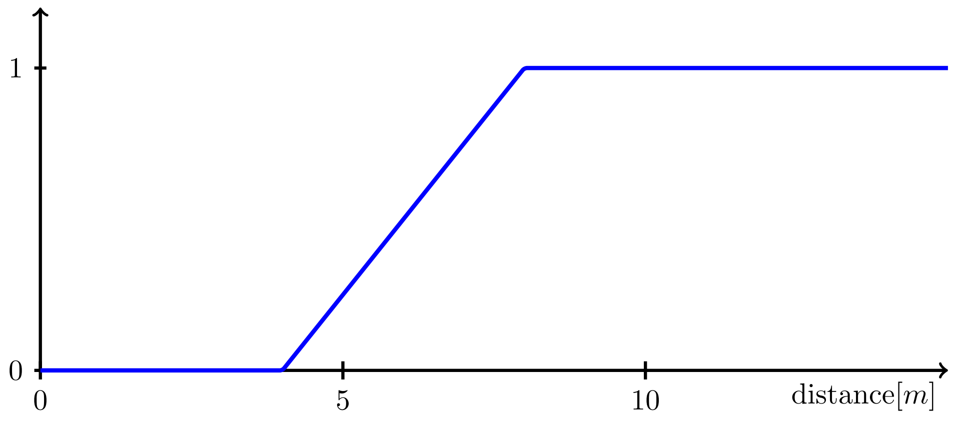

Figure 3.2: Possible Fuzzy Membership Function of the value 'far'

different. Figure 3.3 outlines a possible design.
By putting these two Fuzzy Sets, which use the same set X of numerical input values, but different Membership Functions μ, together, a pair (X, μ^n) of Fuzzy Sets is formed, n in this case being 2. Figure 3.4 shows the resulting plot for 'far' and 'near'. This extended, multidimensional version of a Fuzzy Set provides the aforementioned conversion of numerical to Linguistic Variable. The corresponding function is called *Fuzzy Map*.

Definition 4. *Fuzzy Map*
A Fuzzy Map

$$f := \mu^n : X \to Y$$

maps a numerical value $x \in X$ to a fuzzy state Y.

Definition 5. *Fuzzy State*
A fuzzy state

$$Y \in (a, t)^n$$

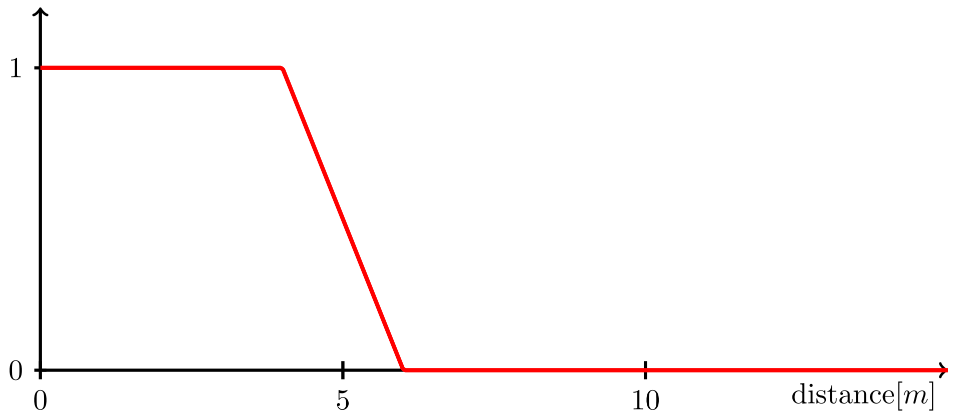

Figure 3.3: Possible Fuzzy Membership Function of the value 'near'

is a set of pairs that assign every Linguistic Value $a_{1...n}$ *a truth value* $t_{1...n} \in [0, 1]$.

For a distance of 5m, the Membership Function for the Linguistic Value 'far' yields a truth value of 0.5 and the one for 'near' a truth value of 0.75. Thus the result is the fuzzy state

$$Y = \{(\text{'far'}, 0.5), (\text{'near'}, 0.75)\}.$$

This result for a distance $x = 5m$ can be interpreted as 'between far and near, but more near than far'. By mapping a crisp numeric distance to a statement like this, a certain amount of precision is definitely lost. In return, controller can now work with statements like the ones in section 3.1. So, by using Fuzzy Logic, a controller can be defined with simple instructions that a human can intuitively provide and understand.
The process of conversion from a numerical to Linguistic Variable is also called *fuzzyfication*. The example of seperating a distance into 'far' and 'near' is of course a very simple one. For more precision after the conversion and more meaningful results it is possible to use more Linguistic Values with more granular Membership Functions. For instance, the fuzzy state

$$Y = \{('far', 1), ('near', 0)\} \tag{3.5}$$

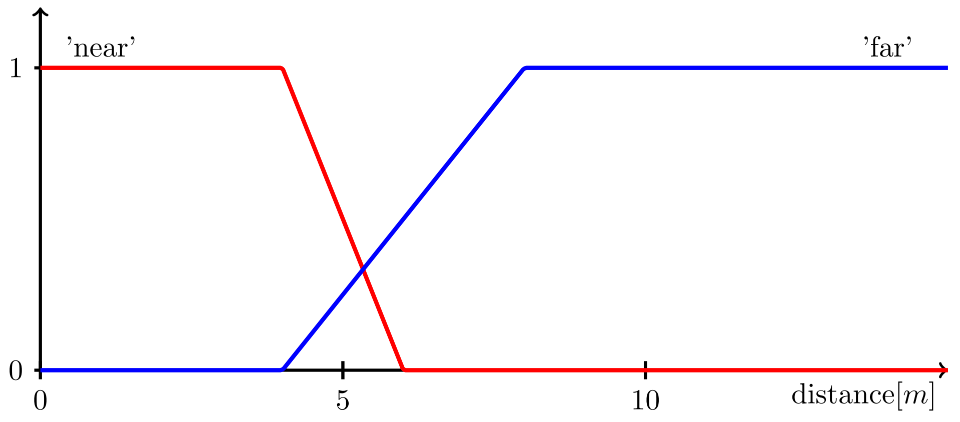

Figure 3.4: Possible Fuzzy Map for distance mapping, combining the individual maps of 'near' and 'far'

does not carry a lot information, the distance could be 8m or more. In the case of the shoot strength controller outlined in section 3.1 this is sufficient, because for any distance greater than 8m the result is simple and the same (high shoot strength). If more Linguistic Values are needed, the Fuzzy Map could be extended by inserting a value in the middle, like shown in figure 3.5 , or by adding another value, like shown in figure 3.6. For an even more continuous behaviour, the membership function can take a sinusoid shape, like shown in figure. It is however often necessary to design the membership function in a piecewise linear way.
As the general notion of boolean logic is extended, it is not possible to use the operations defined on sets of boolean values any more. New definitions for example for conjunction and disjunction have to be found. Especially the conjunction is very important, because a fuzzy controller has to deal with statements like

$$\texttt{distance} \in \texttt{far} \text{ AND } \texttt{goalkeeper} \in \texttt{near}$$

and map them to a single truth value. A widely accepted truth function for conjunction in fuzzy logic is the *t-norm*.

Definition 6. *T-Norm*

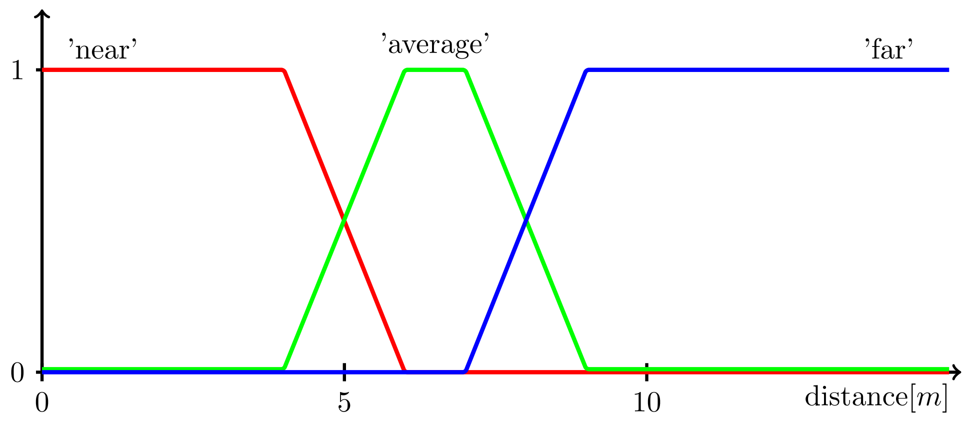

Figure 3.5: Extended Fuzzy Map with additional Linguistic Value 'average'

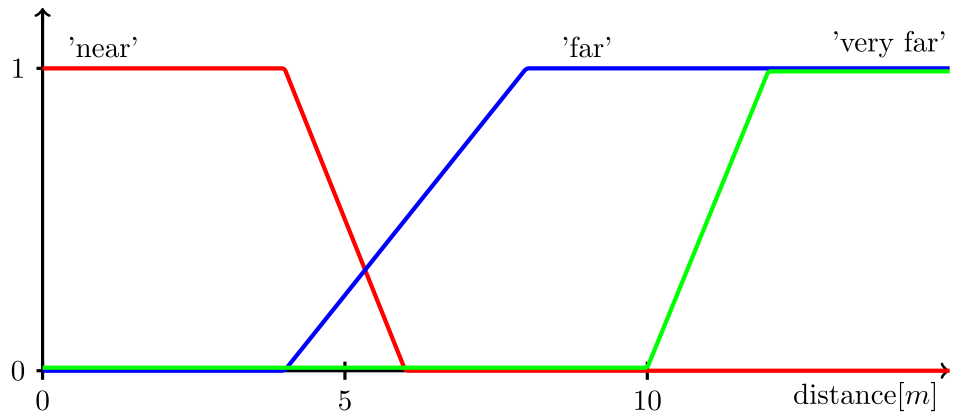

Figure 3.6: Extended fuzzy map with additional linguistic value 'very far'

A t-norm is a map $\top : [0,1] \times [0,1] \rightarrow [0,1]$ *that satisfies the following properties:*

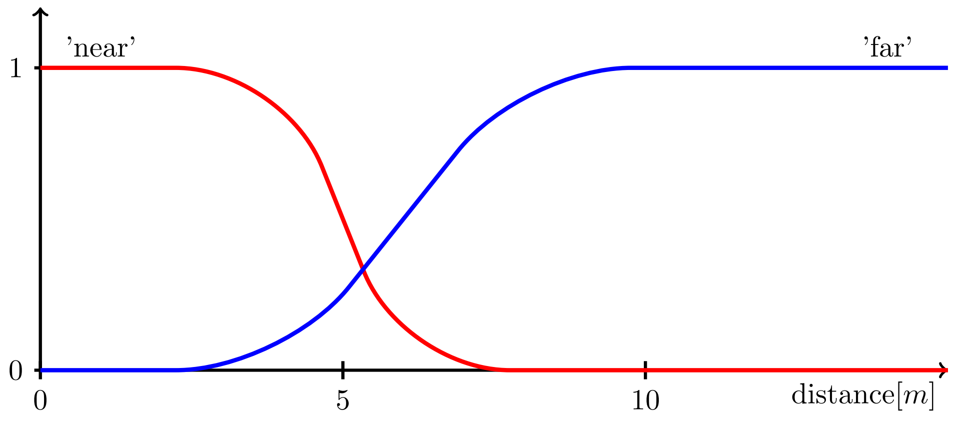

Figure 3.7: Fuzzy map with sinusoid membership functions

- *Commutativity:* $\top(a,b) = \top(b,a)$
- *Monotonicity:* $a \leq c \wedge b \leq c \Rightarrow \top(a,b) \leq \top(c,d)$
- *Associativity:* $\top(a,\top(b,c)) = \top(\top(a,b),c)$
- *Identity Element:* $\top(a,1) = a$

There exist many propositions for t-norms. The simplest and most widely used is the minimum norm, which is

$$\top_{min} := min\{a,b\}.$$

It may not be entirely intuitive, but using this norm for conjunction is continuous and delivers sufficient results in most cases. Another proposition is the product norm

$$\top_{prod} := a \cdot b;$$

which is also continuous, but not idempotent, as

$$\top_{prod}(a,a) \neq a \; \forall a \in (0,1)\,.$$

The selecion of a t-norm depends on the situation, there is no indisputable choice.

Note that the conventions and definitions in this section are not shared by all authors. Fuzzy Logic is a tool used for many purposes, out of pure mathematical interest or to construct expert reasoning systems and controllers among others. The different focus of these applications leads to different definitions and varying rigour in the construction of a mathematical foundation.
For the purpose of this thesis, the selected definitions are sufficient, and most research on fuzzy controllers doesn't rely on a strong mathematical foundation, but more on the general principles of fuzzy thinking.

3.3 Fuzzy Controllers

There many propositions on how to use Fuzzy Logic in a controller. The development and implementations of Fuzzy Controllers has largely taken place in Japan, because Fuzzy Logic was seen as a Artificial Intelligence method and was therefore affected by the so called 'AI winter' during the seventies and eighties in the western world. The breakthrough application for Fuzzy Controller was the autonomous Namboku line of the Sendai subway, which was openend in 1987 (or 1988, sources vary). Accelerating, steering and breaking was controlled by a system based on Fuzzy Logic, and the trains operate smoothly and economically to this day. Since then Fuzzy Controllers have found their way into home appliances and industrial machines all around the globe.

3.3.1 The Mamdani Controller

One of the most basic version of such a Fuzzy Controller is the Mamdani controller. It was introduced by Mamdani and Assilian in 1975. Most Fuzzy Controllers use its structure, albeit often with modifications.
The Mamdani controller consists of four basic modules. They are the aforementioned fuzzyfication, inference, rule base, and defuzzification. Figure 3.8 shows the basic structure of the controller.
Input to and output out of the controller are crisp, numeric values. The modules act as follows:

Fuzzyfication In the example of the shoot strength controller which was introduced in section 3.1 one input is the distance from the player who wants to shoot to the goal. It is provided by the robots sensor system as an exact number of millimeters.
The fuzzyfication of this input is equivalent to the conversion from a numerical to a linguistic variable in section 3.2. Using the simple Fuzzy Map proposed in figure 3.4, the numerical input is mapped to a set of fuzzy values with corresponding truth values. For a distance of 5m, the resulting fuzzy state is

$$Y = \{(\text{'far'}, 0.25), (\text{'near'}, 0.5)\},$$

as presented in section 3.2. With this conversion, the fuzzyfication step is complete. The map with the input mapping is shown in figure 3.9.

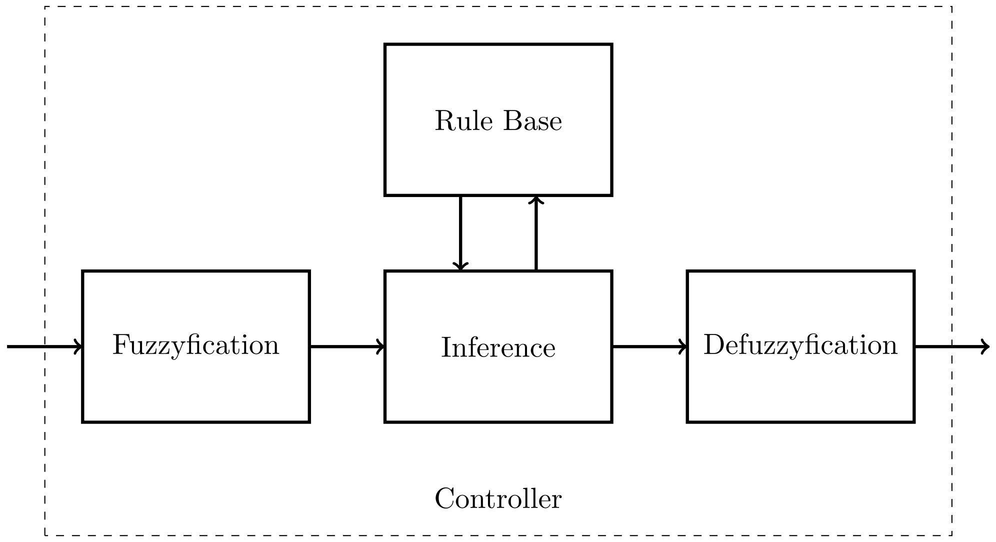

Figure 3.8: General structure of the Mamdani controller. The input and output are both numerical values. The fuzzification step translates them into linguistical values, so that the rulebase can be applied.

