Methods of Computational Intelligence for Modeling and Data Representation of Complex Systems

Várkonyiné Kóczy Annamária, PhD

MTA doktori értekezés
Budapest, 2008

Contents

Introduction and objectives

Nowadays, engineering science tackles problems of previously unseen spatial and temporal complexity. In solving engineering problems, the processing of the information is performed typically by model-based approaches which contain a representation of our knowledge about the nature and the actual circumstances of the problem in hand. These models are part of the (usually computer based) problem solving procedure. Up to recently, classical problem solving methods proved to be entirely sufficient in solving engineering problems, however in our days traditional information processing methods and equipment fail to handle the problems in a large number of cases. It became clear that new ideas are required for the specification, design, implementation, and operation of sophisticated systems.

Fortunately however, parallel with the complexity explosion of the problems in focus, we can witness the appearance of increased computer facilities, and also new, fast and intelligent techniques. As a consequence of the new challenges, not only the new problems, arising from the increasing complexity have to be solved, but also new requirements, formulated for information processing, have to be fulfilled. The previously accepted and used (classical) methods only partially cope with these challenges.

Due to the growth of the amount of available computational resources, more and more complex tasks can/are to be addressed by more and more sophisticated solutions. In many cases, processing has to be solved on-line, parallel with information acquisition or operation. Many processes should run with minimized human interaction or completely autonomously, because the human presence is impossible, uncomfortable, or it is against human nature. Furthermore, no matter how carefully the design of the operation/processing scheme is done, we can not avoid changes in the environment; concerning the goals of the processing, one also have to deal with time/resource insufficient situations caused by failures or alarms. It is an obvious requirement that the actual processing should be continued to ensure appropriate performance in such cases.

What previously was called ‘processing’ became ‘preprocessing’, giving place to more advanced problems. Related to this, the aims of the newly developed preprocessing techniques have also changed. Besides the improvement of the performance of certain algorithms, a new requirement arose: the introduced methods have to give more support to the ‘main’ processing following them. In signal processing, image processing, and computer vision this trend means that the previous processing tasks like noise smoothing, feature extraction (e.g. edge and corner detection), and even pattern recognition became part of the preprocessing phase and processing covers fields like 3D modeling, medical diagnostics, or the automation of intelligent methods and systems (automatic 3D modeling, automatic analysis of systems/processes etc.).

This work deals with the problems outlined above and tries to offer appropriate computational tools in modeling, data representation, and information processing with dedicated applications in the fields of measurement, diagnostics, signal and image processing, computer vision, and control.

The special circumstances of online processing and the insufficient, unambiguous, or even lacking knowledge call for fast methods and techniques, which are flexible with respect to the available amount of resources, time, and information, i.e. which are able to tolerate uncertainty and changing circumstances. Thus, the focus of this work is on methods of this type.

Part I addresses the topic of signal processing and data representation. New fast recursive transformed-domain transformations, filter and filter-bank implementations are presented as replies to the complexity challenges. Besides complexity reduction, and the decrease of processing delay, the introduction of soft computing (e.g. fuzzy and anytime) techniques offers flexible and robust

operational modes. The newly initiated concepts and techniques however require reconsideration of data and error representation. This question is also briefly addressed here and the usage of some new data and error models is analyzed.

In Part II, new problems of image processing are studied. Different aspects of information enhancement, like corner detection, useful information extraction, and high dynamic range imaging are investigated. The proposed novel tools usually include improvements achieved by fuzzy techniques. Besides being very advantageous from the automation point of view, they may also give an efficient support to the increase of the reliability of further processing.

Based on the results of the previous chapter, Part III presents new methods in computer vision together with the basic elements of a possible intelligent expert system in car-crash analysis. Autonomous camera calibration and 3D reconstruction are solved with the help of recent results of epipolar geometry and fuzzy techniques. The proposed analyzer system is able to autonomously determine the 3D model of a crashed car and estimate speed and direction of the crash currently in case of one car hitting a wall.

Part IV introduces the generalized concept of anytime processing. Fuzzy and neural network based models are very advantageous if the processing has to be done under incomplete and imperfect circumstances, or if it is crucial to keep computational complexity low. Here, the usage of such models is investigated in anytime systems. With the help of (Higher Order) Singular Value Decomposition based complexity reduction, certain fuzzy and neural network models are extended to anytime use and new error bounds are determined for the non-exact models. In case of Product-Sum-Gravity-Singleton fuzzy systems, a novel transformation opening the way for iterative evaluation is also derived.

Part V deals with observer based fuzzy and neural network model inversion. Model inversion has a significant role in measurement, diagnostics, and control, however until recently, the inversion of fuzzy and neural network model could be solved only with strong limitations on the model. The presented new concept can be applied in more general circumstances and the condition of low complexity, global convergence can also be met.

Part I New Methods in Digital Signal Processing and Data Representation

Signal processing is involved in almost all kinds of engineering problems. When we measure, estimate, or qualify various parameters and signals during the monitoring, diagnostics, control, etc., procedures, we always execute some kind of signal processing task. The performance of measurement and signal processing has usually a direct effect on the performance of the whole system and vice versa, the requirements of the monitoring, diagnostics, or control raise demands in the measurement and signal processing scheme.
The recent development of modern engineering technology, on one hand lead to new tools like model based approaches, computer based systems (CBS), embedded components, dependability, intelligent and soft computing based techniques offering new possibilities; while on the other hand new problems like complexity explosion, need for real-time processing (very often mission-critical services), changing circumstances, uncertain, inaccurate and insufficient information have to be faced. The current situation of engineering sciences is even more complex due to the fact that in very complex systems various modeling approaches, expressing different aspects of the problem, may be used together with a demand for well orchestrated integration. The price to be paid for integration is that the traditional metrological concepts, like accuracy and error have to be reconsidered and in many cases they are no more applicable in their usual approved sense. Furthermore, the representation method of the uncertainty and the error must be in harmony with both the modeling and information processing method. They also need to be uniform or at least "interpretable" by other representation forms. All these lead to the consideration of new, fast and flexible computational and modeling techniques in measurement and signal processing.
This chapter deals with new methods in the fields of digital signal processing and data representation, which offer solution to some of the above outlined problems.

1.1 Introduction

In signal processing the model of a system in question serves as a basis to design processing methods and to implement them at the equipment level. In case of complex signal processing analytical models, it is not enough to obtain a well defined numerical optimal information processing. The complexity of the problem manifests itself not only in the (possibly huge) amount of tasks to be solved or as a hierarchy of subsystems and relations: often several modeling approaches are needed to grasp the essence of the modeled phenomenon. Analytical models rarely suffice. Numerical information is frequently missing or it is uncertain, making place for various qualitative or symbolic representation methods.
The subject of Part I is a topics with increasing actuality. New methods of signal processing are introduced and their role in overcoming several aspects of the above problems is investigated. Because of the nature of the problems and because we usually need online processing to solve the tasks in hand, only fast algorithms of digital signal processing and new soft computing based methods are concerned. The idea of using fuzzy logic and neural networks combined with other tools like anytime algorithms comes from many other fields of research where similar problems have appeared. It appeared natural to explore and adopt the solutions.

In the fields of Artificial Intelligence (AI), Soft Computing (SC), and Imprecise Computations (IC), numerous methods have been developed, which address the problem of symbolic, i.e. non-numerical information processing and a rational control of limited resources. They may offer a way in signal processing as well when classical methods fail to solve the problems [S46], [S48].
Besides the "basic" method of SC and IC, a major step was done by the introduction of anytime models and algorithms that offer an on-line control over resources and the trade-off between accuracy and complexity (i.e. the use of resources).
The concepts and algorithms presented in this part are advantageous from complexity point of view however the reduction of complexity most often runs parallel with the decrease of accuracy that makes necessary in certain applications to change from one model to another according to the actual situation. The use of different models within one system initiates further difficulties. We can not forget that results (outputs) produced by different representation methods have to be comparable, convertible, interpretable by each other. This leads to a huge number of open questions not fully answered yet.
The chapter is organized as follows: In Section 1.2 a new concept of multisine synthesis and analysis via Walsh-Hadamard transformation is presented. It is a low complexity, symmetrical, highly parallel structure-pair, also decreasing such disadvantageous effects like picket fence and leakage. In Section 1.3 the usage of fast sliding transforms is discussed in transform-domain adaptive filtering. The resulted reduction of the delay is also analyzed. Section 1.4 is devoted to a novel, low complexity implementation of Fourier transformation, resulting in anytime operational mode, which is able to produce good quality frequency and amplitude estimations of multisine signal components as early as the quarter of the signal block. In Section 1.4 a new overcomplete signal representation is proposed for representing non-stationary signals. Finally, in Section 1.5 new problems of data and uncertainty representation, arising from the non-linearity of systems and from the use of non-conventional modeling methods, are investigated together with some possible solutions offered.

1.2 Multisine synthesis and analysis via Walsh-Hadamard transformation

In this section, we present a new multisine synthesizer and analyzer based on a special filter-bank pair. The new tool can efficiently be utilized for solving system identification problems. The filter-banks are fast indirect implementations of the inverse discrete (IDFT) and discrete Fourier Transformation (DFT) algorithms providing low computational complexity and high accuracy. The proposed structures are based on the proper combination of polyphase filtering and the Walsh-Hadamard (WHT) Transformations. The inherent parallelism of these structures enables very high speed in practical implementations and the use of several parallel A/D, D/A converters.

1.2.1 Introduction

The classical solutions in digital signal processing offer very well established methods for the frequency-domain signal representation like the discrete Fourier Transformation (DFT) and its fast algorithm the fast Fourier transformation. [1], [2]. However, we have to face serious limits due to the contradictory requirements of magnitude and frequency resolution. To reduce these problems the application of multisine perturbation signals came into focus [3] recently.
Another disadvantageous aspect of the widely used FFT techniques is that they are operated block-oriented, (i.e. the algorithm processes whole blocks of data). Thus, they do not directly support real-time signal processing, which also has become an important claim in this field.
Using the classical recursive DFT methods [1] the real-time processing can be solved with a higher computational burden relative to the FFT. Later, the classical version based on the Lagrange structure was replaced by an observer structure [4], [5] however, computational

complexity remained the same.
Recently a fast implementation of the recursive DFT has been developed [6] which combines the idea of polyphase filtering [7] and the FFT; it is based on the decomposition of a larger size single-input multiple-output (SIMO) DFT filter-bank into proper parallel combination of smaller ones. Its computational complexity is in direct correspondence with the FFT and besides, its polyphase nature provides additional advantages in parallelization.
Using this structure, a fast adaptive Fourier analyzer [8] can be derived and by applying recursive building blocks within the structure a recursive DFT with fading memory can also be implemented [9]. These have real importance in system identification problems where periodic, multi-frequency perturbation signals are applied.
In this Section, a dedicated novel structure pair is presented for the efficient solution of the above described problems. In Subsection 1.2.2 the application of synchronized signal synthesis and analysis is proposed while in section 1.2.3 the new fast recursive synthesis and analysis structure based on the Walsh-Hadamard Transformation will be detailed. Subsection 1.2.4 is devoted to the extension towards multiple parallel running A/D converters.

1.2.2 Synchronized synthesis and analysis ([S6], [S7], [S34], [S35])

In multisine measurements the perturbation signal of the system to be identified is a multisine signal and the amplitudes and phases of the harmonic components of the response are to be measured. The accuracy of the identification depends on the accuracy of the determination of the input/output ratio of the components. If we synchronize the synthesis and analysis of the harmonic components then the systematic error of the DFT/FFT methods can be eliminated. The synchronization can be solved through the use of a joint pair of signal synthesizer and analyzer. Fig. 1.1 shows the block diagram of the multisine measuring procedure.

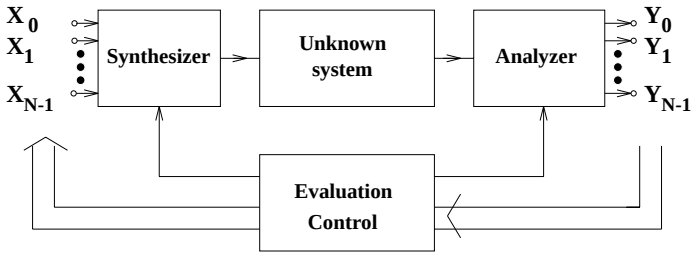


Fig. 1.1 Block diagram of the multisine measuring setup

The synthesizer and analyzer units contain the signal conditioning circuits and the D/A and A/D converters, respectively. The synthesizer generates a multisine input signal for the system with components in given frequency positions and with maximized quasi-uniform amplitudes. This can be optimized through the appropriate setting of the phase positions [3]. The analyzer is the inverse of the synthesizer: its channels are "tuned" to the components of the multisine signal. The transfer function is determined by the ratio of the complex output/input values. This is evaluated by the common control unit.

1.2.3 Synthesis and analysis via Walsh-Hadamard transformation ([S2], [S6], [S33])

The block diagram of the proposed synthesizer is given in Fig. 1.2. It is a multiple input single output (MISO) system which can generate arbitrary periodic waveforms. It operates like a parallel to serial converter, i.e. it has a nonzero parallel input in every *N*-th step and generates a sequence of *N* samples corresponding to the actual input. If this input is repeated in every *N*-th step the output will be a periodic waveform. The overall structure implements a complete weighted set of Walsh-Hadamard basis sequences in an efficient form with a complexity

corresponding to that of the fast algorithms. It is important to note that the input signal is the Walsh-Hadamard representation of the sequence to be generated with an accuracy depending only on the accuracy of the weights since within the structure only additions and subtractions are to be performed. The output is obtained via a demultiplexer which can be applied to a D/A converter.

As an example Fig. 1.3 shows a single sinusoid waveform generated using the above method. The input values of the Walsh-Hadamard synthesizer can be easily calculated as the Walsh-Hadamard transform of the time-sequence to be generated. Let us denote the vector of this sequence by $\underline{x}=[x(0), x(1),...,x(n-1)]^T$, and the $N*N$ transformation matrix by $\underline{\underline{W}}$. The N vectors to be applied in every N-th step at the input of the synthesizer are given by $\underline{X}=\underline{\underline{W}}\underline{x}$.

The block diagram of the analyzer structure is given in Fig. 1.4. It is in complete correspondence with the synthesizer and can be considered as a serial to parallel converter system maximally decimated at its output if necessary. The different channels calculate the Walsh-Hadamard coefficients corresponding to the last N input samples. The complexity remains the same as for the generation.