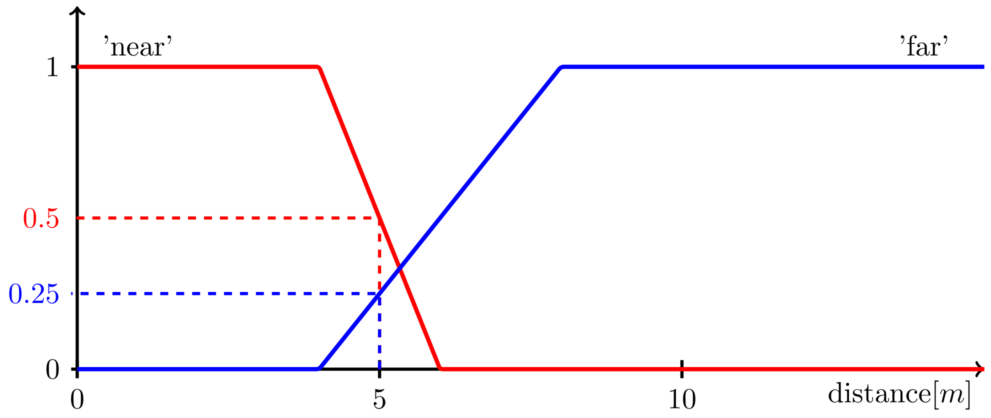

Figure 3.9: Fuzzy map with input mapping. A crisp input of 5m yields a truth value of 0.5 for 'near' and 0.25 for 'far'.

The resulting linguistic variable and its value are forwarded to the inference module. When there are multiple inputs, they are converted in the same way using different fuzzy maps.

Rule Base The rule base is the key component of a Mamdani controller. The exact instructions on how the controller works are stored here, and the correctness and quality of these instructions is vital for a propor control performance. The instructions, called rules, are stored in the rule base. The basic form of a rule is

$$R : \text{IF } \texttt{condition} \text{ THEN } \texttt{consequence}.$$

The conditions are depending on the linguistic input variables, and the consequences are expressed using equally linguistic output variables.

Definition 7. *Rule*
A rule R is an instruction of the form

$$R : \textit{IF } x_1 = a_1 \textit{ AND ... AND } x_n = a_n \textit{ THEN } z = b,$$

where

- *x_k are the inputs as liguistic values,*
- *a_k are the possible linguistic values they can take,*
- *z is the linguistic output variable and*
- *b is a linguistic output value.*

Two possible simple rules for the example used here would be:

$$\begin{aligned} R_1 &: \text{IF } \texttt{distance} = \texttt{far} \text{ THEN } \texttt{shootstrength} = \texttt{high}, \\ R_2 &: \text{IF } \texttt{distance} = \texttt{near} \text{ THEN } \texttt{shootstrength} = \texttt{low}, \end{aligned}$$

These rules have to be provided by humans. They only need to contain linguistic values and no exact numerical values. Therefore, the rules are valid for problems of similar shape, but different scale. If the playing field of the RoboCup would increase in size, only the Fuzzy Maps in the fuzzification step would need to be changed, but not the rule base. It remains valid, as it also would for a human who is able to apply basic rules to new situations. The rules in the rule base are used in the inference module to map the linguistic input variables to the also linguistic output variables.

Inference In this step, every condition in the rules is tested for validity. If the condition holds, the consequence is executed. In boolean logic, this step would need no further explanation. In fuzzy logic, however, there are more cases for a condition than 'it holds' and 'it doesn't hold'. Since conditions can be partially be fulfilled, consequences also need to be partially executed. Both the input A and the output B of this module are fuzzy states. The mapping from input to output is based on the rules in the rule base.
In the simple example, the input is the distance of the player from the goal as a fuzzy state. For a distance of 5m, the input, as previously presented, is

$$A = \{(\text{'far'}, 0.25), (\text{'near'}, 0.5)\}.$$

The first rule, which is

$$R_1 : \text{IF } \texttt{distance} \in \texttt{far} \text{ THEN } \texttt{shootstrength} \in \texttt{high},$$

has a grade of fulfilment of 0.25, because the distance is the only input. If there were multiple inputs, they would have to be conjunctioned using a t-norm. For more than two inputs, the t-Norm has to be used multiple times, which is allowed by the condition of associativity.
In the case of only one input, the truth value of the input state directly translates to the output state. In the case of this rule, the output state B would simply be

$$B_1 = \{(\text{'high'}, 0.25)\}.$$

Similarly, the second rule yields

$$B_2 = \{(\text{'low'}, 0.5)\}.$$

The end result of the inference step in the simple example would hence be the fuzzy output set

$$B = \{(\text{'high'}, 0.25), (\text{'low'}, 0.5)\}.$$

If there is only one output and only one input, this is extremely simple. Usually though there are many linguistic variables for both input and output. If there are multiple inputs, the choice of t-norm influences the behaviour of the controller, it has to carefully chosen.
It is possible that the fuzzy output state contains multiple instances of the same linguistic value with different truth values. For instance, another rule, maybe regarding the distance of the goalkeeper to the goal (and hence his ability to block the shot), could yield the result

$$B_3 = \{(\text{'high'}, 1)\}$$

if the goalkeeper is ready for the shot and can not be overcome with low shoot strength. In this case, there are again multiple choices to combine the truth value into one final output, like maximum, minimum or average. The average value is an obvious choice for most cases, which in this case would produce the final output

$$B = \{(\text{'high'}, 0.625), (\text{'low'}, 0.5)\}.$$

The inference step translates input to output values, but they are still in the form of linguistic variables. The translation from linguistic back to numerical variable is done by the last module, which conducts the defuzzification step.

Defuzzification In order to interprete the linguistic variables to numerical ones, first of all the linguistic output variables that appear in the rulebase have to be defined with a fuzzy map. In the simple example of the shoot strength controller, the output is, of course, the shoot strength. A possible design of the membership functions for the linguistic values 'high' and 'low' is outined in figure 3.11.
The mapping from numerical to linguistic variable is bijective, but the re-

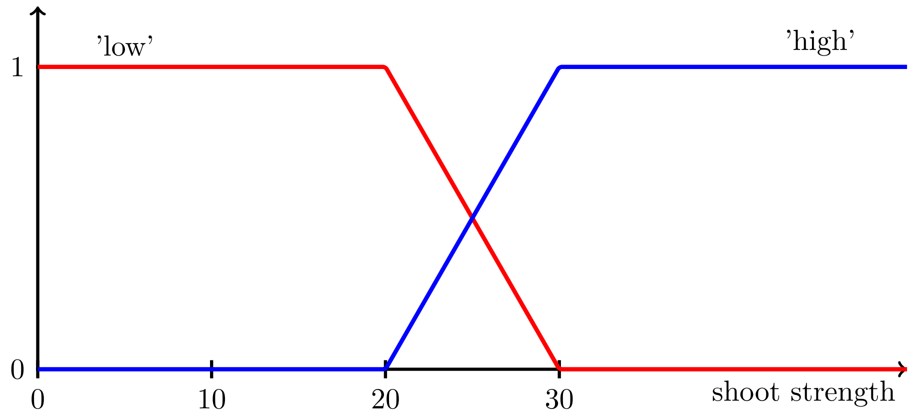

Figure 3.10: Output fuzzy map for linguistic variable 'shoot strength', connecting the linguistic output variables 'low' and 'high' with crisp values.

verse mapping, which is required for the defuzzification, is not unambiguous. There are multiple methods to convert a linguistic variable back to a numeric one, and like for the selection of a t-norm there is no obvious choice. Two widely used defuzzification methods are the *maximum-defuzzification*

(MAX) and the *mean-of-maximum-defuzzification* (MOM).
The maximum-defuzzification is the simplest and most intuitive version. It implies picking the point on the x-axis of the mapping where the truth value of the fuzzy set which is the input to the deffuzification is maximized. A problem with this kind of mapping is that it is not bijective. There may be more than one point with a maximum truth value. One way to determine the final output is to choose a maximum point at random, but this renders the whole controller non-deterministic. A possible solution is to always pick the smallest available value. In the simple example used all through this section, the inference output

$$B = \{(\text{'low'}, 1)\}$$

would lead to the controller output

$$u = 0,$$

which signifies a shoot strength of 0 and therefore does not really make sense. The mean-of-maximum method delivers a better result, as it calculates the controller output as the average of smallest and biggest point with a maximum truth value. For the fuzzy output state B this method would yield

$$u = 0 + \frac{20 - 0}{2} = 10,$$

which makes sense as a controller output when the only information that can be deduced is that the shoot strentgh should be 'low'. For the aforementioned fuzzy output state of

$$B = \{(\text{'high'}, 0.625), (\text{'low'}, 0.5)\},$$

the crisp controller output can be calculated by taking the average of the values that an individual evaluation of both truth values yield. In this case the result is

$$u = 25 + \frac{26.25 - 25}{2} = 25.625.,$$

which also conforms to the intuitive expectaitions. The right choice of a defuzzification method is greatly inluenced by the choice of the fuzzy maps in the fuzzyfication and defuzzification step. The maximum method is valid for detailed, finely grained maps but fails when a broader, less accurate approach is taken. This is certainly the case for the simple example.
A third, widely used defuzzification method is the center-of-area-defuzzification

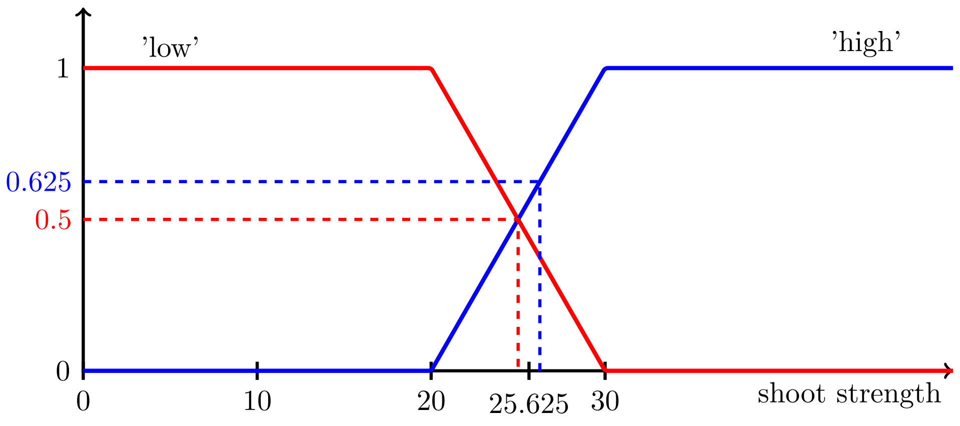

Figure 3.11: Output fuzzy map with mapping of fuzzy state B using the mean-of-maximum defuzzification

(COA). It also depends on more accurate fuzzy maps, but delivers very clean and smooth results. It is often used for classical control tasks that could also be solved using classical methods. For the much more complex tasks of artificial intelligence, this defuzzification method is not particularly useful, as it also requires quite extensive computation.
In summary, the setup of a Mamdani controller requires the following tasks:

1. Define a fuzzy map for every numerical input variable for fuzzification,
2. construct the rulebase from intuitive rules and experience,
3. choose a t-norm that is used in the inference step,
4. define a fuzzy map for every output variable for defuzzification,
5. choose a defuzzification method.

In practice an engineer would conduct each step, evaluate the result and then reexamine the sytem, tweak and optimize it and test until he is satisfied with the result. This is not different from any other controller design. However, a Mamdani controller is a system which delivers points of freedom with a very clear influence. An experienced engineer can in most cases easily recognize which parts of the controller need tweaking.
In the end a well designed Mamdani controller can perform better and more

reliable and robust than a classical PID controller. If the controlled system is simple, the advantages are not worth the effort. If the system is more complex, the choice and calibration of a classical controller like sliding mode, state-space controllers with observers or MIMO-controllers can mean much more effort than designing a fuzzy controller. However, a fuzzy controller restricts the influence of the designer to a more indirect level, and a thorough theoretic stability and performance analysis is not as easy and straightforward as usual.

The really important advantages of a fuzzy controller however lie elsewhere. The seperation of the continuous numerical range of input variables into several linguistic values often represents the true knowledge about the dynamic properties of the system much better than a continuous input-output mapping. The complex differential equations of dynamic controllers can lead to serious problems with stability and robustness, and their undoubted capabilities are, in the case of environments like the RoboCup, not even necessary unless for very specific problems which almost always can be isolated. Robustness and stability are self-evident for a ordinary and well designed fuzzy controller, even for the most complex control tasks.

There is another feature of fuzzy controllers that can be seen as an advantage over classical controllers for some tasks. They have uncertainty, both linguistic and stochastical, built in and is therefore implicitly robust against it. Hence, it is pretty hard to design a fuzzy controller that behaves in an optimal way, but it is easy to design one that is acceptable. This concentration on the basic behaviour of the system is very helpful. Many complex control strategies rely on an exact mathematical description of the plant. If this description contains errors, the controller will behave perfectly in simulations, but may fail in the real world. This gap between lab and real-world conditions is narrower with fuzzy controllers. In the case of the RoboCup it is, as has been discussed in section 2.1, downright impossible to construct a description of the plant. Instead of guessing a rudimentary characterization of the plant, fuzzy logic allows to build the controller on the basis of intuitive, verbally given rules, which are guaranteed to be valid in the real world. The implicit robustness of a fuzzy controller allows the designer to ignore uncertainties in the plant and in the controller input and output. They are hence uniquely qualified for the control of systems with are not entirely known and are subject to chaotic, stochastical influences. The RoboCup is such a system.

3.3.2 The Takagi-Sugeno Controller

Another type of controller that makes use of fuzzy logic is the Takagi-Sugeno controller. It was developed by T. Takagi and M. Sugeno in the early eighties. Building upon Mamdanis proposition for a modular fuzzy controller, they introduced a new way of calculating crisp output values off of the fuzzy input states. Instead of performing the defuzzification as a kind of reverse fuzzification, the numerical output is directly derived from the rules, which contain explicit control functions as their consequences. A rulebase of a Takagi-Sugeno controller could for instance be

$$R_1 \; : \; \text{IF } \texttt{distance} \in \texttt{far} \text{ THEN } \texttt{shootstrength} = 5 + 6 \cdot \texttt{distance}$$
$$R_2 \; : \; \text{IF } \texttt{distance} \in \texttt{near} \text{ THEN } \texttt{shootstrength} = 27.5 + 1.5 \cdot \texttt{distance}$$

This rulebase approximates a smooth mapping along two linear grid lines as shown in figure 3.12. If there are multiple fuzzy states in the condition of a

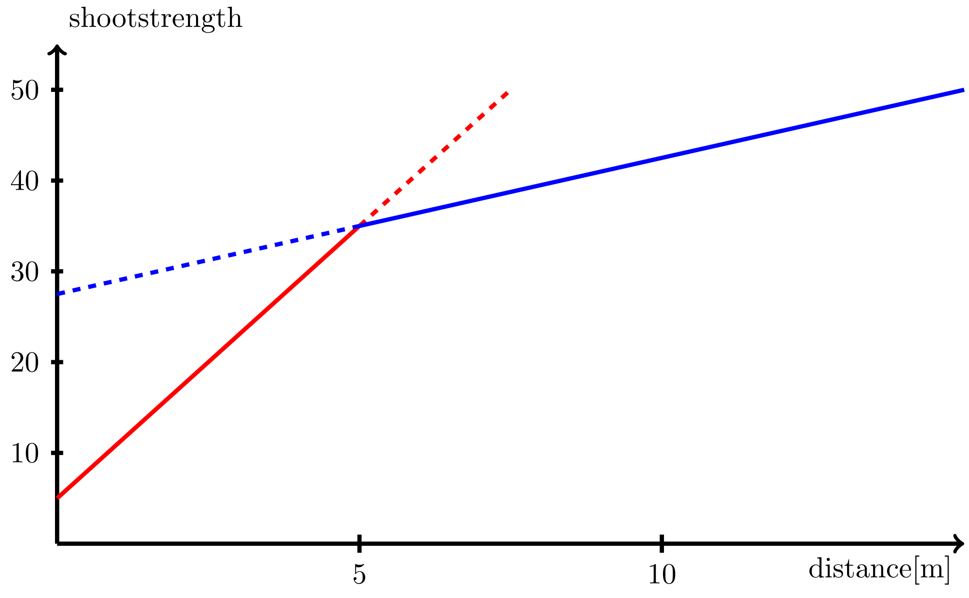

Figure 3.12: Graphical representation of the control laws provided by the rulebase of the example Takagi-Sugeno controller and the resulting mapping from input to output values.

rule, a t-norm has to be applied before the validity of the rule can be stated.