If the output of the synthesizer is connected directly to the input of the analyzer then after N

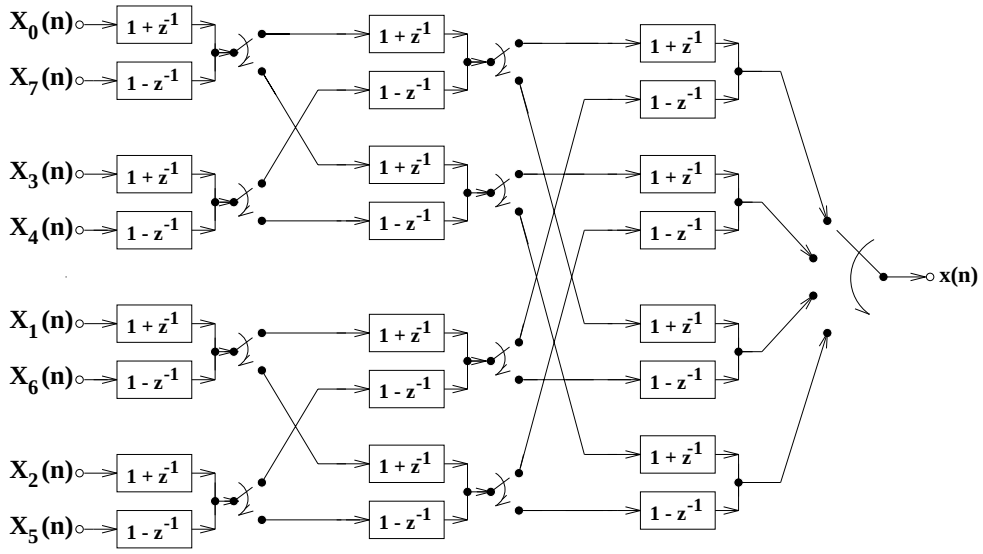


Fig. 1.2 Block diagram of the Walsh-Hadamard synthesizer for N=8

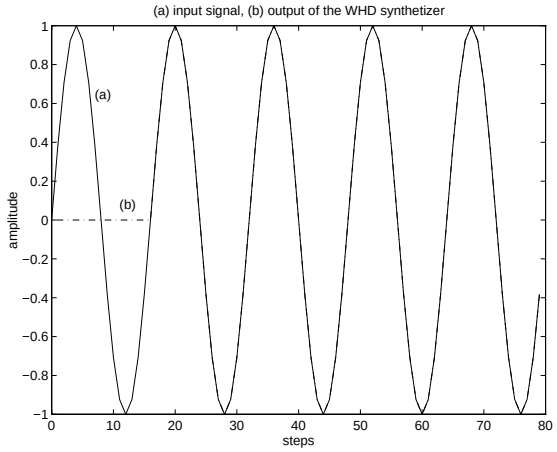


Fig. 1.3 Single sinusoid waveform generated using WHT-synthesizer: (a) desired signal; (b) output signal

steps its output values will equal the Walsh-Hadamard transform components, i.e. the components of $\underline{X}$. If there is a system to be identified in between, then we can characterize the unknown system by the corresponding channel inputs $\underline{X}$ and outputs $\underline{Y}$ of the synthesizer and the analyzer, respectively. The widely used frequency domain characterization of the system, however, requires some additional computations. On the input side the vector of the complex Fourier components $\underline{X}_F$ can be calculated as

$$\underline{X}_F = \underline{\underline{F}}\,\underline{x} = \underline{\underline{F}}\,\underline{\underline{W}}^{-1}\underline{X} = \underline{\underline{V}}\,\underline{X} \tag{1.1}$$

where $\underline{\underline{F}}$ stands for the N*N discrete Fourier Transformation (DFT) matrix, while at the output of the analyzer

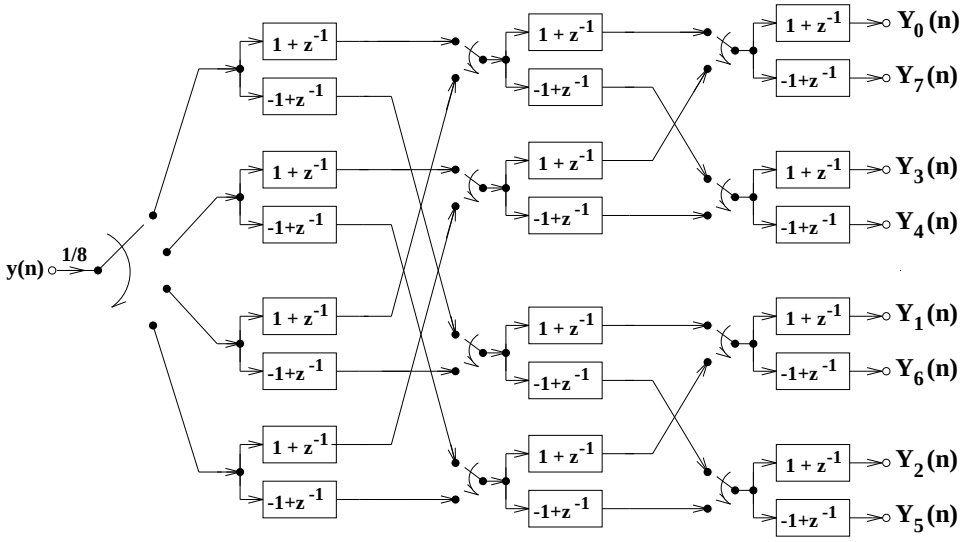


Fig. 1.4 Block diagram of the Walsh-Hadamard analyzer for N=8

$$\underline{Y}_F = \underline{\underline{F}}\,\underline{y} = \underline{\underline{F}}\,\underline{\underline{W}}^{-1}\underline{Y} = \underline{\underline{V}}\,\underline{Y}\,. \tag{1.2}$$

The stationary behavior of the system to be identified can be characterized by the transfer values derived as the ratio of the corresponding components of $\underline{Y}_F$ and $\underline{X}_F$ as the transients of the overall system die out. It is important to note that the results will be available at the end of a complete sequence of N samples, i.e. in every N-th step.

Based on (1.1), the practical measurements start with the specification of the proper multisine signal. This is performed by setting the proper initial magnitude and phase via the components of $\underline{X}_F$ (see [3]). The next step is the calculation of the vector $\underline{X}=\underline{\underline{V}}^{-1}\underline{X}_F$, which is directly used in the signal generation (see Fig. 1.2). Finally, the output of the analyzer (see Fig. 1.4) should be introduced into (1.2) to get the corresponding sine-wave parameters. The measurement setup described above is a finite impulse response FIR filter-bank which performs a sliding-window mode of operation and therefore its outputs characterize always the last block of N samples, i.e. each channel of the bank can be considered as a "moving-average" filter. If we consider the effect of the sliding window also for $\underline{X}_F$ then we can extend the method to get a new measurement in every step.

In practical measurements the presence of noise is unavoidable If we model the noise effects as an additive white noise input to the analyzer having variance σ^2, then this variance will be reduced to σ^2/N in every channel. Further improvement can be achieved if we introduce the fading memory effect described in [11] and [12]. The characterization of this effect can be given as

$$\frac{1-a}{N}\frac{1-z^{-N}}{1-az^{-N}}\frac{1}{1-z^{-1}}, \tag{1.3}$$

where the poles located at the positions determined by the *N*-th roots of 0 < a < 1 are responsible for further noise reduction. The effects of these poles can be illustrated with the corresponding magnitude characteristics (see Fig. 1.5 for different *a* values). The extension of the Walsh-Hadamard transformer to such a fading memory version can be solved with the application of simple second-order recursive blocks. If the first stage of the structure in Fig. 1.4 consisting of *N*/2 2*2 Walsh-Hadamard transformers is realized using e.g. the second-order blocks of Fig. 1.6 with

$$r_0 = r_1 = \frac{1-a}{2} \tag{1.4}$$

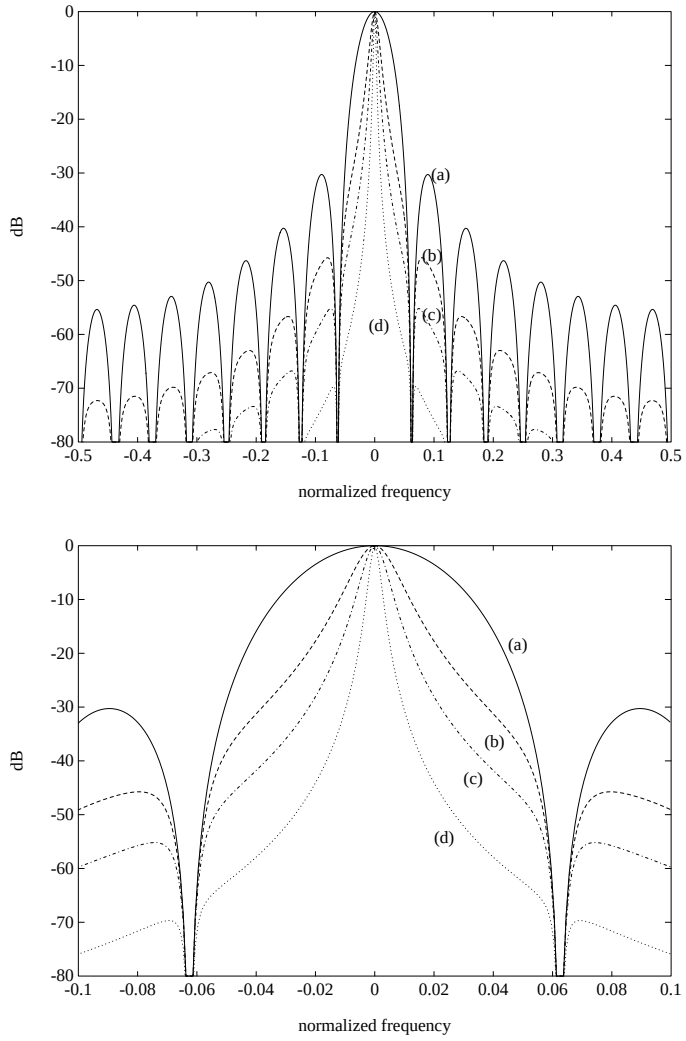


Fig. 1.5 Magnitude characteristics of the filter-banks with different $a > 0$ parameters: (a) a=0, (b) a=0.4, (c) a=0.6, and (d) a=0.8.

then the overall structure will produce the modified performance described by (1.3). The prize to be paid for this improvement in noise reduction is the increase of the measurement time, since the poles introduced will cause longer transients. As simple example Fig. 1.7 shows the simulated measurement results of a 5th-order Butterworth low-pass filter. After 10 complete periods of the multisine input the measurement results show very good coincidence with the calculated theoretical values.

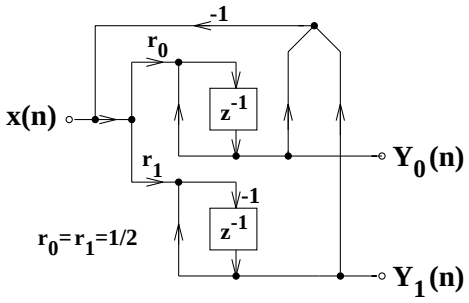


Fig. 1.6 Second-order recursive filter block

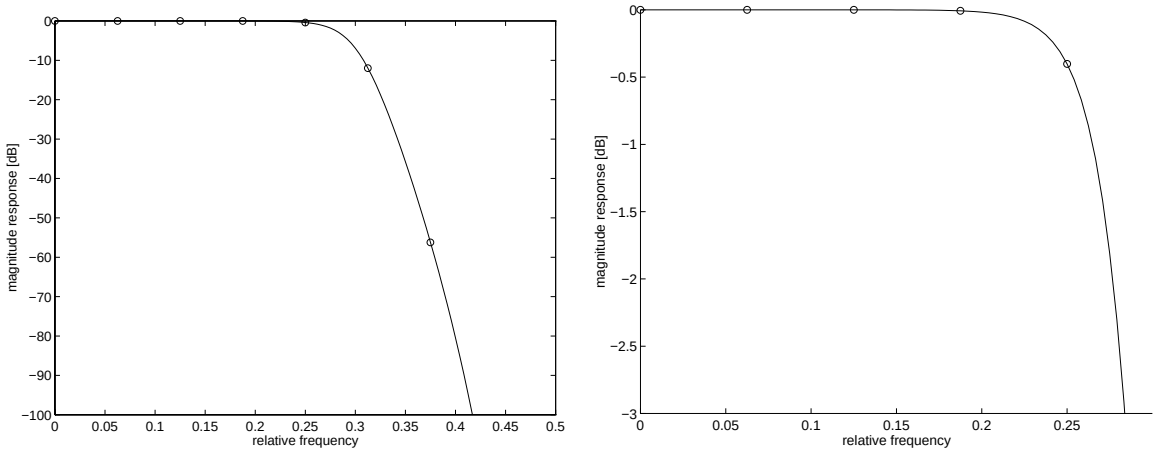


Fig. 1.7 Simulation results of a 5th-order Butterworth low-pass filter: (-) theoretical results, (o) simulated values

1.2.4 Multiple A/D converters within the fast polyphase transformed domain analyzer [S33]

A more detailed block diagram of the complete measuring system shown in Fig. 1.1 is given in Fig. 1.8. This system can be considered as a highly parallel network analyzer, i.e., it is devoted to problems where perturbation signals are to be applied and the system to be measured can be considered as linear. The multisine synthesizer and the signal analyzer operate synchronously together with the D/A and A/D converters. If frequency transposition circuits are also applied then a synchronized "carrier-band" analysis, i.e., "zoom" analysis is also possible. Concerning errors within the system: if we neglect the quantization errors of the digital signal processing parts then only the side-effects of the D/A and A/D conversions and the frequency transpositions are to be considered. Fortunately, frequency mismatch problems can be completely avoided since frequency transpositions can share common frequency reference. On the other hand, the magnitude and phase errors of these circuits can be "measured" by the system itself, since the output of the signal generator can be directly analyzed and the calculated difference of the multisine signal parameters can directly serve for correction. If the D/A and A/D converters share common voltage reference then the system is capable to calibrate itself automatically relative to this reference.