After that, every rule consequence is evaluated, i.e. the crisp output value according to that rule is calculated. The final output u then is the weighted average

$$u = \frac{\sum_{i=1}^{n} u_i \cdot \alpha_i}{\sum_{i=1}^{n} \alpha_i}$$

of those values u_i according to the truth value α_i of their conditions. In this simple example the fuzzy controller only does the simple task of switching between P-Controllers. Generally Takagi-Sugeno controllers place a greater focus on the input variables than on the output variables, which are at no point in the calculation representent as linguistic varables. In many cases this concept does not hurt the positive properties of the controller, but it simplifies it a lot. This observation is going to be important later on. In theory it is possible to construct one of these controllers using only crisp membership functions. In this case the controller would not perform very smoothly, but it could still work.

Chapter 4

An adaptive Fuzzy System

4.1 Design of the Fuzzy System

The properties of the plant and the desired properties of the controller have been outlined in section 2.1. The focus of this section is to introduce a possible controller that meats all these goals (robustness against uncertain input, robustness against uncertain plant, adaptivity and compatibiliy with existing infrastructure). The existing concepts for fuzzy controllers are a good starting point, because they already meet at least the robustness constraints without further changes. It is extremely tempting to be able to use a rulebase for the control of a RoboCup robot, as the desired behaviour of the players is often easy to grasp in words, but hard in code.
One difficulty in the design process of the controller is the great amount and the different nature of inputs and outputs. Most inputs and outputs are parameters with a varying range of values. Some are symbolic like the result of a shot or a pass and some are binary triggers or flags. Ideally all these inputs and outputs should be processed in the same system using the same framework.
The fuzzy controller devised in this thesis is a precalculated fuzzy controller. Precalculated means that the final act of finding the appropriate output values for the input values is not carried out by a fuzzy controller like the Mamdani or Takagi-Sugeno controller presented above, but instead by a simple mapping. Its exact form is calculated beforehand by evaluating a fuzzy controller for every possible input. The result is a controller that 'imitates' a fuzzy controller perfectly and possesses all of its strengths, but in addition is fast to execute.
A fast execution time is necessary because, compared to other controller con-

cepts, fuzzy controllers need a lot of computation time. Instead of solving a simple equation, a single computation of the output of a fuzzy controller requires the evaluation of multiple maps and several independent calculations, which in the case of the center-of-area defuzzification also includes a complex integral. In addition, the implementation of the inference is not trivial, and is hard to accomplish efficiently. Part of the requirement of compatibility is the ability of the controller to be continuously executed in the robots cycle of operation. This may be possible with an efficently implemented Mamdani or Takagi-Sugeno controller, but as more inputs and outputs are added to comprehensively steer the behaviour of the robot, this gets harder and may be impossible at one point.
The basic structure of the precomputed system is outlined in figure 4.1. The arrows that are pointed downwards represent the input that is going into all subsystems that are in a vertical line and not processed on the way.
This system maps a set of m inputs y_j to a set of n outputs $\triangle u_i$ using the

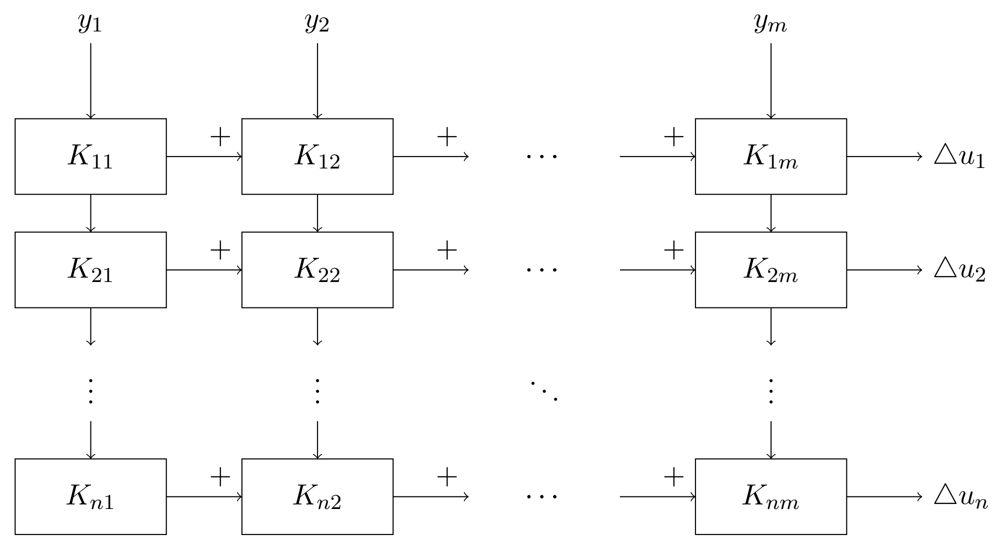

Figure 4.1: Layout of the precalculated fuzzy system. The inputs y_j come in from above and are calculated into the output values $\triangle u_i$ on the right.

control law
$$\triangle u_i = \sum_{j=1}^{m} K_{ij}\left(y_j\right) \tag{4.1}$$
for all $i = 1 \dots n$. This is a trivial control law and makes no direct use of fuzzy logic. In this form, the system is a simple MIMO controller (multiple inputs multiple outputs). They are reliable, fast and, for many control tasks, sufficiently performant. However, the initial configuration is hard when the internals of the plant are unknown and its impossible to use classical approaches like the Ziegler-Nichols method.
The precalculation combines the strengths of both controller types. This way, the main strength of fuzzy controllers, the intuitive design process, is made available to critical real-time system like the RoboCup.
The internals of the gain functions K_{ij} are given by the underlying fuzzy rules, but they can also be directly given if an exact mapping for the concerned input and output is known. Regardless of the origins of the content, K is always a map
$$K_{ij} : y_j \mapsto \triangle u_i. \tag{4.2}$$
Theoretically, K can take any form both from a precalculation and a manual configuration. It is sensible to limit the complexity though, an elaborate lookup would jeopardize the low computation time.
Once the system is properly constructed, it runs without using any form of fuzzy logic, as a simple controller which adjusts the output values and hopefully influences the robot to produce the desired results.
The construction of K by precalculating a fuzzy rule can be executed in multiple ways. The simplest version is to just evaluate the rules and ignore most of the information in the fuzzy maps used in the fuzzyification and defuzzification. A better method is to use at least the fuzzy maps of the linguistic output values to coarsly quantify the outputs. Finally, it is possible to construct a complete fuzzy controller and evaluate it for all possible outputs.
The first method only produces reasonable results when paired with an effective adaptation mechanism. As an example, the mapping from distance to shoot strength, which was used as an example troughout section 3, would in this simple method only be defined by the rules

$$\begin{aligned} R_1 &: \text{IF } \texttt{distance} \in \texttt{far} \text{ THEN } \texttt{shootstrength} \in \texttt{high}, \\ R_2 &: \text{IF } \texttt{distance} \in \texttt{near} \text{ THEN } \texttt{shootstrength} \in \texttt{low}. \end{aligned}$$

These simple rules could, disregarding all but the most trivial information in the fuzzy maps that actually quantifiy the linguistic values far, near, high

and low, be translated to a simple map K like the upper one in figure 4.2. While the exact numerical values can be ignored here, the relation between the linguistic values is actually important, i.e. the information that 'low' is smaller than 'high'. This problem can be circumvented by using the same linguistic values or every fuzzy map, like proposed in section 3.3.1.
In figure 4.2, the basic outline of the map clearly represents the rule, but the exact quantification, in the form of axis values is missing. This information is, in the case of the Mamdani controller, contained in the fuzzy maps used for fuzzyfication and defuzzification. In the case of the input, the values can be normalized in order to simplify the system. By mapping all input values to a range of for instance $0 \ldots 99$, not only do the functions K get simpler, but also an automatic association of the value to linguistic values like 'high' and 'low' becomes possible. In order to normalize the inputs, the possible range of the input variable has to be known. Then, the operation

$$y_j^\star = a + \frac{b-a}{y_{j_{max}} - y_{j_{min}}}, \tag{4.3}$$

where

- a is the start of the target range,
- b is the end of the target range,
- min is the start of the input range,
- max is the end of the input range,

normalizes the inputs.
In the example of the range $0 \ldots 99$ the equation simplifies to

$$y_j^\star = \frac{99}{y_{j_{max}} - y_{j_{min}}}. \tag{4.4}$$

In essence, the normalization means that the same fuzzy map is used for the fuzzyfication of all crisp input variables, but with different range of values. So this simple step combined with the standard linguistic values presented in section 3.3.1 replaces the entire fuzzification.
When using the simplest precalculation method, the numerical values corresponding to the output have to be manually introduced into the map. This means quantifying the entire y-axis either from experience and intuition or by just guessing. This can, when the relation between input and output is trivial, lead to satisfying results, but it is definetely better practice to use

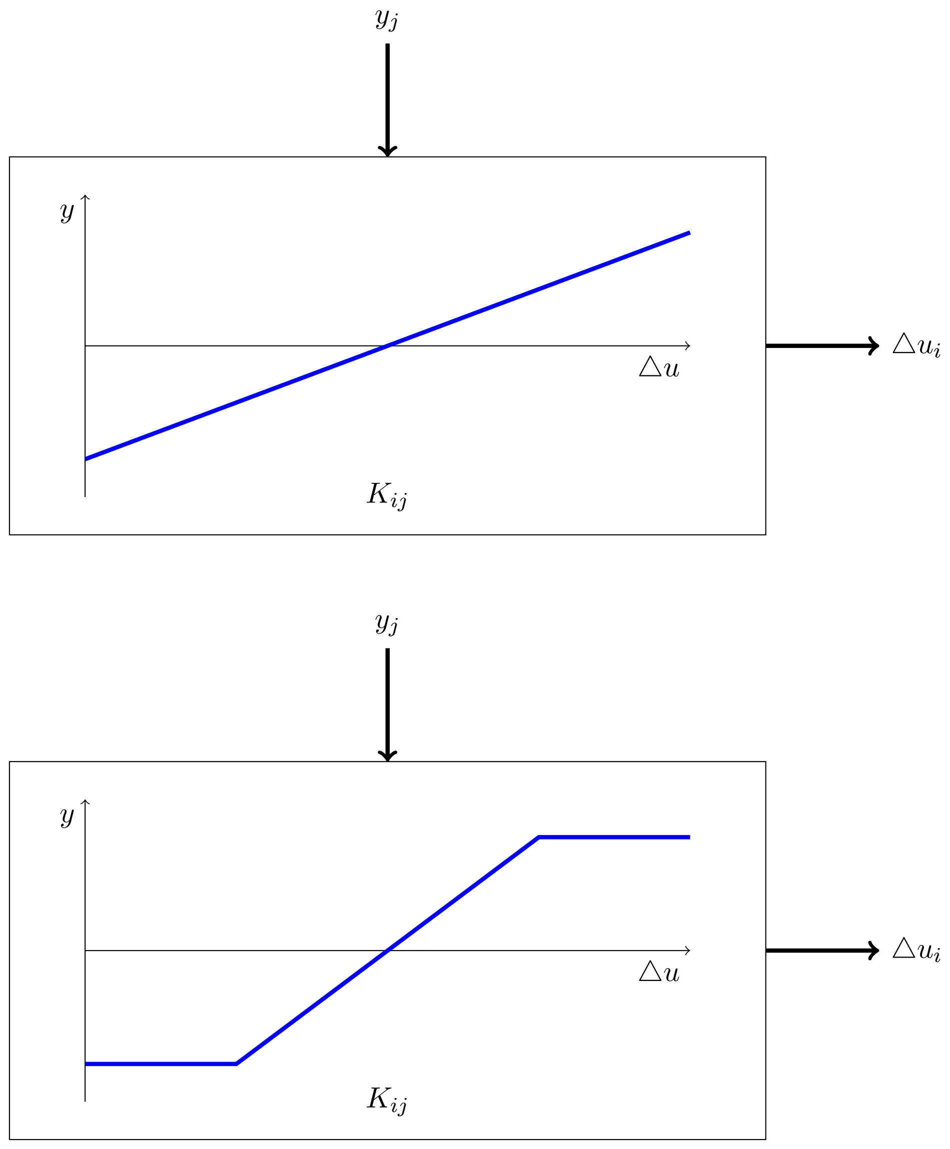

Figure 4.2: Example precalculated maps K for the input variable 'distance' and the output variable 'shootstrength'. Both are reasonable approximations to the exact mapping of a Mamdani controller, but mostly the extra precision of the lower map is not necessary for this specific problem.

the second method and use more of the information contained in the fuzzy controller to construct the precalculated system.
However, carrying over the information embedded in the defuzzification map is not straightforward. An efficient method is to define simple, constant defuzzification maps, practically assigning each linguistic value a specific numerical value. These maps are usually called *Singleton Fuzzy Maps*. The values that are given for 'high' and 'low' can be used to specify the endpoints 0 and 99 of the normalized inputs, while the values in between are defined by the rulebase. A precalculated system constructed using this way can produce meaningful results without further adaptation. It is however possible and sensible to optimize the configuration using the adaptation algorithm, to at least to react to later changes in the plant or other unforseen influences.
Finally, the values on the y-axis could also be exactly calculated by evaluating a complete Mamdani controller for every input value and storing every output value in K. The second map in figure 4.2 is the result of such an exact calculation. It is obvious that the difference to the other two methods is not overwhelmingly significant, but only if the output values are correct. This third method guarantees an accurate mapping that is identical to that of the original Mamdani controller. This brute-force method definitely works, but it greatly increases the computation time that is necessary for the initial, offline setup. As the controller is dynamically adapted anyway, it is more economical to approximate the map using the second method and leave the exact fitting to the adaptation mechanism.
As mentioned in section 2.1, the controller that is devised in this thesis acts as a corrector to the original mechanism that are already implemented in the robots software. This means that the influence of an input can easily be disabled for an output by setting the map K to zero. As the number of inputs and outputs grows, this will be the case for most maps. When there is an influence, it is added up to all other influences and added up to the original value like shown in figure 2.5.
The set of maps that is responsible for one output is called the *rule* of that output. These rules are not equivalent to, but correspond to the rules in the rulebase of a Mamdani controller like presented in section 3.3. There are m maps in one rule, and n rules form a complete system. Therefore, there are $n \cdot m$ maps in the system.
This system architecture has all the advantages of a simple mapping. It is fast and reliable, and it is possible to directly influence the mappings. On the other hand, the design process benefits from the advantages of fuzzy controllers, namely an intuitive setup and the ability to view the control

laws as linguistic rules even after the final controller is in place.
The design process is twofold: At first the system can entirely be designed as a fuzzy controller, by defining the fuzzification maps for every input variable and defuzzification maps for every output variable and stating the rules. In the second step, the maps K_{ij} are defined by precalculation involving only the maps, or further information in the fuzzy maps. After the precalculation it is possible to directly influence the maps, for example to deactivate certain influences or to implement a more exact control law when it is possible.
The final precalculated system sould further be optimized by the adaptation mechanism during operation. This will be covered in section 4.3.

4.2 Implementation of the System

The system was implemented in `C++` and integrated into the robots control system. It consists of three parts that are largely independent from one another. There is the controller itself, which contains only the maps K_{ij} to calculate the output for a given input. Then there is the module that constructs the controller by precalculating a fuzzy controller which uses linguistic rules. At last, there is the adaptation mechanism that optimizes the controller whenever there is an input which allows to evaluate the perfomance of the controller. The adaptation is covered in section 4.3. Figure 4.3 shows the relation of the systems in operation.

During runtime, only the controller itself is active and maps crisp inputs to crisp output values. During the initialization, the rulebase is evaluated in the construction module and the final system is precalculated. All three modules are handbuilt, because while there are libraries for fuzzy controllers like [Rab12] available and some commercial engineering software like Matlab and Modelica support fuzzy controllers, none of them has all the desired features and can be easily integrated in the robots software.

As mentioned, the mapping systems main focus is on execution speed. The configuration of the system is taken care of by the construction module, so no compromise has to be made for user friendlyness. There are four components in the mapping system which are realized as classes. The whole system with m inputs and n outputs is called *fuzzy system.* One line in that system, which also has m inputs, but only 1 output is called a *fuzzy rule*, although it can contain more than one of the rules in an original Mamdani controller. It contains in fact all the information on the calculation of one output variable. A rule is made up of m *fuzzy maps*, which contain the mapping function K and map one input to one output. Finally, a fuzzy map contains one or more *fuzzy elements*, which are linear pieces that together make up the mapping function. There is no real reason to call these 'fuzzy' other than consistency, although their coarse grained nature reflects the fact that the exact mapping is less important than a representation of the underlying rule.

A fuzzy element is just a linear piece of a function that covers a certain range of the mapping. They are stored in a list in the fuzzy map class. The fuzzy element class contains the field variables

- `bool isZero,`
- `bool isConstant,`

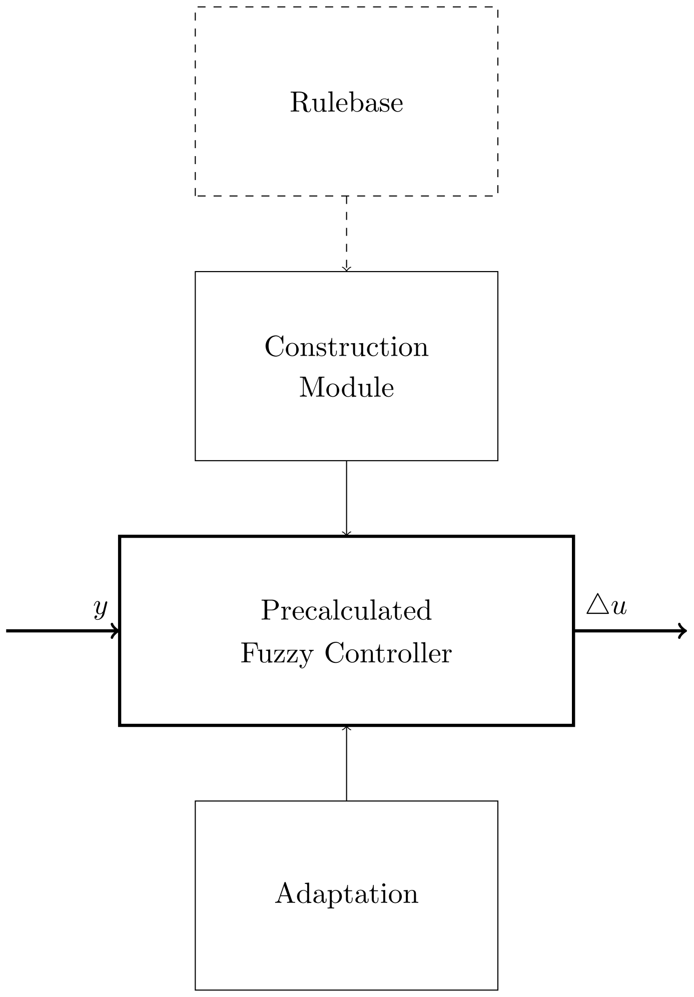

Figure 4.3: Structure of the implemented system. The precalculated fuzzy controller that map the inputs to the outputs is configured and built by the construction module from the rulebase, which is a simple textfile. During operation, the system is optimized by the adaption mechanism.

- `int start,`
- `int end`
- `double startValue,`
- `double endValue,`
- `float gradient.`

When the maps evaluation function is called with an input, the list of elements is linearly searched for the element where

`start < input < end.`

Then, the elements `isZero` and `isConstant` flags are checked. If `isZero` is true, the return value 0 is immediatly passed as the output. If `isConstant` is true, `startValue` is returned, as the element has the same value for every input. If neither of the flags is set, the element represents a linear gradient in the map. In that case, the output value is calculated using the simple formula

$$\texttt{output} = \texttt{startValue} + \texttt{gradient} \cdot (\texttt{input} - \texttt{start})\,.$$

The gradient is precomputed during the initialization of the map and it is updated along with the flags when the element gets changed. The ordering of the elements in the list of the map can be changed to move elements which are frequently used to the front. The frequency of use of the elements can be stored to determine the optimal order of elements. All these small runtime optimizations add up to lower the computation time of the system, they are however not essential to the functionality.
The number of fuzzy elements in a map differs from map to map. A map can contain only one element whose `isZero`-flag is set to represent a deactivation of an influence of the corresponding intput on the output. When the map is finely grained, i.e. contains many elements, they can actually overlap in `start` and `end`-values. In this case, the element which comes first in the list is chosen. Also, the elements do not necessarily need to continuously cover the input range, although the construction module always assures this. In theory, there can be gaps between elements. As a fallback, there is always an element which covers all the input range and the end of the list. It is usually set to zero.
When called, the fuzzy system hands down the given inputs down to the rules, which distributes the inputs to the maps. The results of the maps,

which can be seen as the isolated influences of the corresponding input y_j to the rule output $\triangle u_i$, are added up and form the rule output. Finally these are put together by the system and give the output $\triangle u$.
Figure 4.4 show the structure and the flow of information in the fuzzy sys-

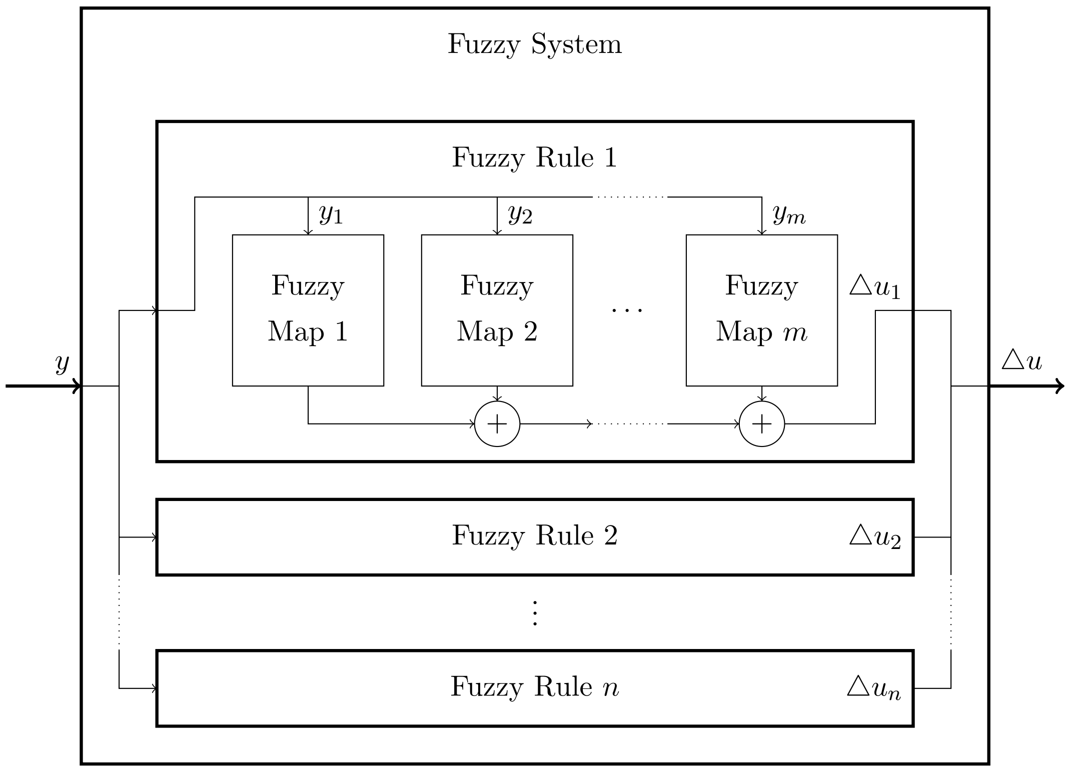

Figure 4.4: Information flow in the precalculated fuzzy system. The vector input y is fed into the rules, where it is split up and deilvered to the corresponding rule. The output of the maps is added up and forms the rule output. Put together to again form a vector, they represent the output of the system.

tem, rules, and maps.
The rules in the system and the maps in a rule are stored in vectors, because their ordering is clear and not changed during the operation.

This implementation of the precalculated mapping system is certainly not the simplest and most straightforward, but it is very fast to execute and is prepared to be constructed from linguistic rules. The construction process is seperated from the precalcuated system and essentially forms an own executable program. The input into this program is a text file which contains the rulebase, where all the rules take the form

$$\text{IF } y_j = \texttt{a} \text{ THEN } \triangle u_i = \texttt{b},$$

where

$$\begin{aligned} \texttt{a} &\in \{\texttt{low, medium, high}\} \\ \texttt{b} &\in \{\texttt{lower,medium,higher}\}. \end{aligned}$$

As the precalculated fuzzy system acts as a corrector, the indications as `higher` and `lower` seem more appropriate. The value `medium` will in this case mostly be 0, as it represents that no change to the original parameter choice is needed.

Of course for a more precise control the number of linguistic values could easily be increased, but the restriction on three values already allows a meaningful rulebase and therefore effective control.

The output of the construction module is a configuration file which can be read by the robots software and is used to initialize the final fuzzy system. These files, which contain all the information about the system and can be manually be edited, are also used to save the system. So, complementing the structure shown in figure 4.3, the real stream of information is illustrated in figure 4.5.

The precalculation of the fuzzy system is performed using the second of three methods presented in section 4.1. The form of the maps is initialized using only the information in the rules, but the exact values of the maps y-axis are set by evaluating singleton defuzzification maps which essentially assign each one of the linguistic values a constant numerical value. Each rule is translated to a point of the map, and these points are connected by elements. The restriction to only three different linguistic values restricts the number of elements to two. This may seem overly limiting, but it is enough to desribe most dependencies in the fuzzy system.

The rules

$$\begin{aligned} R_1 &: \text{ IF } \texttt{distance} = \texttt{high} \text{ THEN } \texttt{shootstrength} = \texttt{higher}, \\ R_2 &: \text{ IF } \texttt{distance} = \texttt{low} \text{ THEN } \texttt{shootstrength} = \texttt{lower}, \end{aligned}$$

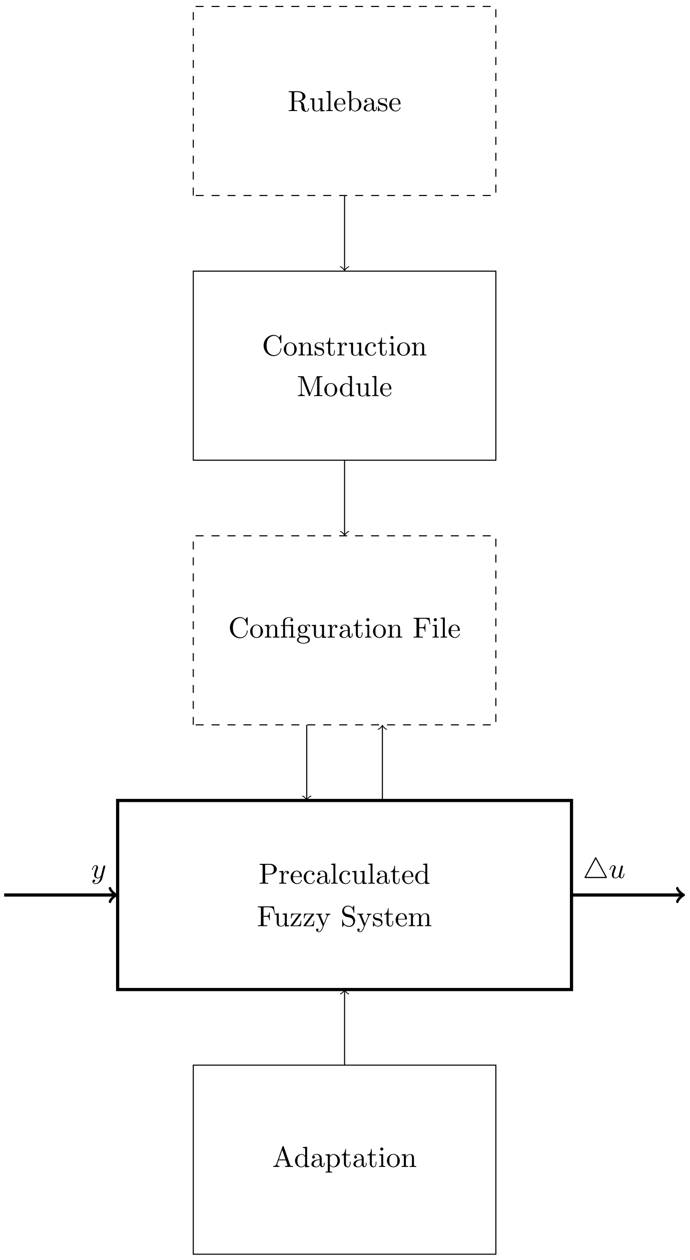

Figure 4.5: Structure of the implemented system including communication via configuration files. The configuration files are created by the construction module using the rulebase, and read by the constructor of the Fuzzy System class to initialize all the rules and maps and connect them to the right inputs and outputs. It is also possible to store the current configuration of the precalculated fuzzy system including the conducted optimizations back to the configuration file.

for instance define the relationship between 'distance' and 'shootstrength'. To translate this into a map, the linguistic values `higher`, `medium` and `lower` of both linguistic variables need to be linked to numerical ones. This can happen by simply listing them as additional information in the rulebase or a seperate database called the *assignment map*. It is technically not part of the rulebase, but carries the information normally contained in the defuzzification maps. For reasons of convenience it is stored in the same file. Since the shootstrength takes a value between 0 and 50, the assignment map could in this case be

```
shootstrength :  lower = -10, medium = 0, higher = +10.
```

Remember that the fuzzy system acts as a corrector to the rest of the robots system and therefore does not directly define the shootstrength by oneself.
The configuration file for the fuzzy system contains the points of the map that are defined by the rules. In this case, these points are $(0, -10)$ and $(99, +10)$. They are scanned during the initialization of the fuzzy system and the elements are fitted, in this case only one. The resulting map is shown in figure 4.6.
The precalculation was performed with the goal of faster execution speeds of for a controller with a high number of inputs and outputs. The runtimes of a precalculated fuzzy system of this particular form with a varying number of inputs and outputs is shown in table 4.1.
The runtime was recorded using the built-in timing functions of ctime.h of the C standard library. Up to a fuzzy system with nearly 1000 inputs and 1000 outputs, the runtime is sufficiently low to be used in the robots software. The runtime of 78 seconds for a system with 10000 outputs and 1000 inputs isnot caused by a sudden peak in problem complexity, but by the limited amout of memory and the subsequent swapping. The `clock()` function, which measures only the CPU cycles, shows a pure calculation time of 1.96s, which fits in with the rest of the times. The fuzzy system with 10000 inputs and outputs overstrains the system and did not finish.
The precision of this method, around 15ms, is far from perfect, but it still allows qualitative judgement about the real-time capabilities of the system.
As a result of the calculation time measurements, it can be said that the system has low enough computation times for all but very extensive systems with thousands of inputs and outputs.
Figure 4.7 shows the runtime of systems with an equal number of inputs and outputs. In the RoboCup, the maximum time the system should take to calculated the output value is 30ms, the cycle time. The simulations, which were conducted using a random construction with 5 elements per