Nowadays the accuracy of measurement systems can be considerably improved by the direct utilization of signal processing techniques. The available DSP processors can provide also good speed performance. Speed and accuracy, however, are contradictory requirements in measurements. As far as A/D conversion is concerned flash converters can operate at very high speed, but their resolution is not acceptable. Sigma-delta A/D converters provide excellent

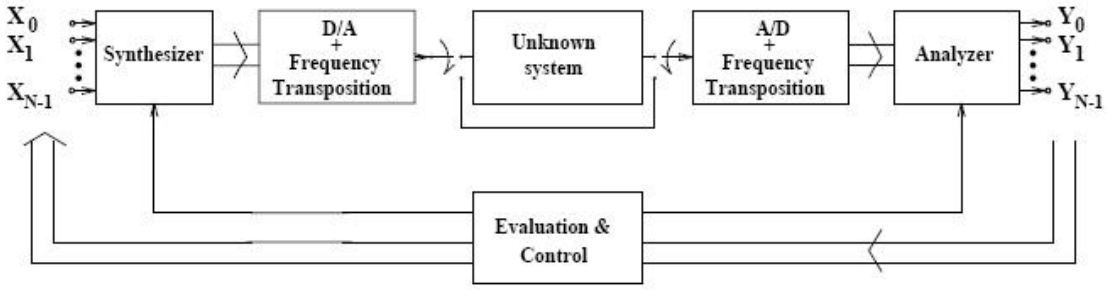


Fig. 1.8 Detailed block diagram of the multisine measuring system

resolution and accuracy, but at the prize of low conversion rate. For high resolution (>16 bits) measurements a possible alternative can be the application of parallel sigma-delta converters at the input of the polyphase DFT/WHT analyzer (see Fig 1.4) since its internal processing elements operate at a much lower rate. The delay caused by the converters can be easily compensated either in the phase of the generated signal or as a correction during self-calibration. If the signal processing part of the system would limit the speed of operation then the proposition in the previous section may help where signal generation and analysis is performed via the computationally extremely efficient Walsh-Hadamard transformation.

1.3 Fast sliding transforms in transform-domain adaptive filtering

Transform domain adaptive signal processing proved to be very successful in numerous applications especially where systems with long impulse responses are to be evaluated. The popularity of these methods is due to the efficiency of the fast signal transformation algorithms and that of the block oriented adaptation mechanisms. In this section the applicability of the fast sliding transformation algorithms is investigated for transform domain adaptive signal processing. It is shown that these sliding transformers may contribute to a better distribution of the computational load along time and therefore enable higher sampling rates. It is also shown that the execution time of the widely used Overlap-Save and Overlap-Add Algorithms can also be shortened. The prize to be paid for these improvements is the increase of the end-to-end delay which in certain configurations may cause some degradation of the tracking capabilities of the overall system. Fortunately, however, there are versions where this delay does not hurt the capabilities of the adaptation technique applied.

1.3.1 Introduction

In recent years, transform-domain adaptive filtering methods became very popular especially for those applications where filters with very long impulse responses are to be considered [13]. The basic idea is to apply the fast Fourier Transformation (FFT) for signal segments and to perform adaptation in the frequency domain controlled by the FFT of an appropriate error sequence. There are several algorithms based on this approach [13] and further improvements can be achieved ([14]). The formulation of the available methods follows two different concepts. The first one considers transformations as a “single” operation to be performed on data sequences (block-oriented approach), while the other emphasizes the role of multirate analyzer and synthesizer filter-banks.
In this section, using some former results concerning fast sliding transformation algorithms ([6], [S33]) a link is developed which helps to identify the common elements of the two approaches. The fast sliding transformers form exact transformations, however, operate as polyphase filter-banks. These features together may offer further advantages in real-time applications, especially

when standard DSP processors are considered for implementation.
There are two major configurations for transform-domain adaptive filtering (see Fig. 1.9). First let us consider standard (not sliding) transformers. Here transformers perform a serial to parallel conversion while inverses convert to the opposite direction. Adaptation is controlled by the transformed input and error signals once for each input block, i.e. decimation is an inherent operation within these algorithms. At the same time, investigations related to the block-oriented approach rarely consider real-time aspects of signal processing. It is typically supposed that sampling frequency is relatively low compared to the computational power of the signal processors and therefore, if a continuous flow of signal blocks must be processed, the block-period is enough to compute the transformations and the filter updating equations. Moreover, in the case of the widely used Overlap-Save and Overlap-Add methods (see e.g. [13] and Subsection 1.3.4) filter updating requires further transformations, therefore further computational power is needed.

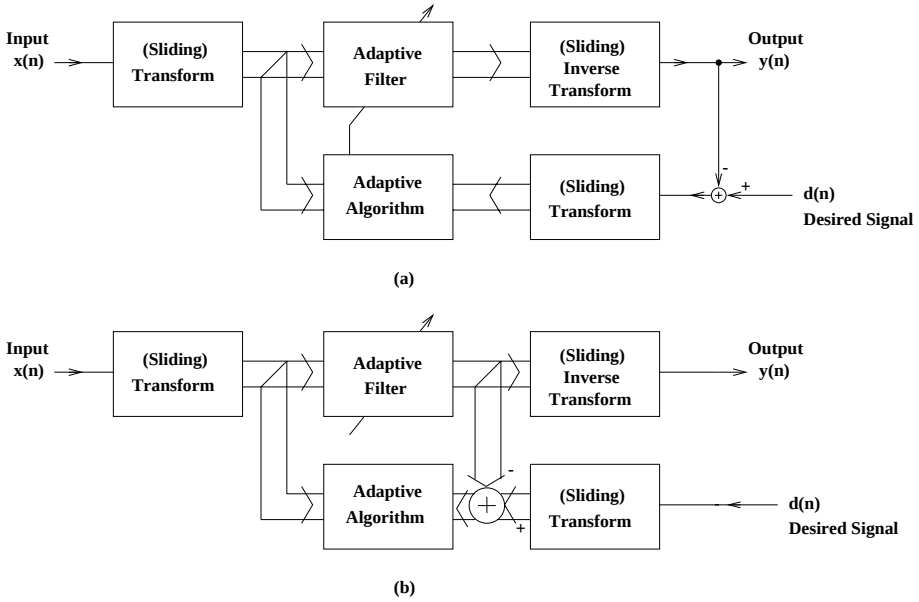


Fig. 1.9 Frequency-domain adaptive filter configurations

Filter-bank and consequently fast sliding transformation techniques do not offer extra savings in computations however they may provide a much better distribution of the computational load with time. This means that potentially they give better behavior when real-time requirements are to be met. This section present a new combined structure having advantageous computational complexity and load features. The section is organized as follows: Subsection 1.3.2 describes the basic idea of the fast sliding transforms while Section 1.3.3 is devoted to review the concepts of transform domain adaptive filtering. The new results of the section are introduced in 1.3.4 where the combined structure is characterized.

1.3.2 Fast sliding transformations

Recently a fast implementation of the recursive Discrete Fourier Transformation DFT has been proposed [6] which combines the idea of polyphase filtering and the Fast Fourier Transformation (FFT) algorithm. Figs. 1.10 and 1.11 show the analyzer and the synthesizer DFT filters, respectively. The operation can be easily understood if we observe that e.g. the analyzer at its input follows the decimation-in-time, while at its output the decimation-in-frequency principle. The computational complexity of these structures is in direct correspondence with that of the FFT and its parallel nature provides additional advantages in parallelization.

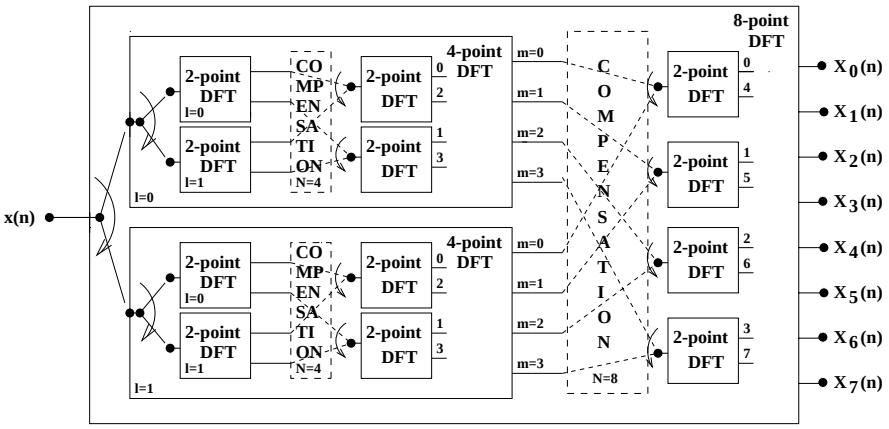


Fig. 1.10 Polyphase DFT analyzer for *N*=8

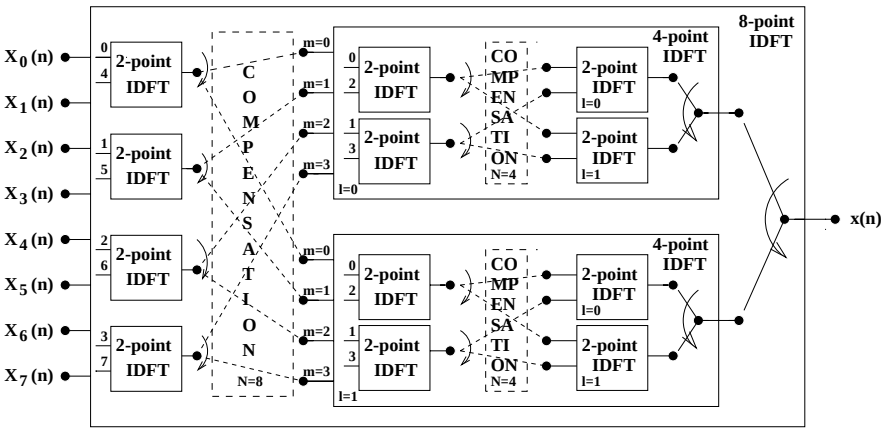


Fig. 1.11 Polyphase DFT synthesizer for *N*=8

The analyzer bank can be operated as a sliding-window DFT or as a block-oriented transformer. The latter one means that the parallel outputs are maximally decimated as it is typical with serial to parallel converters. However, if overlapped data segments are to be transformed, the structure is well suited to support decimation by any integer number. The widely used Overlap-Save Method concatenates two blocks of size *N* to perform linear convolution and calculates 2*N*-point FFTs. A 2*N*-point sliding transformer can easily produce output in every *N*-th step.

In certain applications it may be advantageous to produce signal components instead of the Fourier coefficients as it is dictated by the definition of the DFT. In this case the filter-bank is a set of band-path filters with center frequencies corresponding to the *N*-th roots of unity values. Such DFT filter-banks can easily be derived using the ideas valid for the DFT transformers. With this DFT filter-bank approach, transform-domain signal processing can have the following interpretation: the input signal to be processed first is decomposed by a filter-bank into components and the actual processing is performed on these components. The modified components enter into a *N*-input single output filter-bank (the so-called synthesizer bank) which produces the output sequence.

1.3.3 Transform-domain adaptive filtering [S44]

The concept of transform-domain adaptive filtering offers real advantages if adaptive FIR filters with very long impulse responses are to be handled. The first important aspect is the possible parallelization described above achievable using fast transformation algorithms. The second is the applicability of block adaptive filtering: the parallel channels enable decimation and

therefore a coefficient update only once in every N-th step. In the meantime, however, a much better gradient estimate can be derived.

Fig. 1.9 shows two possible forms of frequency-domain adaptive filtering. In the first version adaptation is controlled by the time-domain difference of the filter output and the desired signal. The adaptive filter performs N multiplications using the N-dimensional weighting vector generated by the adaptive algorithm. From the viewpoint of this section, the adaptation mechanism can be of any kind controlled by an error signal, however, in the majority of the applications the least-mean-square (LMS) algorithm is preferred (see e.g. [13]) for its relative simplicity. Adaptation can be performed in every step however drastic reduction of the computations can be achieved only if the transformer outputs are maximally decimated. The techniques developed for this particular case are the so-called block adaptive filtering methods. [13] gives a very detailed analysis of the most important approaches. It is emphasized that the classical problem of linear versus circular convolution appears also in this context. This problem must be handled because in the majority of the applications a continuous data flow is to be processed and therefore the dependence of the neighboring blocks can not be neglected without consequences. The correct solution is either the Overlap-Save or the Overlap-Add Method. Both require calculations where two subsequent data blocks are to be concatenated and double-sized transformations are to be performed.

If we consider the system of Fig. 1.9a from timing point of view, it is important to observe that at least two transformations must be calculated within the adaptation loop. If real-time requirements are also to be fulfilled, the time needed for these calculations may be a limiting factor. Moreover, if we investigate more thoroughly e.g. the Overlap-Save Method it turns out that the calculation of the proper gradient requires the calculation of two further transformations, i.e. there are altogether four transformations within the loop. This may cause considerable delay and performance degradation especially critical in tracking non-stationary signals.

The adaptation of the transform-domain adaptive filtering scheme on Fig. 1.9b is controlled by an error vector calculated in the transform-domain. Due to this solution the transformer blocks are out of the adaptation loop, therefore the delay within the loop can be kept at a lower level. Here the adaptation is completely parallel and is to be performed separately for every "channel". The operation executed in this scheme corresponds to the circular convolution which may cause performance degradation due to severe aliasing effects.

In the literature of the analysis filter-banks the so-called sub-band adaptive filters see e.g. [13] are suggested for such and similar purposes which provide better aliasing suppression at the prize of smaller $L<N$ decimation rates. The fast sliding transformers are in fact efficiently implemented special filter-banks. If they are maximally decimated they suffer from the side-effects of the circular convolution. In order to reduce aliasing the application of $L=N/2$ can be advised. If the number of the adaptive filter channels is lower than the size of the sliding transformer the standard windowing techniques (see e.g. [1]) can be used for channel-filter design.

1.3.4 Adaptive filtering with fast sliding transformers [S44]

In this subsection the timing conditions of the frequency-domain adaptive filters using the famous Overlap-Save Method are investigated. The block diagram of the method can be followed by Fig. 1.12. The technique is carefully described in [13] therefore here only the most critical elements are emphasized. Other techniques can be analyzed rather similarly.

Fig. 1.13 shows the timing diagram of the standard block-oriented solution. Here the acquisition of one complete data block (N samples) is followed by the processing of this data vector. A continuous sequence of data blocks can be processed if the execution time of one block adaptation is less or equal to the corresponding acquisition time ($t_e \le t_a$). One block adaptation consists of several steps, among them the transformation of the input and the error sequences, respectively (see Fig. 1.9a). The generation of the output sequence requires an additional

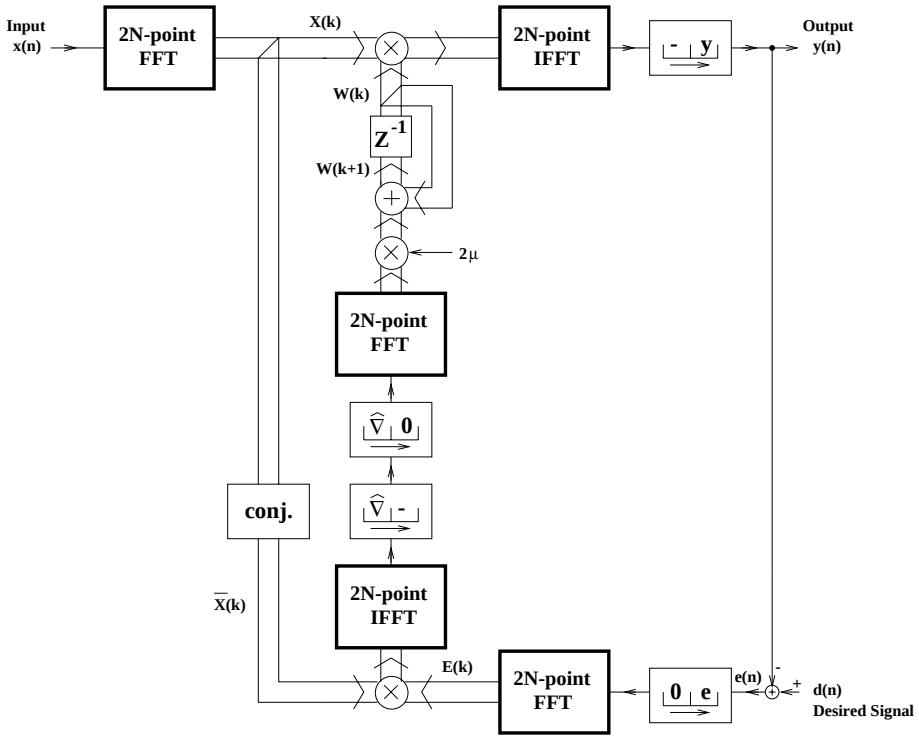


Fig. 1.12 Block diagram of the Overlap-Save Method

transformation. Due to the requirements of the linear convolution all these transformations work on double blocks i.e. on 2*N* data points. The calculation of the gradient and coefficient update requires two further transformations of this type. A detailed analysis of the steps using the 2*N*-point transformations shows that none of them is "complete" i.e. some savings in the computations are possible.

With the introduction of the sliding transformers, the acquisition of the input blocks and the processing can be over-lapped, since the sliding transformers can start working already before

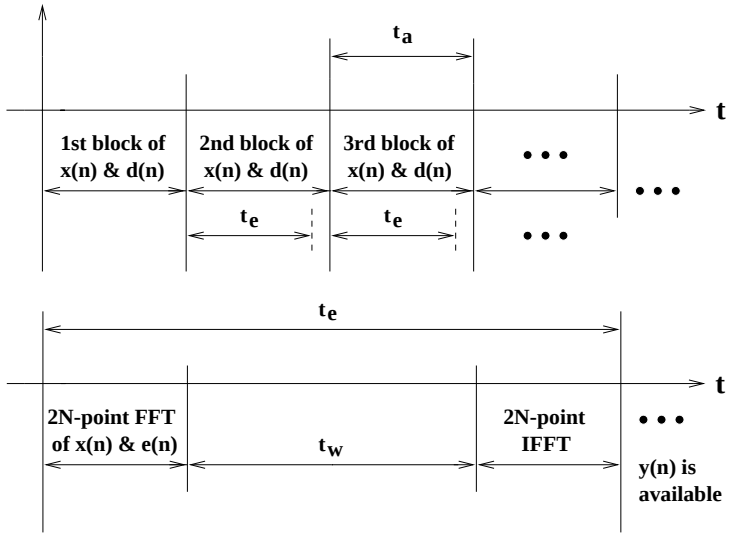


Fig. 1.13 Timing diagram of the standard overlap-save frequency-domain adaptive algorithms (t_a denotes the acquisition time of one data block of *N* samples, t_e stands for the execution time of one block adaptation, and t_w is the calculation time of the gradient and the *W* update). The end-to-end delay is of *N* samples

having the complete block. If we permit an end-to-end delay of 2*N* samples then the processing can be extended for three acquisition intervals (see Fig. 1.14). During the first interval the data acquisition is combined with the transformation, the second can be devoted for finishing the transformation, for updating the coefficient vector and to start the inverse transformation which can be continued in the third interval because the data sampling performed parallel provides the $d(n)$ samples (see Fig. 1.9) sequentially. At the prize of larger end-to-end delay, the execution time can be extended for more blocks, as well.

In the case of the Overlap-Save Method, the gradient calculation consists of an inverse transformation, some simple manipulations and a transformation. Since with the sliding inverse transformation a parallel to serial conversion is performed and the sliding transformer implements a serial to parallel conversion, further overlapping in the execution is possible as it is indicated in Fig. 1.14.

The above considerations can result savings if the granularity of the hardware and software elements of implementation enable smooth distribution of the computational load. If separate hardware units are available for the sliding transformations then the parallelism of the execution can be considerably improved. It is important to note that the precedence conditions of block processing do not support such parallelization: the complete data block must be available for a block-oriented operation like a signal transformation.

To illustrate the achievable gain of using fast transforms here, a comparison is made considering the timing and computational conditions of the two approaches. In the usual implementations of the conventional method, data acquisition and processing are separated in time, i.e. after the arrival of a complete input data block an efficient DSP program calculates the transformed

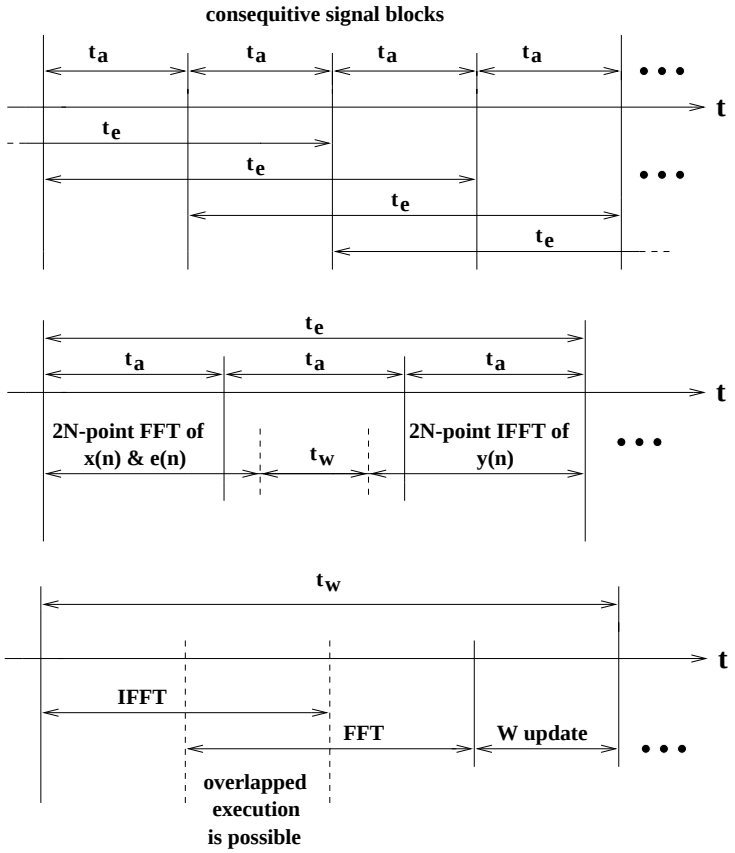


Fig. 1.14 Timing diagrams of a possible frequency-domain adaptive algorithm using sliding transformations. (t_a denotes the acquisition time of one data block, t_e stands for the execution time of one block adaptation, and t_w is the calculation time of the gradient and the W update). The end-to-end delay is of 2*N* samples

values. The sliding FFT introduced in [6] has the same computational complexity as the traditional algorithm. However, data acquisition and processing overlap in the proposed polyphase filter-bank, therefore calculations start well before the complete block becomes available. There are many ways to characterize the complexity of the FFT, one possible figure can be the number of complex multiplications. This figure for the maximally decimated sliding N-point FFT equals (see [6])

$$C1 = \frac{N}{4}\log_2\left(\frac{N}{8}\right)+1, \tag{1.5}$$

i.e. this figure is one of the possible characterizations. These operations, however, are performed partly during the data acquisition, and the number of complex multiplications, which can not be executed before the arrival of the last sample of the block remains only

$$C2 = \frac{N}{2} - \log_2(N). \tag{1.6}$$

As an example, for N=1024 $C1$=1793 and $C2$=502, i.e. the possible time gain due to the overlap can be considerable. Similar figures can be given for other operations, as well.
As a summation of the above investigations we can state that with such techniques further parallelism can be achieved and utilized for applications where higher sampling rates are required. The solutions which follow the scheme of Fig. 1.9b are in good correspondence with some successful multirate filter-bank techniques while the other group related to Fig. 1.9a can significantly be improved. The prize to be paid for the additional parallelism is the increase of the overall end-to-end delay which requires further investigations concerning step-size and stability issues. This can be started following the ideas of [15], where a similar problem was to be solved.
In this section, the term “transform-domain” was used instead of the explicit term “frequency-domain” The reason for this is that all the above developments can be extended for other type of transformations, as well. The application of other transformers may reduce either the computational load or in certain cases they improve the adaptation performance.

1.4 Anytime Fourier Transformation

Anytime signal processing algorithms are to improve the overall performance of larger scale embedded digital signal processing (DSP) systems. The early availability of the amplitude and/or frequency components of digital signals can be very important in different signal processing tasks, where the processing is done on-line, parallel with measurements and input data acquisition. It may offer a possible way for increasing the sampling rate (or the complexity of the tasks to be solved during one sampling period) and also to decrease the delay caused by the necessary information collection for setting the measurement/signal processing scheme.
In this section the concept of anytime Fourier transformation is presented and a new fast anytime fuzzy Fourier transformation algorithm is introduced. The method reduces the delay problem caused by the block-oriented fast algorithms and at the same time keeps the computational complexity on relatively low level. It yields partial results of good quality or estimates before the samples of the period arrive. This is especially advantageous in case of abrupt reaction need and long or possibly infinite input data sequences. As a possible application field, the usage of the presented new method in Adaptive Fourier Analysis of multisine signals is also investigated.
The determination of the frequencies of a multisine signal can be very important at different signal processing tasks, like vibration measurements and active noise control related to rotating machinery and calibration equipment. Adaptive Fourier Analyzers have been developed for

measuring periodic signals with unknown or changing fundamental frequency. Higher frequency applications have limitations since the computational complexity of these analyzers are relatively high as the number of harmonic components to be measured (or suppressed) is usually above 50.
Recently, a fast-filter bank structure has been proposed for Adaptive Fourier Analysis based on the combination of the concept of transform domain signal processing and the adaptation of a simple linear combiner. It results in the reduction of the above computational complexity, however for the correct use we have to have pre-estimation about the range of the fundamental frequency to be able to set the applied single-input multiple-output filter-banks, which in many cases causes significant and possibly non-tolerable delay in the operation.
In this section, a new fast fuzzy logic supported anytime frequency range estimation procedure is also proposed which makes possible to execute the frequency estimation after one quarter of the period of the unknown signal, i.e. the adaptation and Fourier analysis can be performed without any delay.

1.4.1 Introduction

Computer-based monitoring and diagnostic systems are designed to handle abrupt changes due to failures within the supervised system or in its environment. This capability involves on one hand different, simultaneously operated digital signal processors (DSPs), while on the other the corresponding information processing algorithms. These algorithms should be performed under prescribed response time conditions.
Block-oriented signal processing techniques have exceptional role in time critical signal processing applications due to the availability of fast algorithms. However, if larger data segments are to be evaluated in real-time, the delay caused by the block-oriented approach is not always tolerable especially if the response time of our evaluating system is also specified. This can be exceptionally critical if the signal processing is related to feedback loops. The introduction of anytime techniques can help overcoming the problem.
In this section block-oriented signal processing methods are combined with recursive ones thereby resulting in anytime signal processing transformation algorithms. This combination reduces the delay problem caused by the block-oriented fast algorithms and at the same time keeps the computational complexity on relatively low level. The proposed technique makes possible not only the application of fast algorithms in sharp time requirement conditions but also the availability of partial results or estimates in case of long or possibly infinite input data sequences. The latter is very advantageous if pre-considerations based on some features of the signal to be analyzed are needed for the further processing (or for the setting of the processing equipment). A typical example of this case is adaptive Fourier analysis, The early, approximate results can help in starting the processing earlier and to reduce the not always tolerable side-effects of processing delay.
The section is organized as follows: In Subsection 1.4.2 the novel concepts of block-recursive averagers are detailed. Subsection 1.4.3 discusses how the block-recursive averagers can be used in anytime Fourier transformation. Subsection 1.4.4 summarizes the improvement of the anytime Fourier analysis scheme by introducing fuzzy techniques in the interpretation of the results. The next section is devoted to adaptive Fourier analysis, where it is shown that the new methods described in Subsections 1.4.3 and 1.4.4 are suitable to decrease the delay of the adaptive Fourier analysis. In Subsection 1.4.5 illustrative examples are shown.