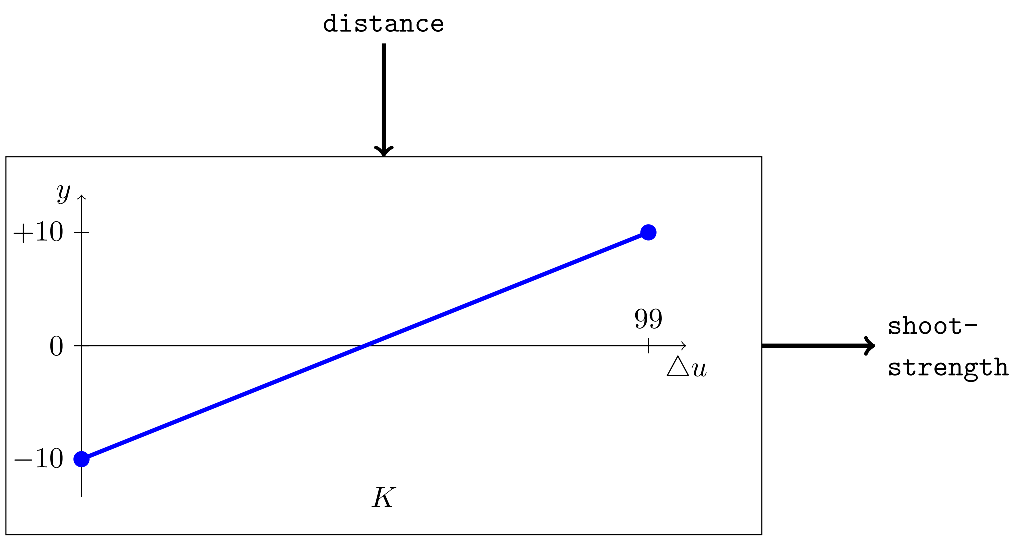

Figure 4.6: Fuzzy Map constructed from given rulebase and assignment map, containing one element, which contains the entire relationship between the intput variable 'distance' and the output variable 'shootstrength'.

i/o	1	10	100	1000	10000
1	0	0	0	0	0
10	0	0	0	0	0.02
100	0	0	0	0.02	0.12
1000	0	0	0.01	0.11	78
10000	0.01	0.01	0.01	1.12	dnf

Table 4.1: Runtimes of the precalculated fuzzy system with a varying number of outputs in seconds. From left to right, the number of outputs in logarithmic scale. From top to to bottom, the number of inputs. Runtime was measured using the built-in functions of `ctime.h`. The precision is roughly 15ms, the measurements were conducted on an Intel quadcore machine with 2.6 Ghz CPU clock frequency. A 0 signifies a runtime below the minimal detection range. For the numbers of inputs and outputs that occur in the robocup, the runtime is sufficient, even for a 1000x1000 system with 1 million maps the computation time is manageable.

map, show that up to a system with 500 inputs and outputs (that means 250000 maps), the runtime stays under that limit.

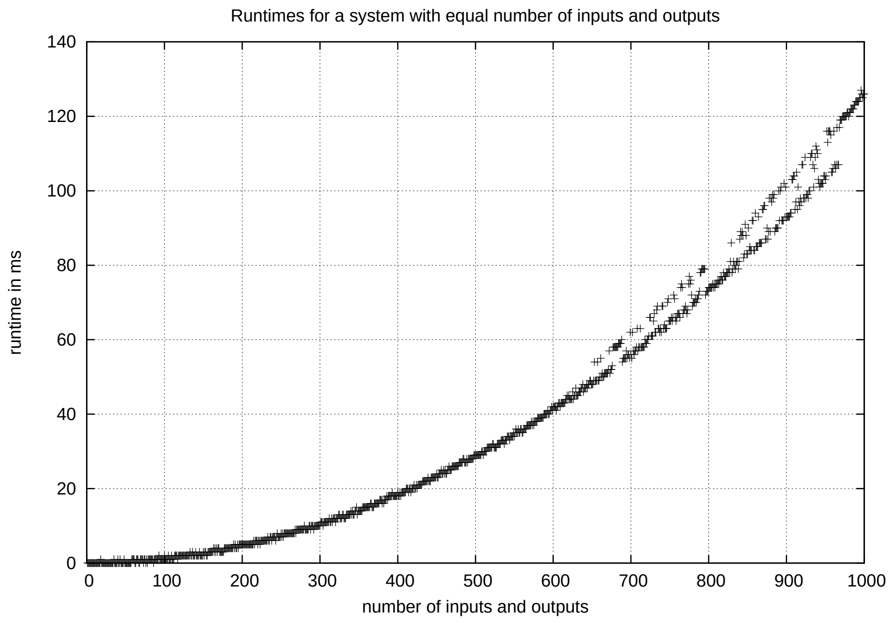

Figure 4.7: Runtimes of the pre-calculated fuzzy system with an equal number of inputs and outputs

4.3 The Adaption Mechanism

After a proper setup, the precalculated fuzzy system should be able to influence the robots behaviour in the desired way and improve its overall performance. However, in order to further enhance the behaviour, adapt the robot to different environments (like different opponents) and compensate initialization errors, it is necessary to include a mechanism that can change the system during operation. The three mentioned tasks of this mechanism are not equivalent. The enhancement of the behaviour that is given by the rules is an optimization task on the parameters of the system. Due to a changing environment, the goal of this optimization is not always the same, i.e. there is no perfect parameter set. If the adaptation mechanism also should be able to correct mistakes during the setup like an incomplete rule base, it needs additional capabilities that exceed those of simple parameter optimization. For simpler systems with manageable complexity it is sufficient to do without this complex task and leave it to a human operator. The parameter optimization and adaptation is however necessary to improve the system by automatically sorting out the exact values of the fuzzy maps rather than the underlying rules.
Mathematically, an optimization is the search for the element in a set O that maximizes or minimizes a target function

$$T : O \mapsto R,$$

with $R \in \mathbb{R}$ being the performance index that is sought to be maximized or minimized. O is also called the *optimization space* and contains all possible solutions to the problem to be solved optimally. The two cases minimizing and maximizing are interchangeable by inverting the target function. In the case of the optimization of the fuzzy system, the optimization space 0 contains all possible parameters that define the individual maps.
The choice of the target function T is, other than in most optimization problems, not clear and unambiguous. Somehow the overall performance of the robot has to be grasped. Since it is not easily possible even for a human to say if the robot is playin well or badly at any given time, the target function has to use information that allows to clearly rate the robots success. This is possible for example by evaluating the referee events during the game. Whenever the game is interrupted, the referee sends a broadcast message to every player announcing the decision. When a robot shoots on the goal the possible outcomes are:

- goal,

- ball deflected and still in the game,
- miss, followed by a goal-kick when the ball is not deflected or, less probable, deflected by an own player,
- miss, followed by a corner when the ball is deflected by an opponent.

Each one of these situations can be clearly determined by the referee event that follows. A goal is directly broadcasted as such. If the ball is deflected and the game continues, there is no referee event, so if there is none after a certain waiting period, this case has to be assumed. The referee events 'goal-kick" and 'corner-kick' allow, besides the conclusion that the ball missed, also insight on whether the ball missed right or left, by the side of the goal-kick or corner-kick. If the ball misses the goal without touching any player, so there is a goal-kick, the robots parameter set is clearly not ideal, so this is the worst outcome. If the ball is deflected, no real conclusion on the performance can be drawn.
In order to be used as a performance value, the referee events have to be mapped to a real value. A possible rating is presented in table 4.2.
The mapping is the task of the target function. It would however be not

Event	**Rating**
goal	10
no interruption	5
goal-kick	0
corner-kick	5

Table 4.2: Rating of referee events

reasonable to to use one event as the single indicator of the performance of the robot. For only one shot on the goal for instance there are too many random influences on the outcome to assume that it is only caused by the chosen parameter set. A solution is to sum up the last for instance 10 events. The performance of the robot is then measured over a longer period, and a better performance is a stronger indicator of a superior parameter set.
To perform the optimization, the parameter set is changed, then the target function is evaluated, and based upon the result the new parameter set is kept or not. This is the basic procedure in every optimization. The steps are once again illustrated in figure 4.8.
Virtually every optimization algorithm operates in this way. The most im-

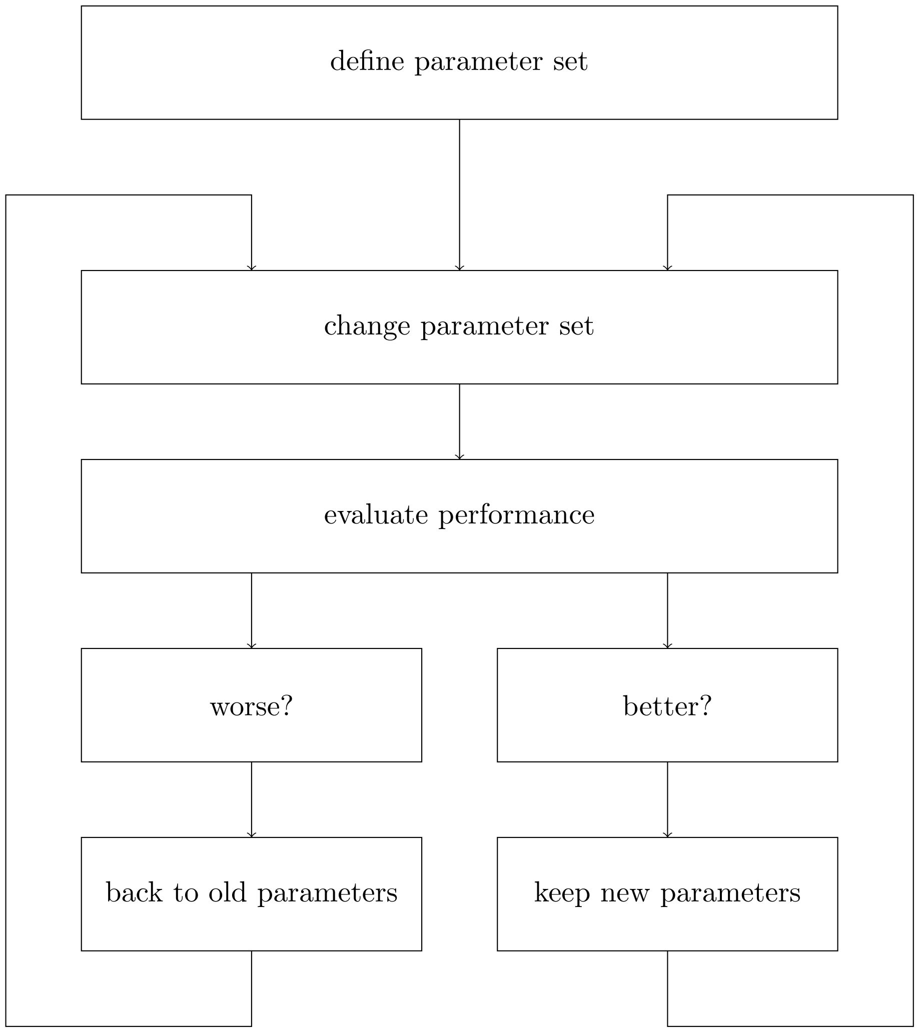

Figure 4.8: Basic structure of a standard parameter optimization process

portant difference between the algorithms concerns the change of parameters. Probably the most basic optimization method is *hillclimbing.* It changes the parameter set by incrementing or decreasing only one parameter at a time by only a limited amount, thus performing a search in the neighbourhood of the currrent parameter set. Exhaustive hillclimbing would mean checking the whole neighbourhood for the optimal parameter set before committing to the found set and reapeating from there. Greedy hillclimbing on the other hand implies that a set which is better than the original one is instantly chosen as the new reference. The choice depends highly on how easily and quickly the target function can be evaluated. If the target function is a mathematical function that does not require much computation time, exhaustive hillclimbing is the better choice, because it makes bigger steps towards the optimal solution. In the case of the RoboCup however, the target function can not be easily and quickly evaluated. It requires in fact a certain amount of playing time to come to a meaningful performance index. Exhaustive hillclimbing would mean that the entire neighbourhood of the current parameter set is tested and rated. Greedy hillclimbing is definetely the better choice here, as it allows small, but steady steps towards a better performance.

The way the parameter set is composed and the values are changed is an important decision. The parameter set that defines a fuzzy system can be defined in several ways. One intuitive way is to use the elements that are defined during the construction process of the system. The map in figure 4.9 for instance is defined in its entirety by the one element.

Using the field variables of its class used in section 4.2, it could be described by the quadruple

$$E = \left(0, 0, \left(0, -10\right), \left(99, 10\right)\right).$$

The alteration of values in the first step of the optimization could change any of these values. The most influencial change can probably be expected when the ouput values are adapted. This could for instance change the element to

$$E_{alt_1} = \left(0, 0, \left(0, -5\right), \left(99, 5\right)\right),$$

which would yield the map in figure 4.10.

This adaptation is able to correct mistakes in the construction of the system, especially when the defuzzification values are not precise enough. By adapting the base points of the element, the map can be structurally changed. The element

$$E_{alt_2} = \left(0, 0, \left(20, -10\right), \left(99, 10\right)\right)$$

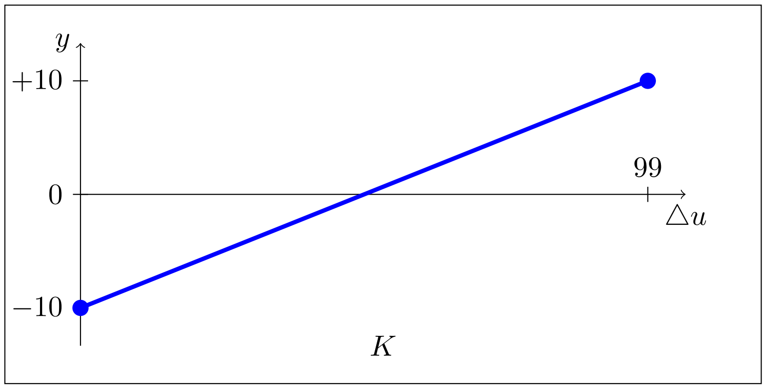

Figure 4.9: Original Fuzzy Map constructed from given rulebase and assignment map in section 4.2

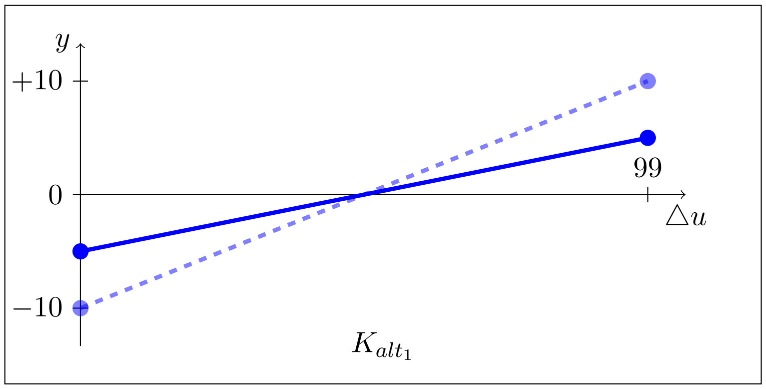

Figure 4.10: Altered Fuzzy Map, representing element E_{alt_1}

for example redefines the map in a way that another element

$$E_{alt_2}^{supp} = (0, 1, (0, -10), (20, -10)),$$

has to be introduced by elongating the left base point to 0. The generated map 4.11 shows a behaviour that differs from the one that has been defined in the corresponding rule. It is of course also possible to adapt the two flags

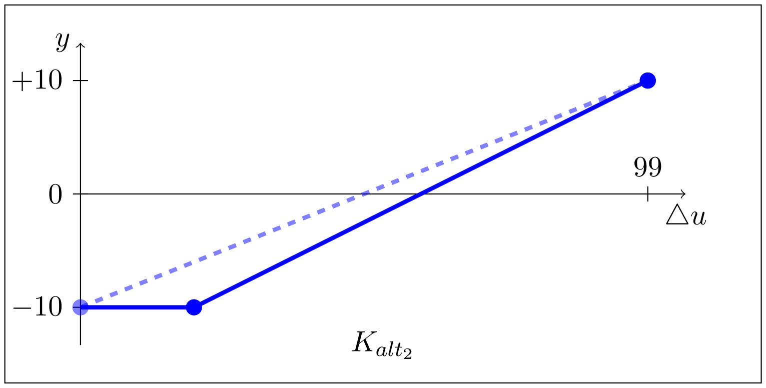

Figure 4.11: Altered Fuzzy Map representing element E_{alt_2}

of the element that define constant and zero behaviour, respectively. The results would be in the case of the constant element

$$E_{alt_3} = (0, 1, (0, -10), (99, -10)),$$

or

$$E_{alt_3} = (0, 1, (0, 10), (99, 10)),$$

depending on which base point is utilized as a constant value. This leads to the map 4.12.
Setting the whole element to zero is certainly a drastical adaptation, but it is still reasonable to test if the influence the map represents is still viable. The element

$$E_{alt_4} = (1, 1, (0, 0), (99, 0))$$

deactivates the map and with it any influence of the correponding input to the output the rule the map 4.13 belongs to.