1.4.2 The novel concept of block recursive averaging ([S8], [S45])

In this section the standard algorithms for recursive averaging are extended for data-blocks as single elements. To illustrate the key steps, first the block-recursive linear averaging will be introduced. For an input sequence x(n), n=1,2, ..., the recursive linear averaging can be expressed as

$$y(n) = \frac{n-1}{n} y(n-1) + \frac{1}{n} x(n-1) \tag{1.7}$$

For $n \geq N$ the "block-oriented" linear averaging has the form of

$$X(n-N) = \frac{1}{N} \sum_{k=1}^{N} x(n-k) \tag{1.8}$$

while the block-recursive average can be written as

$$y(n) = \frac{n-N}{n} y(n-N) + \frac{N}{n} X(n-N) \tag{1.9}$$

If (1.9) is evaluated only in every N-th step, i.e. it is maximally decimated, then we can replace (1.9) with $n=mN$, $m=1,2, ...$, by

$$y(mN) = \frac{m-1}{m} y[(m-1)N] + \frac{1}{m} X[(m-1)N] \tag{1.10}$$

or simply

$$y(m) = \frac{m-1}{m} y(m-1) + \frac{1}{m} X(m-1) \tag{1.11}$$

where m stands as block identifier. Note the formal correspondence with (1.6).
If the block identifier m in equation (1.11) is replaced by a constant $Q > 1$ then an exponential averaging effect is achieved. This change makes the above block-oriented filter time-invariant and thus a frequency-domain characterization is also possible.
In many practical applications exponential averaging provides the best compromise if both the noise reduction and the signal tracking capabilities are important. This is valid in our case as well, however, in this section only the linear and the sliding averagers are investigated because they can be used directly to extend the size of certain signal transformation channels and they can be applied in anytime systems.
A similar development can be provided for the sliding-window averagers. The recursive form of this algorithm is given for a block size of N by

$$y(n) = y(n-1) + \frac{1}{N}[x(n-1) - x(n-N-1)] \tag{1.12}$$

If in (1.12) the input samples are replaced by preprocessed data, e.g. as in (1.8), then a block-recursive form is also possible:

$$y(n) = y(n-N) + [X(n-N) - X(n-2N)] \tag{1.13}$$

which, however, has no practical meaning, since it gives back (1.8). But if the window size is integer multiple of N, e.g. MN, then the form

$$y(n) = y(n-N) + \frac{1}{M}[X(n-N) - X(n-(M+1)N) \tag{1.14}$$

has real importance. If (1.14) is evaluated only in every N-th step, i.e. it is maximally decimated, then we can replace (1.14) with $n=mN$, $m=1,2, ...$, by

$$y(mN) = y[(m-1)N] + \frac{1}{M}[X((m-1)N) - X((m-M-1)N)] \tag{1.15}$$

or simply

$$y(m) = y(m-1) + \frac{1}{M}[X(m-1) - X(m-M-1)] \tag{1.16}$$

where *m* stands as block identifier. Note the formal analogy to (1.12).
The generalization of these averaging schemes to signal transformations and/or filter-banks is straightforward. Only (1.8) should be replaced by the corresponding "block-oriented" operation. Fig. 1.15 shows the block diagram of the linear averaging scheme. This is valid also for the exponential averaging except *m* must be replaced by *Q*. In Fig. 1.16 the sliding window averager is presented. These frameworks can incorporate a variety of possible transformations and corresponding filter-banks which permit decimation by the block-size. Standard references, e.g. [7] provide the necessary theoretical and practical background.

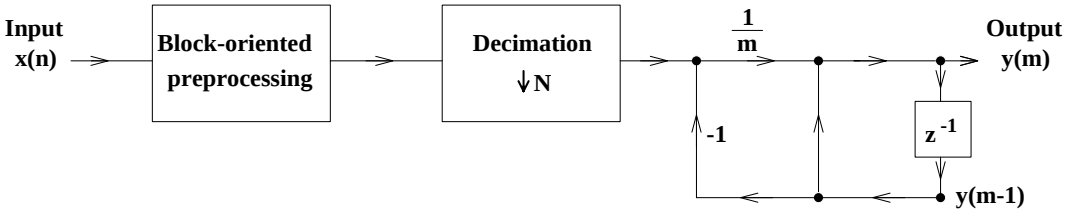


Fig. 1.15 Block-recursive linear averaging signal processing scheme, *n=mN*

The idea of transform-domain signal processing proved to be very efficient especially in adaptive filtering (see e.g. [13]). The contribution of this section is directly applicable for the majority of these intensively cited algorithms. The most important practical advantage here compared to other methods is the early availability of rough estimates which can orientate in making decisions concerning further processing.

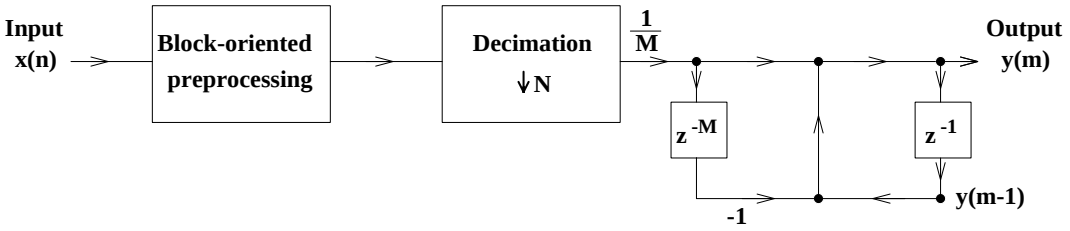


Fig. 1.16 Block-recursive sliding-window averaging scheme, *n=mN*, window-size *MN*

The multiple-block sliding-window technique can be mentioned as a very characteristic algorithm of the proposed family. For this the computational complexity figures are also advantageous. Using conventional methods to evaluate in "block-sliding-window" mode the transform of a block of *MN* samples would require *M* times an (*MN*)*(*MN*) transformation, while the block-recursive solution calculates only for the last input block of *N* samples, i.e. *M* times an (*MN*)*(*N*) "transformation".

1.4.3 The new Anytime Fast Fourier Transformation algorithm (AnDFT, AnFFT) ([S8], [S110], [S120], [S112])

As block-oriented preprocessing the DFT is the most widely used transformation for its fast algorithms (FFTs) and relatively easy interpretation. The above schemes can be operated for every "channel" of the DFT and after averaging this will correspond to the channel of a larger

scale DFT. If linear averager is applied, this scale equals mN while for sliding averager this figure is MN. The number of channels obviously remains N unless further parallel DFTs are applied.

These additional DFTs have to locate their channel to the positions not covered by the existing channels. For the case where M=2 (i.e. only one additional parallel DFT is needed), this positioning can be solved by the so-called complementary DFT which is generated using the N-th roots of -1. This DFT locates its channels into the positions π/N, $3\pi/N$, etc. For M>2 proper frequency transposition techniques must be applied. If e.g. M=4 then the full DFT will be of size $4N$ and four N-point DFTs (working on complex data) are to be used (Fig. 1.17). The first DFT is responsible for the channels in positions 0, $8\pi/4N$, etc. The second DFT should cover the $2\pi/4N$, $10\pi/4N$, etc., the third the $4\pi/4N$, $12\pi/4N$, etc, and finally the fourth the $6\pi/4N$, $14\pi/4N$, etc. positions, respectively (Fig. 1.18). The first DFT does not need extra frequency transposition. The second and the fourth process complex input data coming from a complex modulator which multiplies the input samples by $e^{j2\pi n/4N}$ and $e^{j6\pi n/4N}$, respectively. The third DFT should be a complementary DFT.

It is obvious from the above development that if a full DFT is required the sliding-window DFT must be preferred otherwise the number of the parallel channels should grow with m.

Here we would like to remark that with appropriate frequency transposition, this scheme can be further extended theoretically: more than 4 FFT blocks may run parallel, however with a burden of higher complexity.

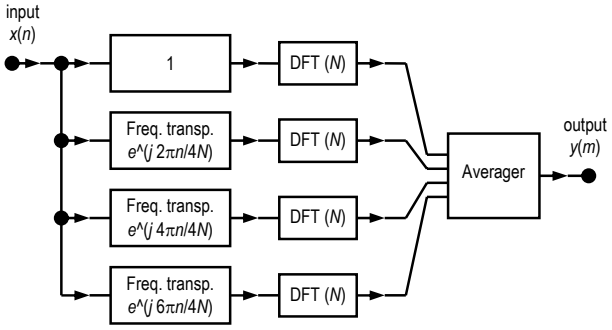


Fig. 1.17 Block diagram of the anytime FFT scheme

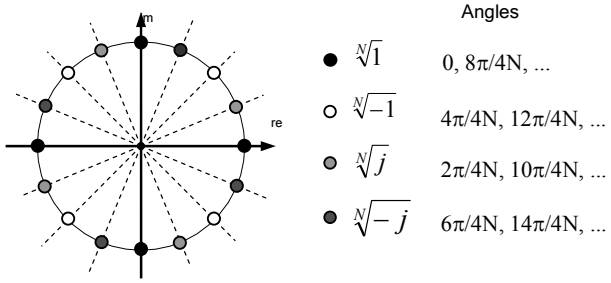


Fig. 1.18 The poles of a ($4N$)-point DFT composed of four N-point DFTs

The majority of the transform-domain signal processing methods prefers the DFT to other possible transformations. However, there are certain applications where other orthogonal transformations can also be utilized possibly with much better overall performance. A further aspect of practical interest can be the end-to-end delay of the block-oriented processing. The time-recursive transformation algorithms described e.g. in [16] are sliding-window transformations, i.e. they are filter-banks providing transform domain representation of the last input data block in every step. Decimation is not "inherent" as it is the case if the transformation is considered as a serial to parallel conversion, therefore the processing rate can be the input rate,

the maximally decimated one, or any other in between. These techniques are not fast algorithms, however, they "produce" less delay as those block-oriented algorithms which start working only after the arrival of the complete input data block.

1.4.4 Anytime Fuzzy Fast Fourier Transformation (FAnFFT) ([S41])

We have developed a new interpretation method of the non-exact estimations of the anytime results. According to it, the noisy frequency characteristics is viewed as a fuzzy set over the Universe of Frequencies. Before the defuzzification we first evaluate the α-cut of the fuzzy set based on a properly chosen α value. This value serves to determine the limit separating the "useful" signals from what is interpreted as noise. In the obtained α-cut, the separated "picks" are handled and defuzzificated separately since each pick, as an individual fuzzy set, represents the frequency of a signal component. As defuzzification, the indexed Center of Gravity (iCoG) defuzzification method is applied based on the chosen α-cut of the output.
Since the most typical errors, the picket fence and the leakage cause symmetrical error around the accurate value, the applied fuzzy defuzzification method results in high accuracy. This is because instead of taking the non-accurate values as exact ones, thus bringing error into the interpretation, we apply value imprecisiation which in reality means 'meaning precisiation' [17].

1.4.5 Decreasing the delay of Adaptive Fourier Analysis based on FAnFFT ([S14], [S38], [S42], [S51])

Recently, the measurement of periodic signals with unknown or changing fundamental frequencies has come into the focus. It has a significant role in such application fields like vibration analysis, active noise control, rotating machinery, calibration equipment, and the autonomous analysis of signals and noises. If we want to measure the unknown fundamental frequency of a periodic signal or the frequency components of multi-rate signals, the classical solution is to use a Fourier transformer however we have to apply some kind of synchronization between the (unknown) frequencies and the sampling frequency of the Fourier transformer. Otherwise, we face the problems of picket-fence effect and leakage.
The earlier solutions can be classified into two groups. The first applies re-sampling by the estimate of the fundamental frequency, i.e. after an interpolation, it applies re-sampling which is followed by an FFT (see e.g. [18]).
The other group uses a so called Adaptive Fourier Analyzer (AFA) filter-bank structure originally proposed by Nagy in [19]. The adaptive filter-bank can be tuned to each signal component. This adaptation procedure "locks" the fundamental frequency component of the periodic signals like a PLL and tunes the recursive DFT channels accordingly.
The method implements a structurally adaptive system with an order depending on the actual ratio of the fundamental to the sampling frequencies. The actual number of the channels $N(n)$ should meet the condition of

$$[N(n) - 1]f_1(n) < f_s < [N(n) + 1]f_1(n), \qquad (1.17)$$

where $f_1(n)$ and f_s denote the fundamental frequency at time instant n and the sampling frequency, respectively. This results in applying, always, as many DFT channels as can be accommodated within the frequency range up to one half of the sampling frequency.
Besides many advantages, the disadvantage of the AFA is that fast transformations cannot directly be utilized.
The author of this thesis introduced a special implementation of the AFA structure in [S14], which is a filter-bank version of a fast transformation (e.g. FFT) applying adaptive linear combiners at the output of the transformer (Fig.1.19). This opens the possibility for block oriented operation, i.e. the computational complexity of this latter solution is much lower than that of the previous one.

The consequence of the possible decimation is that the condition in (1.17) has to be replaced by a stronger requirement. If the adaptation is done in every k-th sampling step then the following requirement has to be met

$$\frac{f_s}{2k} < f_1(n) < \frac{3f_s}{2k} \quad . \tag{1.18}$$

In case of maximum decimation, i.e. when $k=N$, it takes the form of

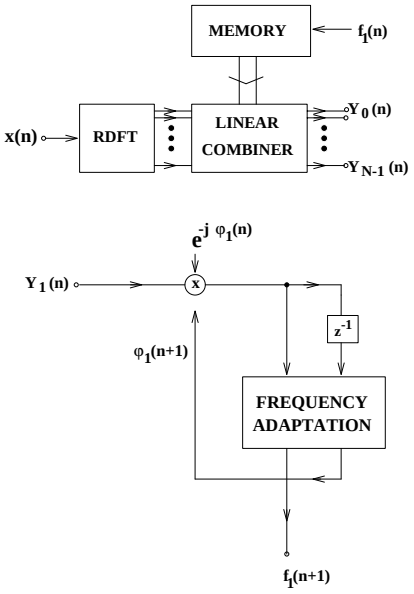


Fig. 1.19 Block diagram of the Adaptive Fourier Analysis

$$\frac{f_s}{2N} < f_1(n) < \frac{3f_s}{2N} . \tag{1.19}$$

For more details about the AFA see [S14].
As a consequence of (1.18) and (1.19), we have to pre-estimate the range of the frequencies to be measured before starting with the frequency estimation, to be able to set the AFA structure properly. Normally, this may cause a significant delay in the procedure which cannot always be tolerated, especially in real-time processing.
The fast anytime Fourier transformation methods presented in the last two subsections are excellent tools for obtaining pre-estimates of the frequencies of multisine signals to be analyzed. It only needs the operation of the FAnFFT or AnFFT algorithm in the first part of the period. Based on the frequency estimations, the filter-banks of the AFA scheme can properly be set.

1.4.6 Illustrative examples

In the followings three simple examples are presented illustrating the usability of the results. For more experiments and examples, see ([S8], [S14], [S35], [S45], [S38], [S42], [S51], [S110], [S120], [S112]).
In the first example (AnFFT method, Subsection 1.4.2) a 256-channel DFT is calculated recursively, in anytime mode with N=64 for m=1,2,3,4. The input sequence applied is

$$x(n) = \cos\left(\frac{\pi(N+0.5)n}{2N}\right). \tag{1.20}$$

This single sinusoid is just in the middle between two measuring channels. The MATLAB simulations after processing the first, second, etc. blocks are given in Fig. 1.20.