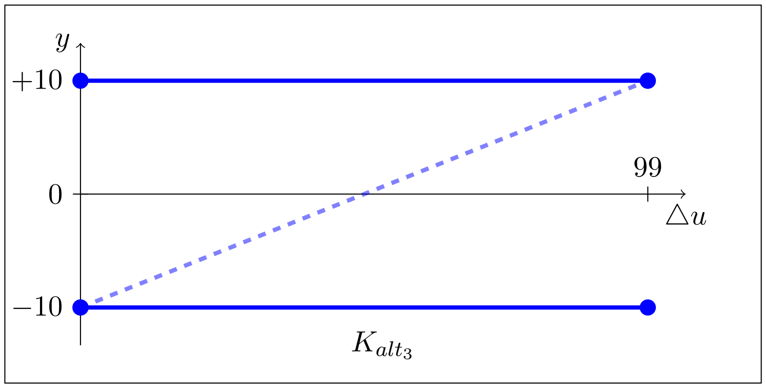

Figure 4.12: Altered Fuzzy Map, representing element E_{alt_3}

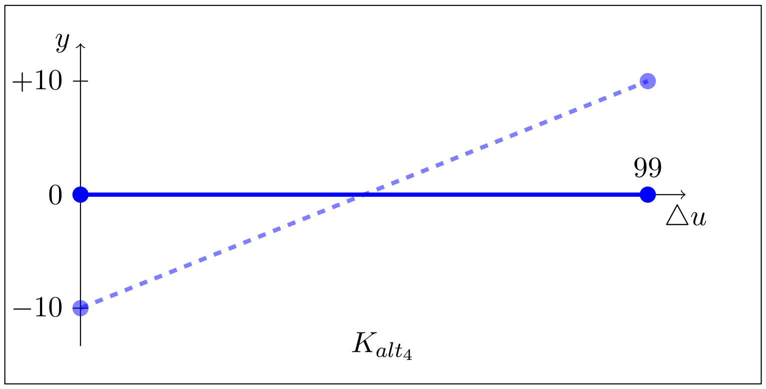

Figure 4.13: Altered Fuzzy Map, representing element E_{alt_4}

These are the adaptions that are available to change an element of the fuzzy system and observe the results. In order to have an influence on the optimization process, a *certainty value* $c \in [0, 100]$ is introduced for every map. It can be used save certain maps from extensive change when there is no doubt on the exact configuration of the map. A certainty value of $c = 100$ would mean that the map is changed under no cirumstances, while a value of 0 leaves the map subject to comprehensive adaptation.
A possible implementation of this mechanism is presented in algorithm 1. The given values $10, 30, 60$ and 100 translate into a probability of $10, 20, 30$

Algorithm 1 Adaptation Algorithm

```
$r \leftarrow rand(100)$
**if** $(r + c) \leq 10$ **then**
  adapt_isZero()
**else if** $(r + c) \leq 30$ **then**
  adapt_isConstant()
**else if** $(r + c) \leq 60$ **then**
  adapt_basePoints(c)
**else if** $(r + c) \leq 100$ **then**
  adapt_values(c)
**end if**
```

and 40%, respectively for the given operation and a certainty value of 0. As the certainty value increases towards 100, the probability of grave change to the elements decreases. A certainty value of $c \geq 10$ inhibits any change to the parameter `isZero`, and so on until a certainty value $c = 100$, which allows no change at all to the map.
It is possible to set the certainty values of all maps manually, but it is also possible to calculate an appropriate certainty value using the information provided. When there is are several rule and defuzzification information about a particular map, the certainty about its exact shape is usually high, but when there is only one rule, or no rule at all, the certainty is usually low. This mechanism was not implemented, it represents a possibility to further automate the optimization process.
To meat the goal of adaptation to changing environments, it is necessary to leave at least some room for optimization even after a seemingly ideal setup has been found. It is for instance possible to continually decrease the certainty value over time, so that every map is at least challenged once in a while. Inversely, the certainty value can increase when the optimization algorihtm does not find better configurations.

It is questionable whether this optimization method ever leads to a converging behaviour. It is also unclear whether a converging behaviour is necessary at all. Probably the adaptation is more alternating and dynamic, but while this might be undesirable in many circumstances, in the RoboCup this constantly changing behaviour of the robot is in most cases not obstructive and might even fool and interfere with an opponent who is trying to adapt itself.

Chapter 5

Special Problem: Learning to shoot

5.1 Properties of the Problem

Instead of immediately tackling the challenge of controlling the whole behaviour, it is useful to take a look at a more specific scenario. A big problem of the task to devise a controller for the behaviour in a RoboCup game is the lack of testing possibilities. Methods which require many test samples of real-world application are hard to apply in the RoboCup, because these samples can only be collected during real games. These games are not conducted very regularly though, and so it helpful to find a simpler scenario for testing, which makes it possible to find the necessary samples and enable a visible optimization process of the controller.
The scenario that is chosen to gain first experiences with the problem at to test the efficiency of the control algorithm is that of a simple shot on the goal. Basically, the ball is set to a random position and the robot, alone on the field, searches it, dribbles it towards the enemy goal and shoots. The feedback given after the shot comes from the referee and consists of the decision a referee would also communicate in a real, human game. As mentioned in chapter 1.2, during official RoboCup matches, the referee is a human sitting next to the field who can broadcast events like 'goal' and 'corner kick' to all the players. In this test scenario, the referee events also have to be manually provided. They are the only way to evaluate the performance of the controller.
This problem has a highly reduced complexity from what the system was originally designed for. However, a hopefully positive influence on the shoot-

ing performance can be expected. The learning capabilities could be useful to adapt to certain types of ground and different balls which maybe behave differently.
The system that is constructed for this test is outlined in figure 5.1 As inputs and outputs, the controller uses the values in table 5.1. The inputs are evaluated directly before the shot and are fed into the controller. The output are then added to the statically computed values as shown in figure 2.5.
By defining all inputs and outputs and setting up the fuzzy system all pre-

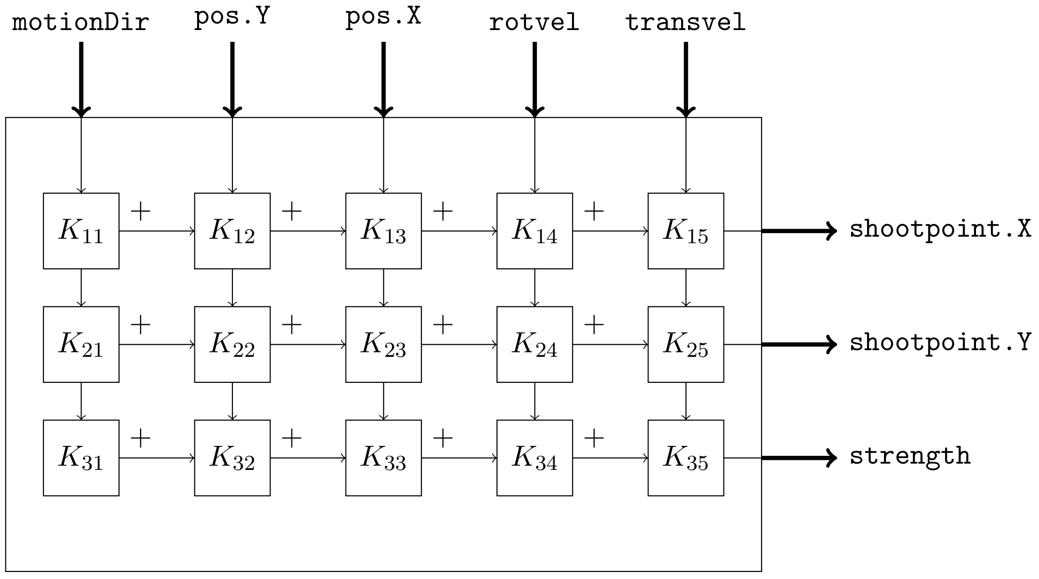

Figure 5.1: Fuzzy system for the special problem. The five inputs are mapped to three outputs using 15 individual maps which are constructed using fuzzy rules.

requisites for the construction of the system have been met.

variable name	**range**	**description**
`motionDir`	$-180 \ldots 180$	direction in the robot is facing
`pos.X`	$-6000 \ldots 6000$	vertical position in mm
`pos.Y`	$-9000 \ldots 9000$	horizontal position in mm
`rotvel`	$-180 \ldots 180$	rotation velocity
`transvel`	$-500 \ldots 500$	translation velocity
`shootpoint.X`	$-9000 \ldots 9000$	aim point on x-axis
`shootpoint.Y`	$-6000 \ldots 6000$	aim point on y-axis
`strength`	$0 \ldots 50$	shoot strength

Table 5.1: List of inputs and outputs for the reduced, special problem with value ranges.

5.2 Constructing the Special System

The first step in the construction of the fuzzy system for this special problem is the formulation of the rulebase. This is where human knowledge and experience comes in. Analysing the inputs and outputs one by one and identifying possible depedencies and influences is the most systematic way to construct the rulebase.
The first input, the direction of movement `motionDir`, has no clear influence on any of the outputs. The corresponding maps will stay empty at first, but they may be changed by the adaptation mechanism later.
The input `pos.Y` has a clearer influence at least on `strength`. One can generally say that the farther away the robot is from the goal, the stronger the shot should be. This influence is the same for `pos.X`.
The rotation velocity of the robot is a major influence on the output `shootpoint.Y`. If the robot rotates to the left, the `shootpoint.Y` has to be further right because the shot will veer off to the left.
Finally, the translation velocity `transvel` should have an influence on the strength again. If the robot is travelling at great speed, the shoot does not need to be as powerful.
These influence can be translated to the following rulebase:
Unfortunately, this rulebase makes no statement on most of the maps. As

R_1 :	IF	`pos.Y`	=	`high`	THEN	`shootstrength`	=	`higher`
R_2 :	IF	`pos.Y`	=	`low`	THEN	`shootstrength`	=	`higher`
R_3 :	IF	`pos.Y`	=	`average`	THEN	`shootstrength`	=	`lower`
R_4 :	IF	`pos.X`	=	`high`	THEN	`shootstrength`	=	`higher`
R_5 :	IF	`pos.X`	=	`low`	THEN	`shootstrength`	=	`lower`
R_6 :	IF	`rotvel`	=	`low`	THEN	`shootpoint.Y`	=	`higher`
R_7 :	IF	`rotvel`	=	`high`	THEN	`shootpoint.Y`	=	`lower`
R_8 :	IF	`transvel`	=	`high`	THEN	`shootstrength`	=	`lower`
R_9 :	IF	`transvel`	=	`low`	THEN	`shootstrength`	=	`higher`

Table 5.2: Rulebase for the special problem

every input is connected to every input, this is no worry, and this will hap-

pen even more so when the number of inputs and outputs increases. When there is no known relation between the input and output, the map should be left blank. The adaptation mechanism might find an influence later during operation.
The formulated rules allow the construction of the maps K_{32}, K_{33}, K_{24} and K_{35}. To finally define these maps, the defuzzification values for the outputs have to be defined. That is all the information that has to be provided by

name of output	**ling. value**	**num. value**
shootpoint.X	higher	+500
shootpoint.X	lower	-500
shootpoint.Y	higher	+200
shootpoint.Y	lower	-200
shootstrength	higher	+10
shootstrength	lower	-10

Table 5.3: Defuzzification info for the outputs of the special system

a human. The resulting maps are shown in figures 5.2, 5.3, 5.4 and 5.5.