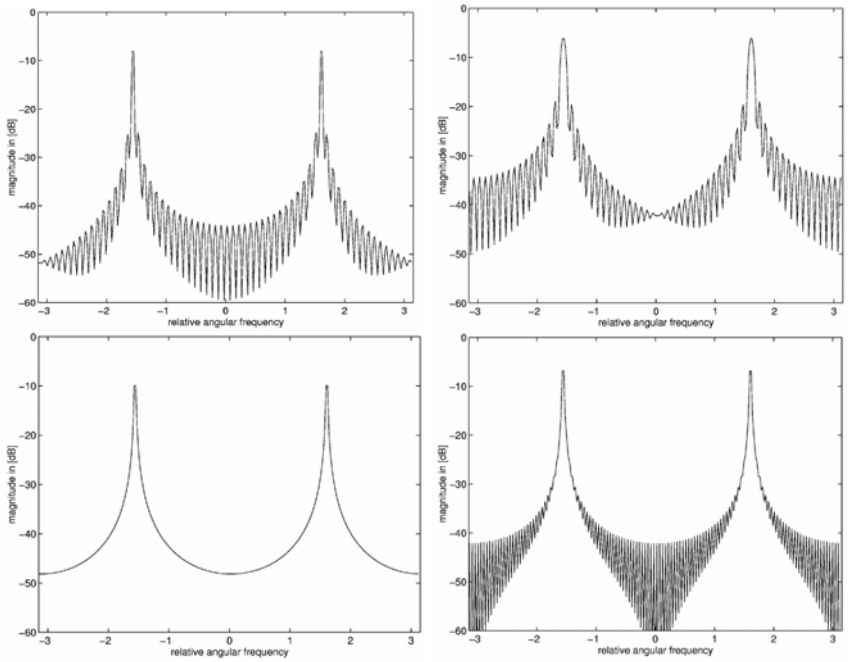


Fig. 1.20 Ex. 1: 256-channel anytime DFT of a single sinusoid in the middle between two measuring channels. N=64, m=1, 2, 3, 4

In the second example (AnFFT method, Subsection 1.4.2) a 256-channel DFT is calculated recursively, in anytime mode with N=64 for m=1,2,8,16. The input sequence is

$$x(n) = \cos\left(\frac{\pi n}{2}\right) + rand - 0.5 \tag{1.21}$$

where *rand* stands for a random number generated by MATLAB between 0 and 1. The sinusoid is located exactly to a DFT channel position. The simulation results for m=1,2,8, and 16 are given in Fig. 1.21. The improvement in resolution and noise reduction is remarkable.
The third example is illustration for the improvement achieved by the fuzzy techniques (see also FAnFFT method in Subsection 1.4.3). The FAnDFT is applied on a noisy multi-sine signal. The 256-channel DFT of two sinusoids, exactly at the DFT channel positions, corrupted by noise is calculated according to the presented anytime method recursively with N=64 for m=1, 2, 8, 16. The input sequence is

$$x(n) = \sin\left(\frac{2\pi 50n}{4N}\right) + 0.8\sin\left(\frac{2\pi 112n}{4N}\right) + \text{rand} - 0.5 \cdot \tag{1.22}$$

The simulation results for m=1, 2, 8, and 16 are given in Fig. 1.22. Fig. 1.23 shows the α-cuts with α=-17 dB for m=1, 2, 8, 16. Table I summarizes the obtained frequencies evaluated by α = -17 dB. In Fig. 1.24 the convergence of the non-zero amplitude components can be followed if (a) the non-noisy and (b) the noisy signals are processed. The improvement in both the resolution and noise reduction is remarkable.
Here we would like to remark that if value α is chosen too small then false picks caused by the noise may also appear in the spectrum. At α=-25 dB e.g., in case of m=1 and 2 we obtain 10-11 picks (of which 6-7 come of noise) however as the approximation becomes more accurate along the time, at m=8 and 16, the false picks disappear and only the 4 "useful" frequencies remain in the spectrum.

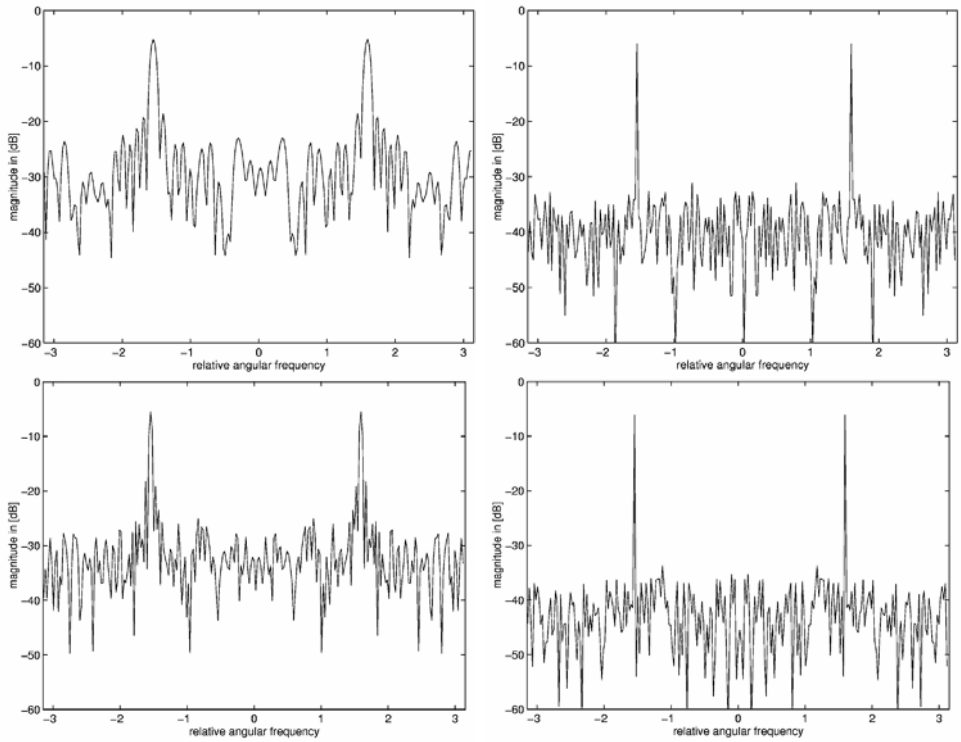


Fig. 1.21 Ex. 2: 256-channel anytime DFT of a single sinusoid, exactly at a DFT channel, plus noise. N=64, m=1, 2, 8, 16.

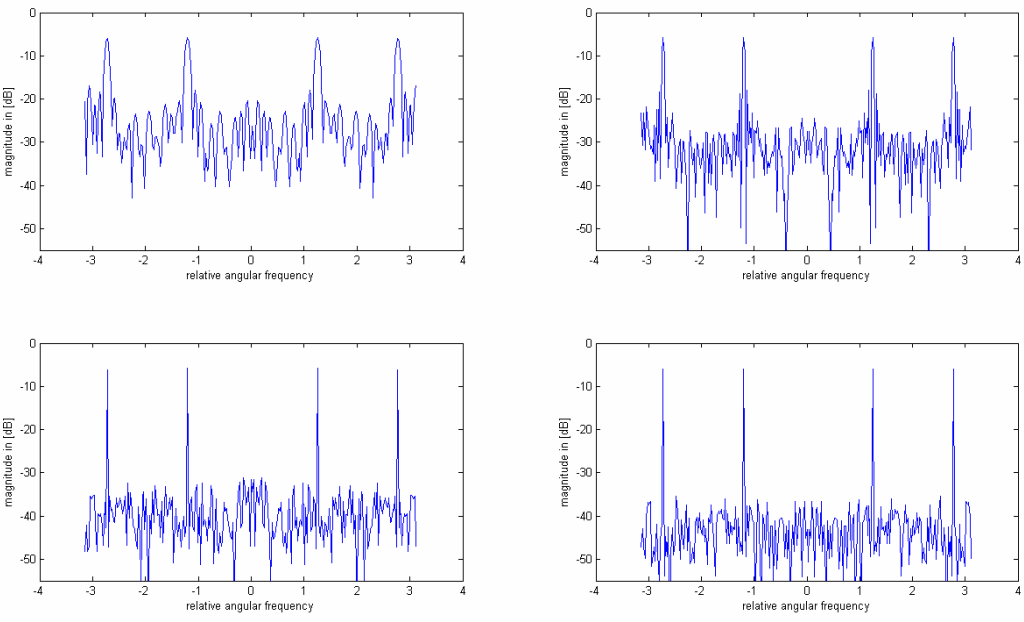


Fig. 1.22 Ex. 3: 256-channel anytime DFT of a noisy multi-sine, exactly at a DFT channel. N=64, m=1, 2, 8, 16.

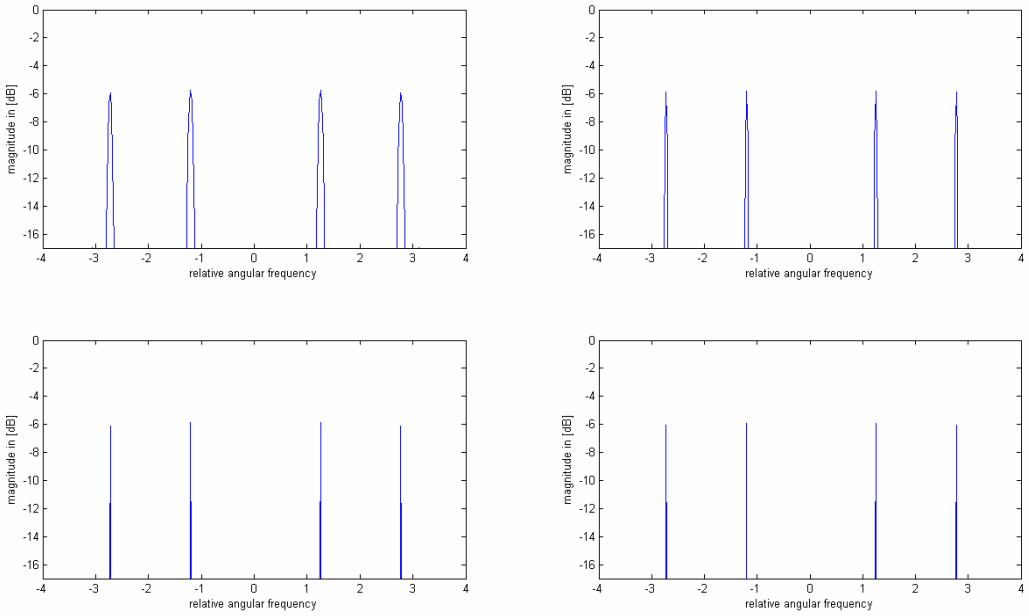


Fig. 1.23 Ex. 3: α-cuts with α=-17 dB of the 256-channel anytime DFT of a noisy multi-sine, exactly at a DFT channel. N=64, m=1, 2, 8, 16.

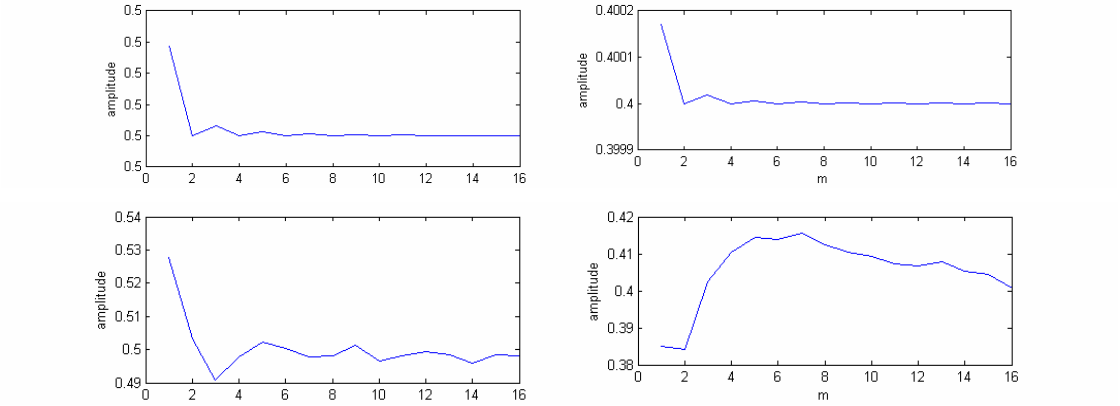


Fig. 1.24 Convergence of the amplitude of the two non-zero multisine components: the non-noisy signal in ex. 3 (upper) and ex. 3 (with noise) (lower)

Table 1.1
Ex. 3: Obtained Frequencies for m = 1, 2, 8, and 16, α=-17 dB

	obtained frequencies			
m=1	-2.7515	-1.2268	1.2267	2.7515
m=2	-2.7490	-1.2275	1.2275	2.7490
m=8	-2.7517	-1.2267	1.2267	2.7517
m=16	-2.7488	-1.2271	1.2271	2.7488

1.5 Overcomplete signal representations

For representing stationary signals, several well-established methods are available. For non-stationary signals, however, these approaches can be used only with serious limitations. If the

signal can be characterized as sequence of stationary intervals overcomplete signal representations help to handle such problems.
This section introduces the concept of recursive overcomplete representations using different recursive signal processing algorithms. The novelty of the approach is that an on-going set of signal transformations together with appropriate (e.g., L_1 norm) minimization procedures can provide optimal on-going representations, on-going signal segmentations into stationary intervals, and on-going feature extractions for immediate utilization in diagnostics, or other applications. The proposed technique may be advantageous in case of processing non-stationary signals when complete signal representations can be used only with serious limitations because of their relative weakness in adaptive matching of signal structures.

1.5.1 Introduction

The standard methods for representing stationary (non-time-varying) signals include techniques utilizing special signal representations based on dedicated signal transformations. The trigonometric transformations are typical examples. For non-stationary signals however, the applicability of these approaches is limited. Sliding-window transformation techniques may offer a good compromise, but the window-size limits the resolution.
The so-called overcomplete signal representation is a technique where a given signal is decomposed using more basis components than the necessary minimum. Obviously, without further constraints, this representation is not unique. Having an accurate model, a reasonable metric for further evaluation is the compaction of the representation that the model provides. This means that solutions with the minimum number of non-zero coefficients are required. If a representation is both accurate and compact, then it will capture the "principal components" of the signal behavior. It is an important result that by minimizing the L_1 norm of this representation, optimal solutions can be obtained [20], [21].
A compact representation is useful for compression. If all the basis vectors have the same norm, a good approximation of the signal can be generated using only those components that have significant weights. The negligible components can be thresholded (i.e. set to zero) without substantially degrading the signal reconstruction. It is obvious that representations where the coefficients are all of similar value, thresholding is not acceptable and compaction cannot be readily achieved.
An additional aspect is that compression and denoising are linked. A white noise sequence is essentially incompressible even if it is transformed by an orthogonal transform. If a deterministic signal degraded by additive noise is compressed, this representation will extract the primary structure of the signal, and the reconstruction based on such a compact model will be a denoised or enhanced version of the original.
With the exception of overcompleteness, all the above considerations, compactness, accuracy, and optimum behavior with respect to data compression are taken into account with the Karhunen-Loeve transformation (see [22]). However, this technique requires intensive calculations hardly performable in real time applications.
In this section, a relatively simple technique is introduced to provide an approximate solution to the compact representation/compression problem for the case of non-stationary signals. The method is based on recursive signal transformers (see, e.g., [S6] and [5]) running in parallel, and providing an on-going overcomplete signal representation (i.e. the signal processing is running parallel with the sampling (data collection)). The novelty of the approach is that an on-going set of signal transformations together with appropriate (e.g., L_1 norm) minimization procedures can provide nearly optimal on-going representations and support on-going signal segmentation into stationary intervals, together with feature extraction for immediate utilization in diagnostics, or other applications.
Subsection 1.5.2 gives an overview of overcomplete signal representation. Subsection 1.5.3 describes the proposed new method, while Section 1.5.4 is devoted to an illustrative example.

1.5.2 Overcomplete signal representation

In transformed-domain digital signal processing, the traditional method is to represent a signal using a discrete, preferably orthogonal basis, like the discrete Fourier (DFT), the discrete cosine (DCT), the Hadamard, Wavelet, etc. bases. Each transformation uses one complete basis (for a vector space V of dimension N, the complete basis $\boldsymbol{B}$ contains exactly N basis vectors, $\underline{b}_0, \underline{b}_1, \ldots, \underline{b}_{N-1}$) and the representation is unique. The coefficients of an expansion can be derived using an inverse matrix computation. Well-established theories and fast transformation algorithms help the applications. The disadvantage of such a representation is its relative weakness in adaptive matching of signal structures.

Recently a new method [22], the overcomplete signal representation (OSR), has been reported which is an adaptive signal representation method having special advantages in cases of non-stationary signals. Here the signals are decomposed onto a number of optimal basis components that are found from an overcomplete basis dictionary via some optimization method [22] (i.e. the representation is not unique). The overcomplete dictionary consists of a collection of bases, and the applied basis for a given signal expansion is chosen from the set of bases according to a metric. This means that the basis is redundant (e.g. some extra basis components are added to a complete basis or several complete bases are merged), for a vector space V of dimension N the overcomplete basis $\boldsymbol{D}$ contains L basis vectors, $\underline{b}_0, \underline{b}_1, \ldots, \underline{b}_{L-1}$ where $L>N$. The expansion means that a nonzero solution is found for the following equation

$$\boldsymbol{x} = \boldsymbol{D}\underline{\alpha}, \tag{1.23}$$

or in case of approximate decomposition

$$\mathbf{x} = \mathbf{D}\,\tilde{\alpha} + r, \tag{1.24}$$

where $\boldsymbol{x}=[x_0, x_1, \ldots, x_{N-1}]$ stands for the input signal, $\underline{\alpha}=[\alpha_0, \alpha_1, \ldots, \alpha_{L-1}]^T$, $\tilde{\underline{\alpha}}=[\tilde{\alpha}_0, \tilde{\alpha}_1, \ldots, \tilde{\alpha}_{L-1}]^T$ denote the coefficient (basis component selection) vector, $\boldsymbol{D}=[\underline{b}_0, \underline{b}_1, \ldots, \underline{b}_{L-1}]$ denotes the overcomplete basis dictionary, and $\underline{r}$ is the error vector.

The expansion functions (from $\boldsymbol{D}$) are chosen in a signal-adaptive fashion and the algorithms for choosing the functions are decidedly nonlinear in both cases. There are many different possible overcomplete expansions corresponding to the different metrics and methods used as the criteria of optimum. This leads to signal adaptivity and compact representations with a burden of additional computations, since the coefficients have to be determined by using some optimization task, like singular value decomposition, different norm optimizations, method of frame bounds, etc. The properties of adaptively tracking or matching the varying structure of a given non-stationary signal depend also on the content and the number of components of the overcomplete basis dictionary, so a balance has to be found between the computational costs and performance level.

The first class of the optimization methods (1.23) results in exact solutions. Best basis methods are typical examples to this. One of the most promising methods among the numerous possibilities is the basis pursuit (BP) method proposed by Chen and Donoho [23]. The idea of this method is to define the basis selection vector by minimizing the L_1-norm of $\underline{\alpha}$ under the constraint of $\boldsymbol{x}=\boldsymbol{D}\underline{\alpha}$ which leads to the minimum fuel problem (see e.g. [24]). Minimizing $\|\underline{\alpha}\|_1$ ensures finding the least number of basis components from an overcomplete basis dictionary to exactly represent a given signal. In this section, the concept of the BP method is used to select from a predefined set of transformations the most compact one, and to improve its compactness by replacing one of its basis vectors with another providing good approximation of the signal in hand.

1.5.3 Recursive Overcomplete Signal Representation and Compression ([S19], [S66])

In this section a relatively simple technique is introduced to provide an approximate solution to the compact representation/compression problem for the case of non-stationary signals. The overcomplete signal representation method is based on recursive signal transformers running in parallel, and on an appropriate minimization procedure which ensures near optimal on-going signal representation. It also supports on-going signal segmentation into stationary intervals, together with feature extraction for immediate utilization in diagnostics, or other applications.
The transform library contains several normalized, complete, orthogonal or "weakly" overcomplete transformations (using more bases means on one hand that the adaptivity increases, while on the other hand, that the additional computational needs of the representation also increase compared to methods based on less bases or on one complete basis). "Weakly" overcomplete transformation means here a transformation where a complete, preferably orthogonal, transform is merged with one "extra" basis function.
For optimum-criteria the L_1 norm minimum is chosen which provides optimal solution, i.e., the representation having the minimum L_1 norm contains the minimum number of non-zero coefficients.

Method 1.1 ([S19], [S66])

The steps of the algorithm are as follows (see Fig. 1.25):

Coding:

Step 1. *Parallel transformations:* The incoming signal is processed in blocks and to each of the blocks the best fitting transformation is chosen from the transform library. The block length corresponds to the transformation sizes, i.e. if the transformations are N-point transformations than the input signal blocks will include N samples. The input signal blocks are transformed into the different transform-domains of the transform library. If the transformation is a "weakly" overcomplete one (having N+1 basis functions) then the basis dictionary is to be reduced appropriately.

Theorem 1.1. ([S19], [S66])
The input signal blocks can always be transformed into the possible most compact basis representation of the weakly overcomplete transform library.

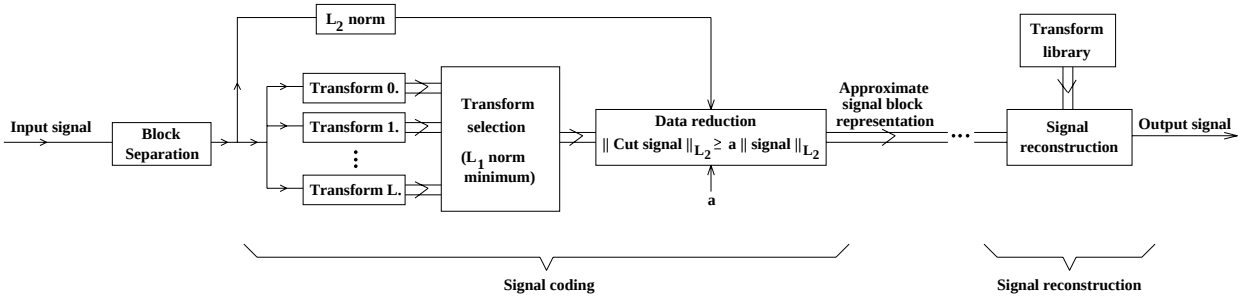


Fig. 1.25 Block diagram of the signal coding and signal reconstruction for the proposed OSR model

Proof:

As first step, the input signal block is transformed into the starting, complete basis ([X_0, X_1, … , X_{N-1}]). The additional "extra" basis function is also expressed by the same basis ([Y_0, Y_1,…, Y_{N-1}]). If the "extra" basis function is taken into consideration by an unknown weight factor c then the overcomplete representation of the input signal will take the following form: [X_0-cY$_0$, X_1-cY$_1$, … , X_{N-1}-cY$_{N-1}$, c]. Since we know that the L_1 norm minimum ensures the minimum number of

non-zero coefficients, at least one of the above N+1 coefficients has to be zero which means that according to the L_1 norm minimum criteria the value of c is of $\{0, X_0/Y_0, X_1/Y_1, \dots, X_{N-1}/Y_{N-1}\}$. Substituting c with these values, the one which has the minimum L_1 norm provides the most compact representation and we can drop out the corresponding, zero-weighted basis function from the basis dictionary. This results in a complete, not necessarily orthogonal transform-basis.

Step2. *Transform selection:* The optimal representation for the given input block is determined by the L_1 norm minimum of the input signal, i.e., the transformation having the smallest L_1 norm contains the minimum number of non-zero coefficients, thus, the given signal block will be represented in this transform-domain and this transform gives the basis for the signal coding of the block.

Step 3. *Data reduction:* The "principal components" of the "winner" are selected, i.e., the basis functions having the most significant weights. The accuracy of the approximation is ensured by using an L_2 norm bound, i.e., the zero-valued or less significant basis components can be dropped while it holds that the L_2 norm of the approximate signal exceeds an appropriate ratio (a) of the L_2 norm of the original input signal represented in the time-domain.

Step 4. *Signal coding:* The remaining coefficients and the identifier of the optimal or near optimal transformation are coded in a suitable form.

Reconstruction:

Signal reconstruction: Using the coded information the approximate signal can be reconstructed.

1.5.4 Illustrative examples

In Figs. 1.25-1.30, simple examples are presented to illustrate the proposed OSR method. For more examples and details, see [S19], [S66]. Theoretically, the more complete and weakly overcomplete transformations in the transform library we use, the more increased adaptivity we get, however, with a burden of higher additional computational need. Using bigger transformation (and block) size means on one hand that the code segments will be longer, however, the transferred information amount may be altogether less. On the other hand, transforming longer input signal blocks has negative effect on the adaptivity property of the method, i.e., the adaptivity decreases. This may cause serious limitations especially in case of representing highly non-stationary signals. Thus, in general, smaller transformation sizes may perform better, however, we have to find a compromise for it.
In the illustrative examples the applied transform library contains 8 transformations of size 16 (see Table 1.2). Besides some "pure" transformations (T1-T4: Hadamard, DFT, DCT, Haar transformations) the following overcomplete transformations are taken into consideration: "pure" transformations merged with the "principal" basis function of another transformation (T5-T6: Hadamard basis + 1 DCT basis function, DFT basis + 1 Hadamard basis function) and "pure" transformations merged with the previous block of the approximate signal (T7-T8: Hadamard basis + the last sent signal block, DCT basis + the last sent signal block). All the basis functions are normalized according to the same rule.
Each figure consists of 4 parts. The first shows the input signal, while the second the reconstructed signal based on the proposed overcomplete transformation (OT). The chosen transformation (T.) and the necessary number of coefficients (C.) are also presented for each signal block. On the third part of the figures the signal is reconstructed based on DFT using as many of the most significant basis components as is necessary for the coding in case of the OT to fulfill the L_2 norm requirement. The fourth part compares the L_2 errors of the approximate signals reconstructed in the two different ways compared with the original input signal.
The above results for a single sinusoid with a periodicity of N_1=14 are illustrated in Fig. 1.26 (the L_2 bound is 0.99). Fig. 1.27 shows the signals and errors for a square waveform input signal

with period N1=14, the L2 bound is 0.99. The input signal in Fig. 1.28 is pure white noise, the L2 bound is 0.99. Fig. 1.29 shows the case of sinusoid input with period N1=14 + 50% noise (the L2 bound is 0.98), Figs. 1.30-31 illustrate the cases of noisy square inputs (the noise ratio is 50 %), N_1=14, L_2 bound is 0.99 and N_1=16, L_2 bound is 0.90, accordingly.