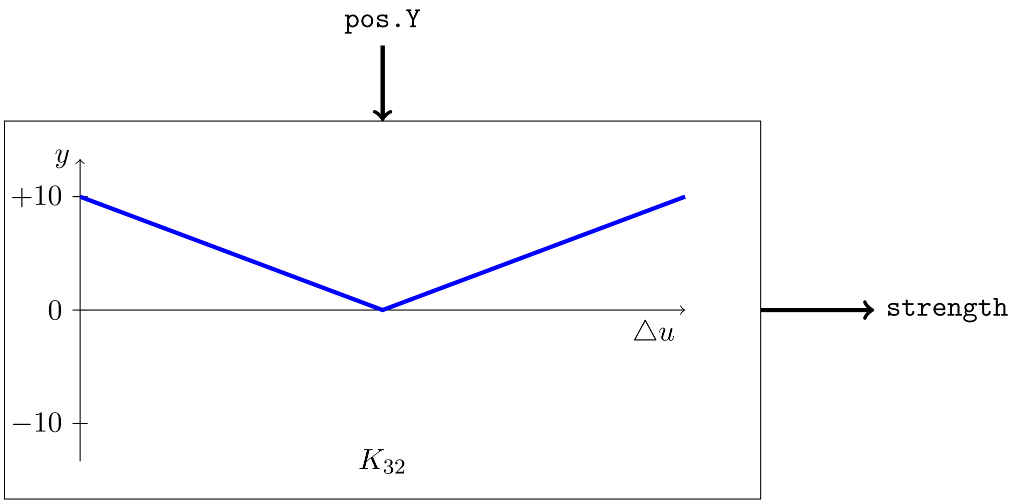

Figure 5.2: Initial configuration of the map K_{32}

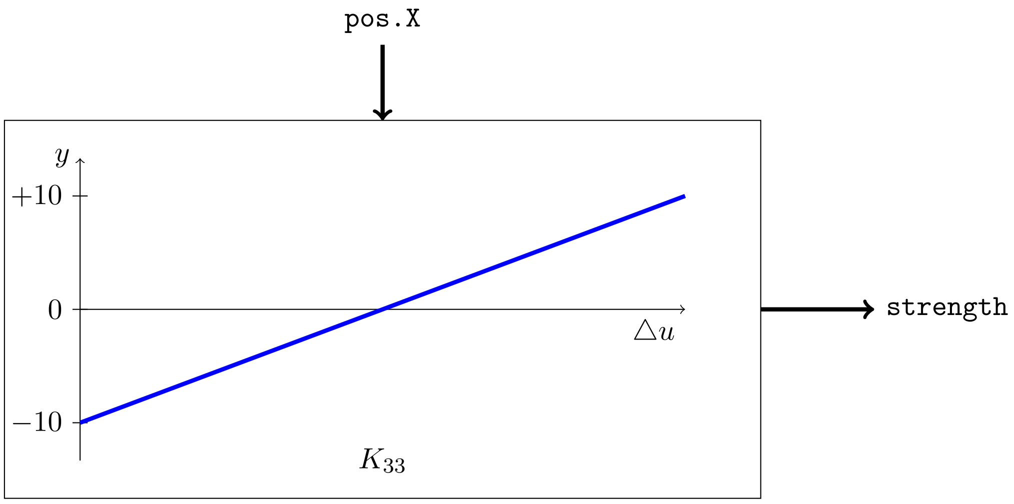

Figure 5.3: Initial configuration of the map K_{33}

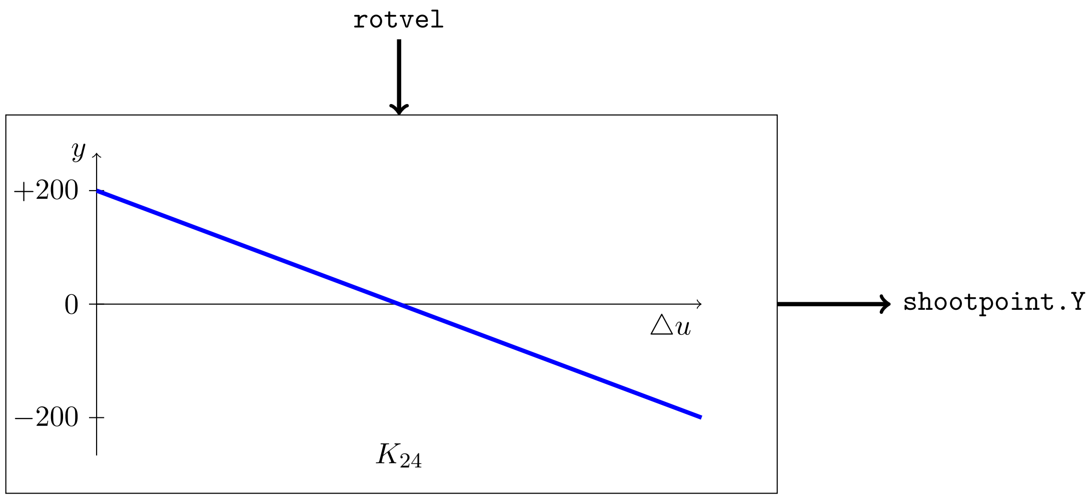

Figure 5.4: Initial configuration of the map K_{24}

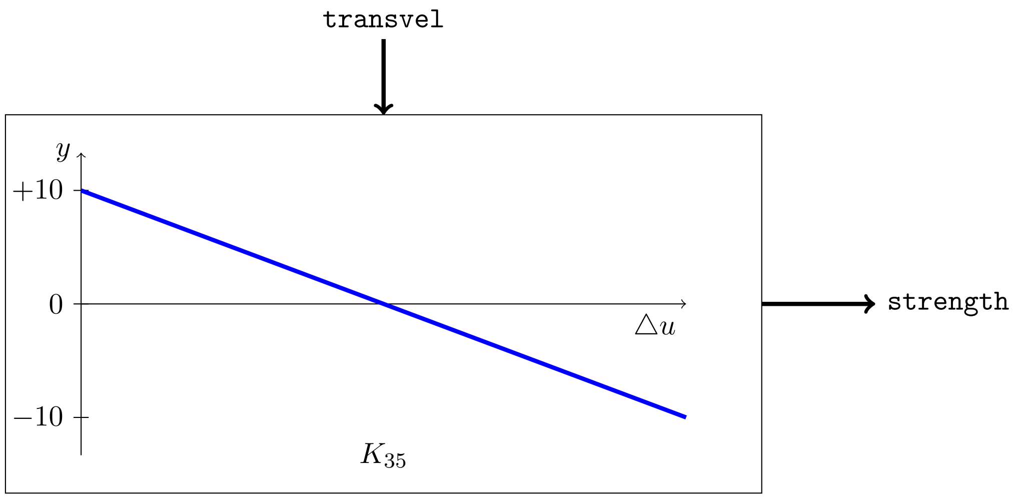

Figure 5.5: Initial configuration of the map K_{35}

The certainty values for these maps can be set high, as there is little doubt about the general form of the influences, just about the exact values. An initial certainty value of 90 for these four maps, and a certainty value of 30 for all other means that for now, only the predefined maps are slowly adapted, and eventually other configurations are tested out for all other maps.
The rulebase, the defuzzification information and the initial certainty values is all a human operator needs to provide in order to get the system to work. The presented fuzzy system should increase the shooting performance and show learning capabilities as the adaptation mechanism kicks in.

5.3 Results

The presented precalculated fuzzy system performs the task of influencing the the parameters responsible for shooting as expected. The adaption mechanism also works in the intended way. As the system behaves mostly identical to a regular fuzzy system, this means that the simplifications have not diminished its capabilities.
The fuzzy system was tested using a simulator which simulates the ball, the environment and the players behaviour, but no opponents. As mentioned, it is difficult to test the entire system in realistic, competetive conditions because these occur only a few times a year. The simulator, in turn, has several other drawbacks. First of all, the lack of opponents means that there is also no enemy goalkeeper. The performance will hence only be measured by detecting if the ball hits the goal or not. Secondly, some of the sensor values that are an input into the fuzzy system are not accurately available. This concerns especially the rotation velocity. The inflence of this value can not be reasonably optimizied using the simulation.
In the simulation environment, a number of shots was performed, and after 3 shots with a recorded result, the adaptation mechanism took over the current parameter set or rejected it. The performance was measured using the average grade using the system presented in table 4.2. However, since the event 'corner kick' can not occur due to the lack of opponents and the lack of a goalkeeper means that every instance of a goal kick means that the robot has missed, the table was adapted for the simulated environment. A miss, including occurences of a goal kick, yields a grade of 1, while a goal is still worth 10.
For 30 shots that were performed, the results are presented in table 5.4.
The first column shows the number of the run of 3 shots. In the second col-

run	1	2	3	4	5	6	7	8	9	10
grade	4	7	4	10	3.66	4	10	4	7	10
mov. avg.	4	5.5	5	6.25	5.73	5.44	6.09	5.83	5.96	6.37

Table 5.4: Restults of 30 performed shots on the goal in a simulated environment. The first row indicates the number of run, each run included 3 runs, the second column contains the resulting grade of the run and the third column shows the moving average grade. Only configurations which yielded a better moving average grade than before were adopted, but the grade was constantly updated in order to not produce perfect average grade than can not be surpassed.

umn the achieved grade is indicated. The third colum contains the moving average grade.
Before each run apart from the very first, the fuzzy system was adapted using the algorithm presented in section 4.3. After the run, the grade was compared to the current average grade. If it was better, the configuration that the optimization yielded was adopted, if not, it was dismissed.
This particular procedure assures that the system develops into the correct direction, but does not completely shut down the process when the moving average is too high to be surpassed. Note that the moving average grade can very well decrease when the grade of the current run in inferior to the moving average, only the fuzzy system is not adopted.
As a result the adapted configuration was taken over only 5 times of the possible 10 times, after the second, fourth, seventh, ninth and tenth run. The maps subsequently took the shape outlined in the figures 5.6, 5.7, 5.8 and 5.9.

In the case of the maps K_{24} and K_{32}, the value optimization influenced all

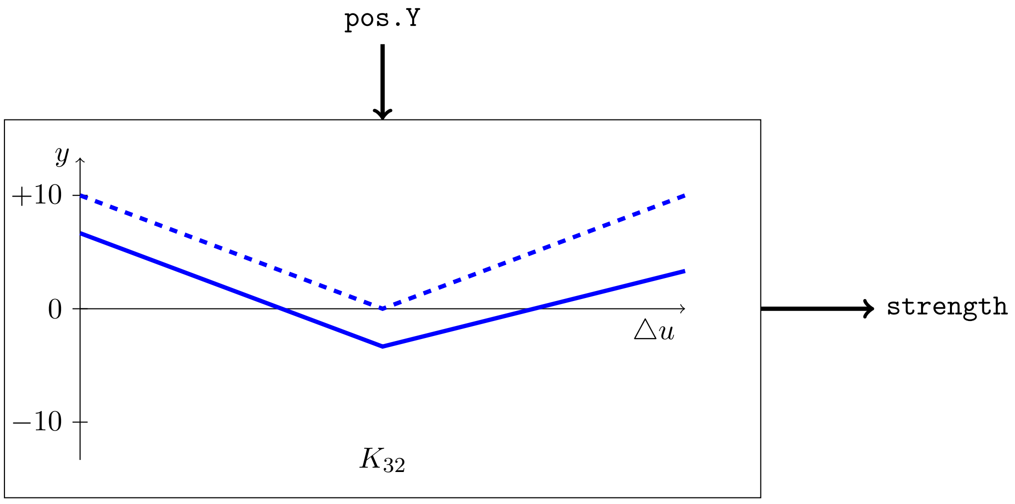

Figure 5.6: Optimized configuration of the map K_{32}. All values were dropped, practically diminishing the influence of the input `pos.Y` on the output `strength`. No base points were adapted, and no flags were reset. The shape of the underlying rule however was also retained.

of the values, but no base points or flags. It increased the influence of the rotation velocity `rotvel` on the vertical shootpoint `shootpoint.Y` and reduced

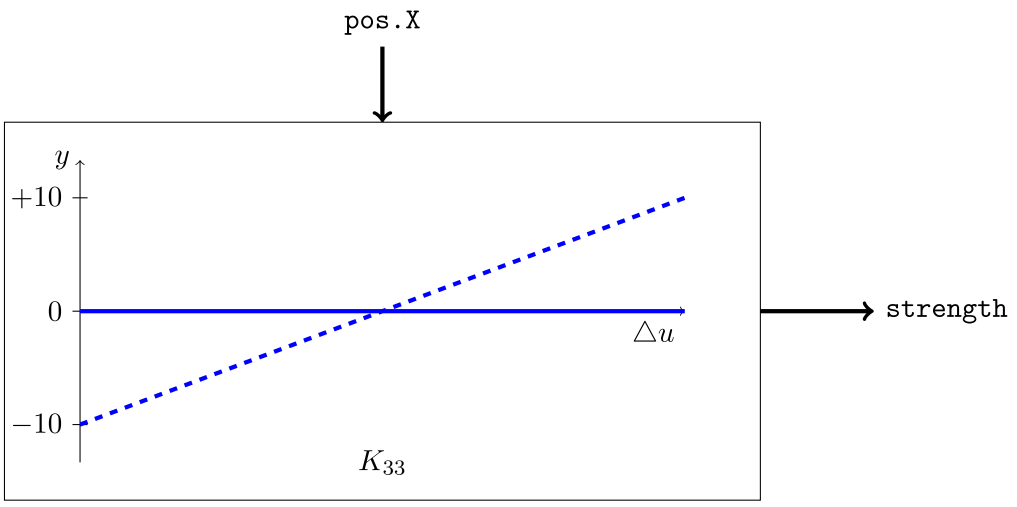

Figure 5.7: Optimized configuration of the map K_{33}. The influence of the input `pos.X` on the output `strength` was deactivated. This does not necessarily mean that there is no influence, but at the moment in the ongoing optimization the deactivation did not impair the results. The deactivation is reversible.

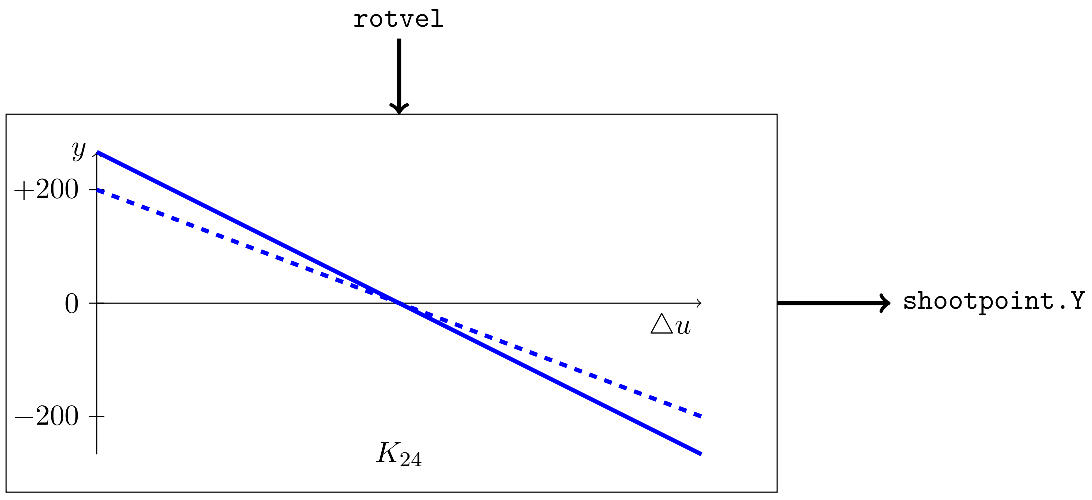

Figure 5.8: Optimized configuration of the map K_{24}. The influence of the input `rotvel` on the output `shootpoint.Y` was increased. This particular map however should be overlooked as the rotation velocity can not be accurately measured in the simulator.

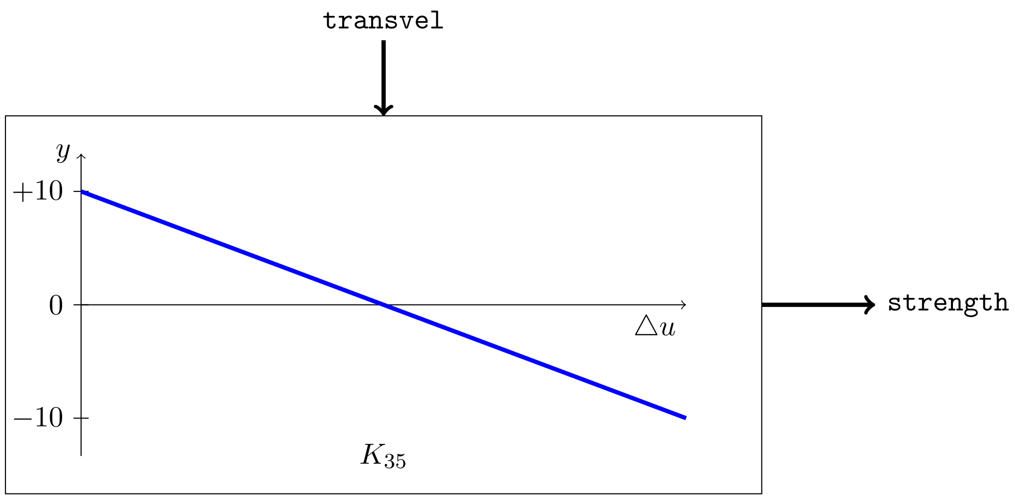

Figure 5.9: Optimized configuration of the map K_{35}. This map remained unchanged from the original configuration. One alternation was tested and proved inefficient.

the influence of the vertival position `pas.Y` on the shootstrength `strength`. The influence of the horizontal position `pos.X` on the shootstrength was deactivated altogether, while the map that describes the influence of the velocity `transvel` on the shootstrength remained unchanged. Some of the other maps that were deactivated from the start were changed during the optimization, but their influence was not beneficial or insufficient to the performance.

These results may be in some way random and the resulting maps certainly can not be directly used in a competition, but the overall system works as expected. The exact parameters of the optimization were chosen specifically for the simulation environment and should be changed when the system is used in a real competition. This concerns for instance number of shots before an adaptation is made. Also, the rulebase could be extended and the defuzzification values should be chosen with more care and experimentation.

Chapter 6

Outlook

6.1 Comprehensive Control of the Robots Behaviour

In this thesis the presented controller was only applied to the shoot mechanism. It was however developped to steer the robots entire behaviour at least to a limited extent. The application to this comprehensive task would really put the capabilities of the system to represent the underlying complexity to test. Also, the challenge to perform the adaptation is much greater, as the feedback on the success of general behaviour is much weaker than on specific actions like a shot on the goal.
For a meaningful comprehensive control the inputs can't just be momentary reads of certain sensors, but they also have to more comprehensive. For example, the decision on positioning requires (besides a cooperation among the robots, see section 6.3) information about past states in the game. It certainly is an open challenge to work out how this should ideally happen.
In the example of the shooting controller the outputs were clear and crisp variables with a direct influence. When the system is expanded, this won't necessarily be the case. Some outputs may show non-smooth behaviour and some may be interconnected. A method to deal with these output interconnections adn introduce many other new features is direct signal feedback, i.e. the outputs are directly fed back to the system as inputs. Figure 6.1 shows how a system with a direct feedback could look. The system has two 'real' inputs and one input that is a fed back output. This construction gives the controller an own dynamic, that means it now has a certain kind of internal memory and the output values are not always the same for identical inputs. This enables a stabilizing behaviour, but it can also lead to unstabilities, i.e. values that diverge to infinity. Feedback is a very powerful

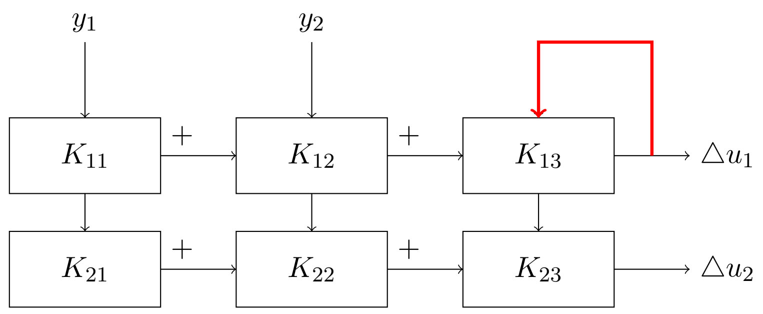

Figure 6.1: Precalculated fuzzy system with feedback

concept with great possible advantages and disadvantages. It would add a whole new level of complexity to the system.
The expansion of the systems inputs and outputs will certainly open up many new challenges and might expose flaws in the systems design.

6.2 Adaptation via Evolutionary Algorithms

The adaptation mechanism presented in section 4.3 works well to enhance the performance of the system and slowly adapt to new environments. However, it is not capable to fomulate new rules, or systematically recognize and preserve patterns. This is possible using evolutionary algorithms. It works different than the presented greedy hillclimbing algorithm in two ways. First, rather than storing only one configuration at a time, the algorithm works on a population of different ones. Then, the alteration of the elements complies to different rules, as it uses recombination and mutation modelled on similar procedures in nature. The basic procedure of an evolutionary algorithm is shown in figure 6.2.
To be able to apply an evolutionary algorithm, a *population* of several solution candidates has to be defined. The idea behind this is that the different strengths of the candidates can be combined into one solution during the recombination stage. This is definetely a desirable property for the adaptation mechanism of the fuzzy system, especially with a high number of inputs and outputs. The mutation stage could reuse the alternation methods developped for hillclimbing to generate neighbours of each solution. However, the complex and time consuming evaluation of the target function is a problem, as the recombination tends to generate many useless solution candidates, i.e. parameter sets which gravely impair the robots abilities.
In general, an evolutionary algorithm has to be devised very carefully, because if it isn't, it might aimlessly change the parameter set without ever committing to a promising path. However, if adjusted properly, an evolutionary algorithm can greatly increase the capabilities of the adaptation by recognizing and imitating patterns.

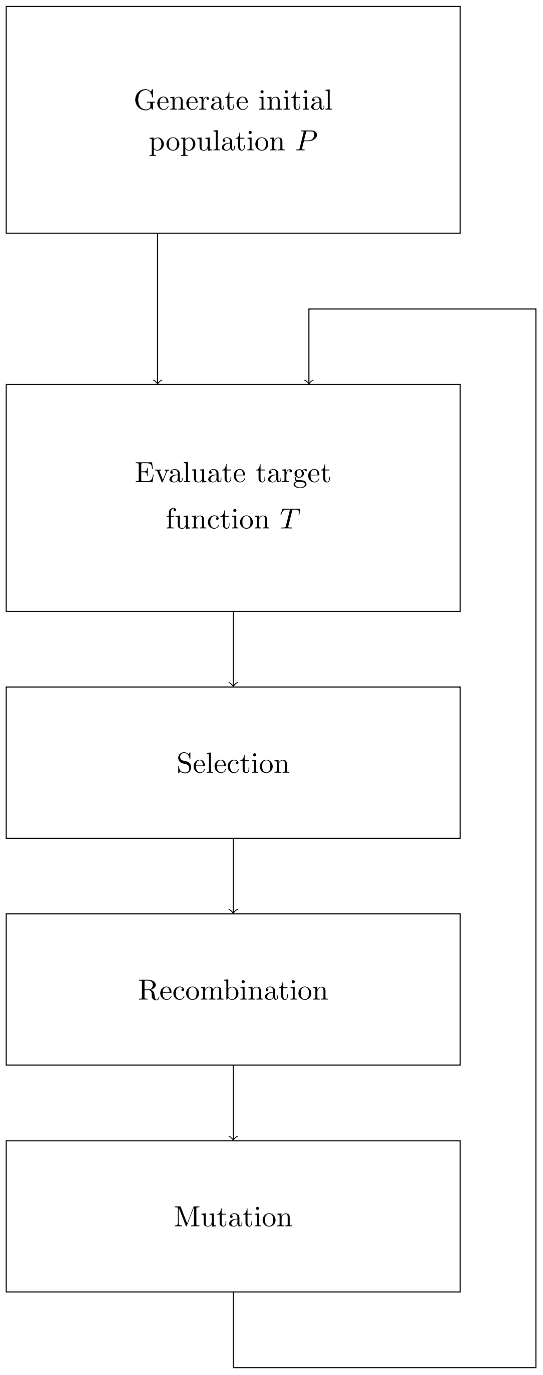

Figure 6.2: Basic structure of an evolutionary algorithm

6.3 Cooperative Control

Another field this thesis has not touched is cooperation. At the moment, the fuzzy system influences each robot individually. When the output parameters are to control parameters that affect the behaviour of multiple robots or even of the entire team, some sort of cooperation mechanism has to be devised. A cooperation between robots of the team is also helpful when parameters that only affect one robot are influenced, like in the example of the shooting controller.
Although each robot can individually execute its own optimization of the shooting controller, it makes sense to share results and coordinate the optimization process. There are multiple ways to implement a cooperative optimization. In the case of the shooting controller, the results of the most recent optimization step can just be shared among all robots, as there will never, or at least very rarely, be a conflict among two robots which perfomed a shot at the same time, or in such a short time interval that an optimization step could not be finished. Hence, a robot which obtained a new, superior fuzzy system can simply update the fuzzy systems of all other robots directly or via a dedicated server.
In other cases, the coordination between the robots is not that simple. If the optimization step can be executed so frequently that the entire fuzzy system can not be shared among all robots between the steps, some sort of negotiation algorithm needs to be introduced in order to coordiante the results and pick the best. This falls into the domain of multi-agent-systems and distributed optimization, see for instance [JSJJ08]. The methods of this active domain of research can also be used when a coordination is only helpful and desirable, but necessary and imparative. This is the case for parameters that steer the team behaviour, like the defensive and offensive formation, role distribution and behaviour during set pieces. In these cases the parameters need to be either identical or a set of parameters needs to be coordinated to result in a sensible behaviour. There is no trivial solution to all the problems that can occur. At the moment all these parameters are manually set following experience and experimentation.
Due to the very specific nature of the parameters, it requires some effort to apply and implement theoretic concepts to the system. However, a module that implements cooperative control of parameters in all these three cases would be an interesting and effective addition to the system.

Bibliography

[AB05] Andujar, Jose M. ; Bravo, Jose M.: Multivariable fuzzy control applied to the physicalÄìchemical treatment facility of a Cellulose factory. In: *Fuzzy Sets and Systems* 150 (2005), Nr. 3, 475 - 492. `http://dx.doi.org/10.1016/j.fss.2004.03.023`. – DOI 10.1016/j.fss.2004.03.023. – ISSN 0165–0114

[ACR99] Ascia, G. ; Catania, V. ; Russo, M.: VLSI hardware architecture for complex fuzzy systems. In: *Fuzzy Systems, IEEE Transactions on* 7 (1999), oct, Nr. 5, S. 553 –570. `http://dx.doi.org/10.1109/91.797979`. – DOI 10.1109/91.797979. – ISSN 1063–6706

[ADM87] Aicardi, M. ; Davoli, F. ; Minciardi, R.: Decentralized optimal control of Markov chains with a common past information set. In: *Automatic Control, IEEE Transactions on* 32 (1987), nov, Nr. 11, S. 1028 – 1031. `http://dx.doi.org/10.1109/TAC.1987.1104483`. – DOI 10.1109/TAC.1987.1104483. – ISSN 0018–9286

[BH94] Buckley, James J. ; Hayashi, Yoichi: Fuzzy neural networks: A survey. In: *Fuzzy Sets and Systems* 66 (1994), Nr. 1, 1 - 13. `http://dx.doi.org/10.1016/0165-0114(94)90297-6`. – DOI 10.1016/0165–0114(94)90297–6. – ISSN 0165–0114

[BPC00] Betin, F. ; Pinchon, D. ; Capolino, G.-A.: Fuzzy logic applied to speed control of a stepping motor drive. In: *Industrial Electronics, IEEE Transactions on* 47 (2000), jun, Nr. 3, S. 610 –622. `http://dx.doi.org/10.1109/41.847902`. – DOI 10.1109/41.847902. – ISSN 0278–0046

[BV79] Borkar, V. ; Varaiya, P.: Adaptive control of Markov chains, I: Finite parameter set. In: *Automatic Control,* *IEEE Transactions on* 24 (1979), dec, Nr. 6, S. 953 – 957. http://dx.doi.org/10.1109/TAC.1979.1102191. – DOI 10.1109/TAC.1979.1102191. – ISSN 0018–9286

[BY95] Buckley, James J. ; Yoichi, Hayashi: Neural nets for fuzzy systems. In: *Fuzzy Sets and Systems* 71 (1995), Nr. 3, 265 - 276. http://dx.doi.org/10.1016/0165-0114(94)00282-C. – DOI 10.1016/0165–0114(94)00282–C. – ISSN 0165–0114

[CT92] Cheng, B. ; Tong, H.: On Consistent Nonparametric Order Determination and Chaos. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 54 (1992), Nr. 2, pp. 427-449. http://www.jstor.org/stable/2346136. – ISSN 00359246

[HJV90] Horne, Bill ; Jamshidi, M. ; Vadiee, Nader: Neural networks in robotics: A survey. In: *Journal of Intelligent and Robotic Systems* 3 (1990), 51-66. http://dx.doi.org/10.1007/BF00368972. – ISSN 0921–0296. – 10.1007/BF00368972

[JSJJ08] Johansson, Björn ; Speranzon, Alberto ; Johansson, Mikael ; Johansson, Karl H.: On decentralized negotiation of optimal consensus. In: *Automatica* 44 (2008), Nr. 4, 1175 - 1179. http://dx.doi.org/10.1016/j.automatica.2007.09.003. – DOI 10.1016/j.automatica.2007.09.003. – ISSN 0005–1098

[Lee90] Lee, C.C.: Fuzzy logic in control systems: fuzzy logic controller. I. In: *Systems, Man and Cybernetics, IEEE Transactions on* 20 (1990), mar/apr, Nr. 2, S. 404 –418. http://dx.doi.org/10.1109/21.52551. – DOI 10.1109/21.52551. – ISSN 0018–9472

[LH97] Lund, H.H. ; Hallam, J.: Evolving sufficient robot controllers. In: *Evolutionary Computation, 1997., IEEE International Conference on*, 1997, S. 495 –499

[LT03] Li, Han-Xiong ; Tong, Shaocheng: A hybrid adaptive fuzzy control for a class of nonlinear MIMO systems. In: *Fuzzy Systems, IEEE Transactions on* 11 (2003), feb, Nr. 1, S. 24 – 34. http://dx.doi.org/10.1109/TFUZZ.2002.806314. – DOI 10.1109/TFUZZ.2002.806314. – ISSN 1063–6706

[LYL96] Lewis, F.L. ; Yesildirek, A. ; Liu, Kai: Multilayer neural-net robot controller with guaranteed tracking performance. In: *Neural Networks, IEEE Transactions on* 7 (1996), march, Nr. 2,

S. 388 –399. http://dx.doi.org/10.1109/72.485674. – DOI 10.1109/72.485674. – ISSN 1045–9227

[Man74] Mandl, P.: Estimation and Control in Markov Chains. In: *Advances in Applied Probability* 6 (1974), Nr. 1, pp. 40-60. http://www.jstor.org/stable/1426206. – ISSN 00018678

[MH00] Mitra, S. ; Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. In: *Neural Networks, IEEE Transactions on* 11 (2000), may, Nr. 3, S. 748 –768. http://dx.doi.org/10.1109/72.846746. – DOI 10.1109/72.846746. – ISSN 1045–9227

[MP43] McCulloch, Warren ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *Bulletin of Mathematical Biology* 5 (1943), 115-133. http://dx.doi.org/10.1007/BF02478259. – ISSN 0092–8240. – 10.1007/BF02478259

[Paw82] Pawlak, Zdzisław: Rough sets. In: *International Journal of Parallel Programming* 11 (1982), 341-356. http://dx.doi.org/10.1007/BF01001956. – ISSN 0885–7458. – 10.1007/BF01001956

[PS94] Pawlak, Zdzisaw ; Sowinski, Roman: Rough set approach to multi-attribute decision analysis. In: *European Journal of Operational Research* 72 (1994), Nr. 3, 443 - 459. http://dx.doi.org/10.1016/0377-2217(94)90415-4. – DOI 10.1016/0377–2217(94)90415–4. – ISSN 0377–2217

[Rab12] Rabin, Steve: *Free Fuzzy Logic Library (FFLL)*. http://ffll.sourceforge.net/, 1 2012

[Ros58] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review;Psychological Review* 65 (1958), Nr. 6, S. 386–408. http://dx.doi.org/10.1037/h0042519. – DOI 10.1037/h0042519. ISBN 1939–1471(Electronic);0033–295X(Print)

[RPW^{+}03] Rusu, P. ; Petriu, E.M. ; Whalen, T.E. ; Cornell, A. ; Spoelder, H.J.W.: Behavior-based neuro-fuzzy controller for mobile robot navigation. In: *Instrumentation and Measurement, IEEE Transactions on* 52 (2003), aug., Nr. 4, S. 1335 –

1340. http://dx.doi.org/10.1109/TIM.2003.816846. – DOI 10.1109/TIM.2003.816846. – ISSN 0018–9456

[TCC^{+}95] TONG, Howell ; CHAN, K. S. ; COX, D. R. ; CUTLER, Colleen D. ; GUÉGAN, D. ; JENSEN, Jens L. ; JOHANSEN, Søren ; LAWRANCE, A. J. ; LEBARON, Blake ; OZAKI, T. ; NYCHKA, Douglas W. ; ELLNER, Stephen ; BAILEY, Barbara A. ; GALLANT, A. R. ; SMITH, L. R. ; SMITH, R. L. ; WOLFF, Rodney C. L.: A Personal Overview of Non-Linear Time Series Analysis from a Chaos Perspective [with Discussion and Rejoinder]. In: *Scandinavian Journal of Statistics* 22 (1995), Nr. 4, pp. 399-445. http://www.jstor.org/stable/4616372. – ISSN 03036898

[Wan93] WANG, L.-X.: Stable adaptive fuzzy control of nonlinear systems. In: *Fuzzy Systems, IEEE Transactions on* 1 (1993), may, Nr. 2, S. 146 –155. http://dx.doi.org/10.1109/91.227383. – DOI 10.1109/91.227383. – ISSN 1063–6706

[Zad65] ZADEH, L.A.: Fuzzy sets. In: *Information and Control* 8 (1965), Nr. 3, 338 - 353. http://dx.doi.org/10.1016/S0019-9958(65)90241-X. – DOI 10.1016/S0019–9958(65)90241–X. – ISSN 0019–9958