Table 1.2 The transform library

Overcomplete Transformation #	Transformation Dictionary
T1	Hadamard basis
T2	DFT basis
T3	DCT basis
T4	Haar basis
T5	Hadamard basis + 1 DCT basis function
T6	DFT basis + 1 Hadamard basis function
T7	Hadamard basis + the last approximate signal block
T8	DCT basis + the last approximate signal block

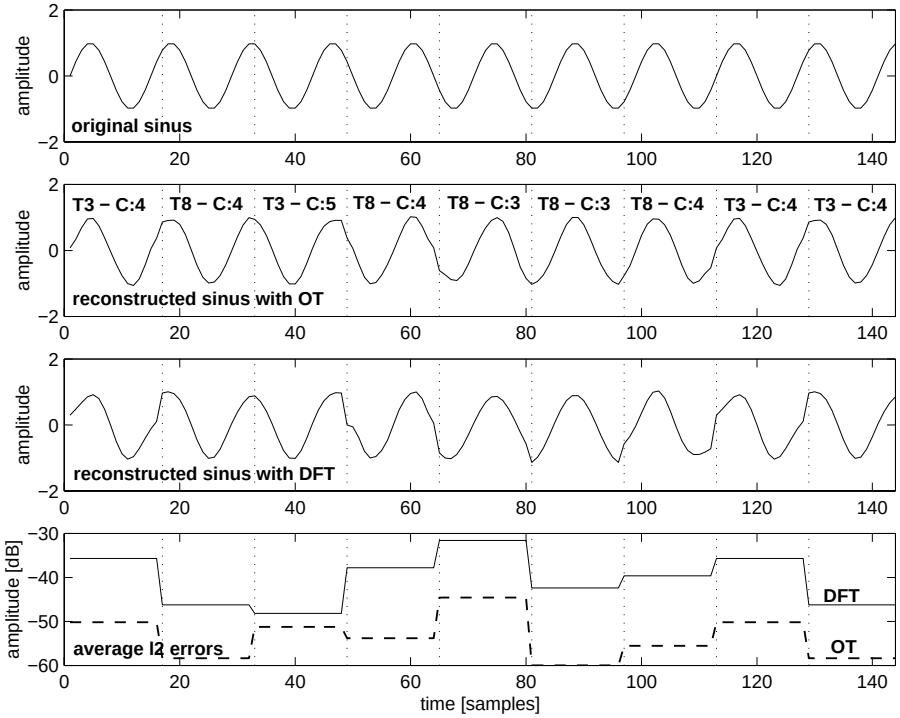


Fig. 1.26 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for a single sinusoid input with period N_1=14 (the L_2 bound is 0.99)

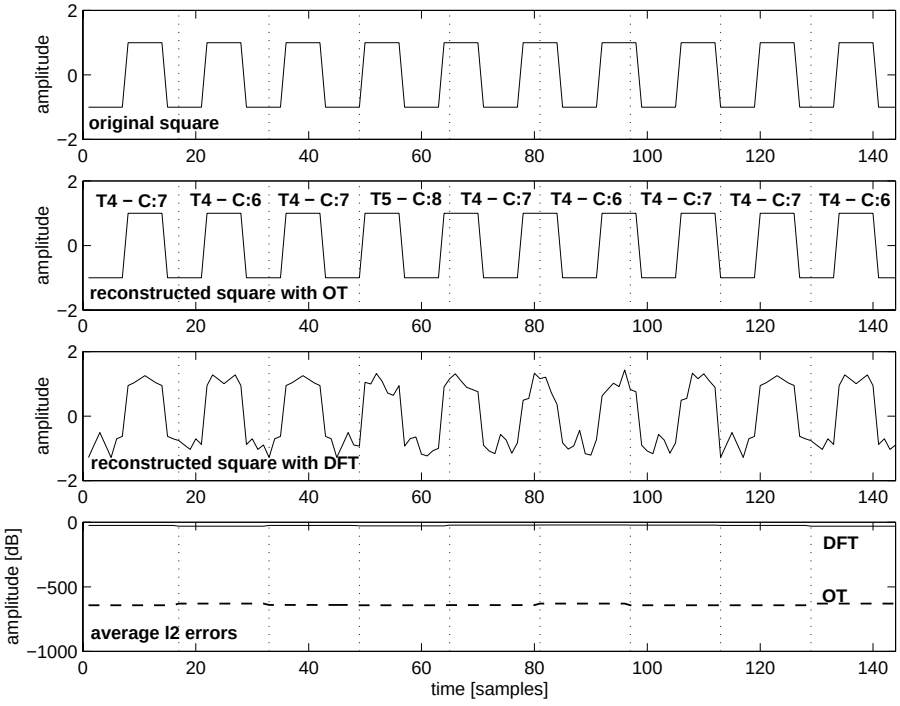


Fig. 1.27 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for a square waveform input signal with period N_1=14 (the L_2 bound is 0.99)

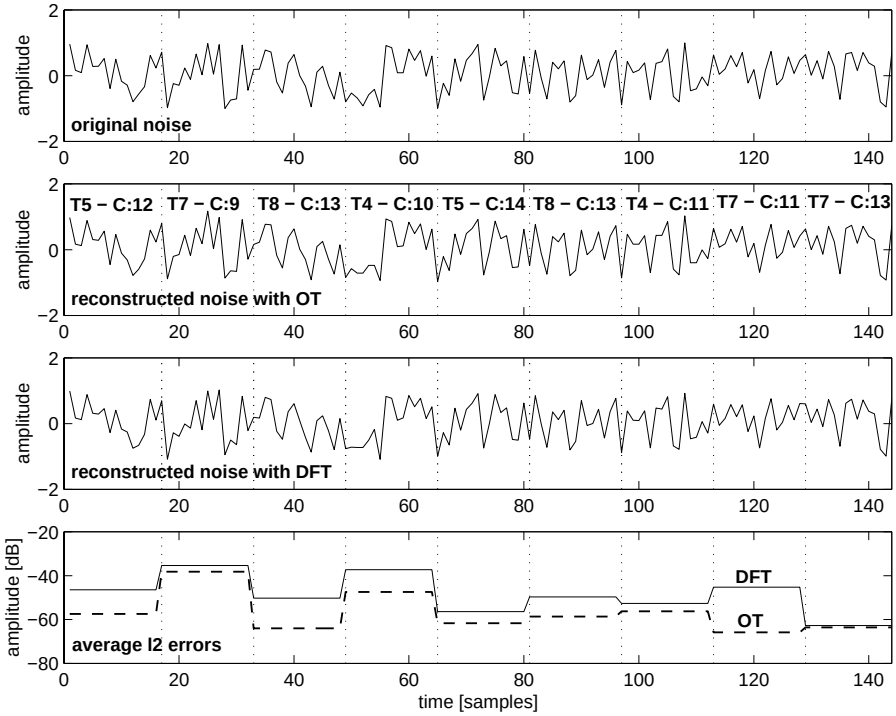


Fig. 1.28 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for a white noise input (the L_2 bound is 0.99)

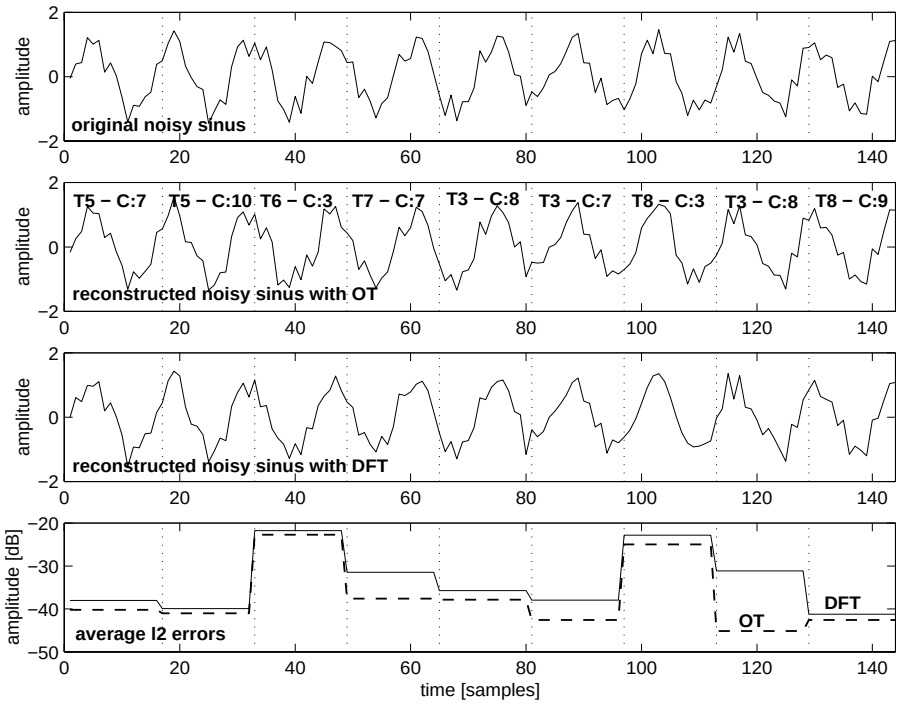


Fig. 1.29 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for a sinusoid input with period N_1=14 + 50% noise (the L_2 bound is 0.98)

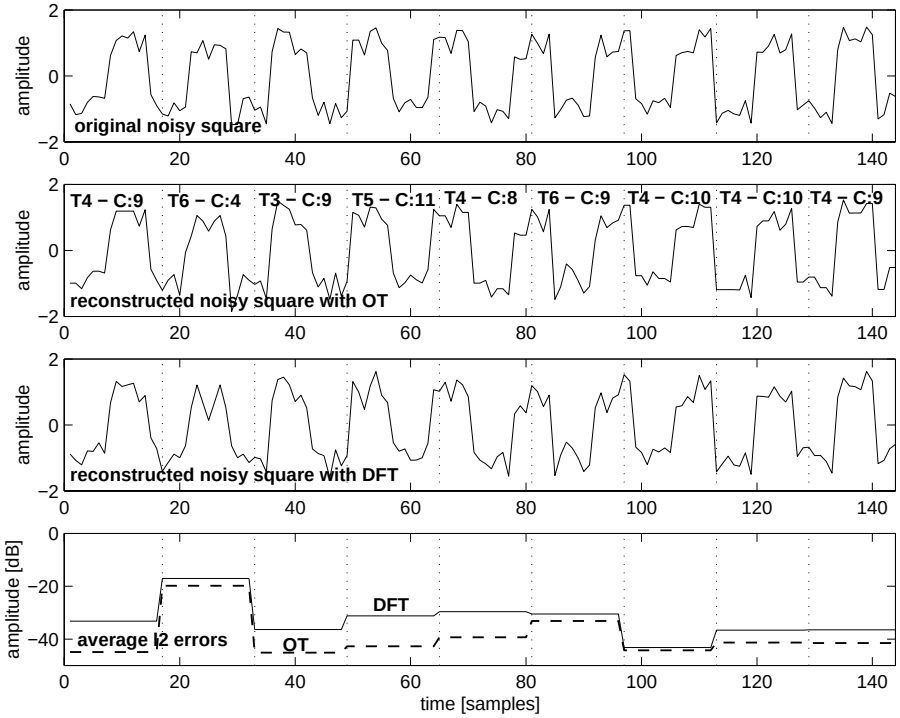


Fig. 1.30 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for square waveform input signal with period N_1=14 + 50% noise (the L_2 bound is 0.99)

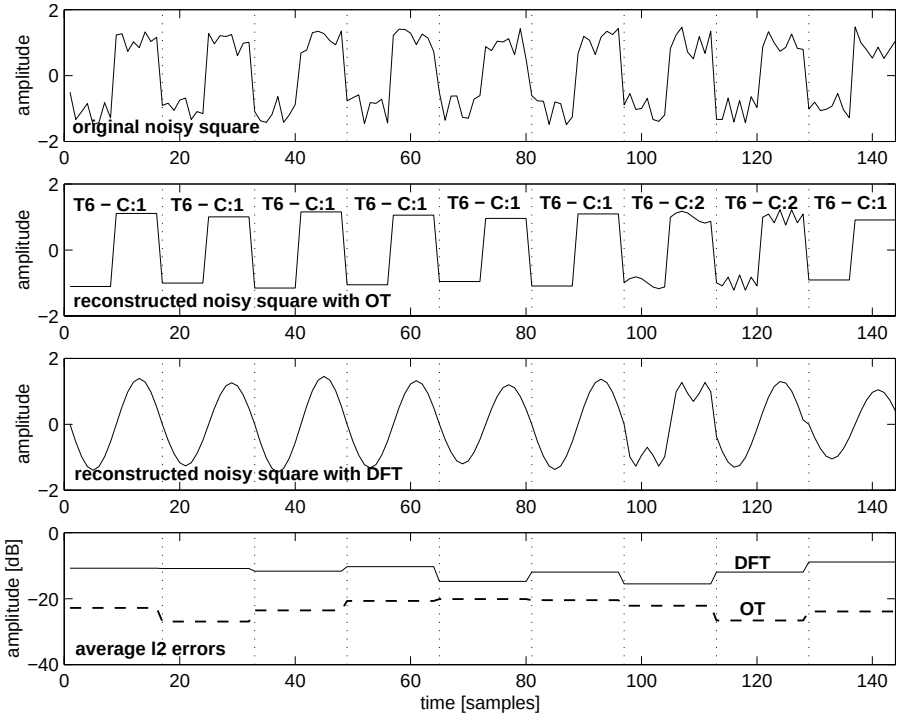


Fig. 1.31 The input signal, the reconstructed signal based on OT, the reconstructed signal based on DFT, and the L_2 errors (OT - dashed line, DFT - continuous line) for square waveform input signal with period N=16 + 50% noise (the L_2 bound is 0.90)

1.6 Uncertainty handling in non-linear systems

Measurements of any kind are characterized by their uncertainty and/or accuracy. Unfortunately for several reasons this characterization is not easy and in many cases requires human and/or machine based considerations and intensive computing. The complexity of the measurement problems of current interest has considerably increased. Thus, since all of these computations require time, additional requirements like speed, costs, etc. may strongly limit the achievable precision. With the appearance and spreading of new modeling and computing techniques, especially in real-time and/or embedded measurement systems, new possibilities arise to overcome the problems but the price to be paid for this is the reconsideration of measurement uncertainty. The representation method of the uncertainty must be on one hand in harmony with the modeling and information processing method; on the other hand it has to be uniform or at least "interpretable" by other representation forms.
This section deals with the above questions. First of all we point out the limits of the "classical", probability theory based uncertainty representation and we examine some possible new, fuzzy based uncertainty representation methods. This is followed by simulations focusing on the questions of "communication" between entirely different representations and how to express "optimally" the resultant (output) uncertainty based on hybrid (input) information.

1.6.1 Introduction

The information coming from our environment - including the results of measurement procedures - usually involves a certain amount of uncertainty, inaccuracy. The degree of this uncertainty is important additional information, because we can decide only in possession of this knowledge, in what degree and for what purpose the information is usable. The problem becomes even more critical due to the fact, that the information often comes through complex, nonlinear channels and most of the results are computed based on different, possible uncertain input data. Thus, it is important to have a uniform method for the representation and handling of uncertainty, by the aid of which the resultant uncertainty can be computed during the information process.
When we choose an uncertainty representation method, we have to consider several aspects of the problem. It is an essential expectation that the measure of uncertainty is analogous to the subjective opinion, i.e. a higher degree of uncertainty has to be connected to subjectively "more uncertain" data. The uniformity and popularity are also important factors, because the uncertainty of the information must be interpretable by everybody.
Important aspects are the nature and characteristics of the information processing system, as well. Traditionally the probability and interval analysis based data models are used to represent the information, and its uncertainty. Thus, in case of simpler, nearly linear systems based on "classical" modeling and computing methods, this well-proved and theoretically well established probability theory based uncertainty representation is preferable. In case of complex, non-linear and/or "soft" systems it can be more suitable - or even necessary - to find other representation forms [S48], [25], [26], which should cope with the data representation, resulting in a more accurate uncertainty expression and more comfortable usage. On the other hand, sometimes the type of uncertainty makes the probability theory-based representation difficult or even impossible (e.g. uncertainty originated from incomplete data and/or data-processing or subjective information sources).
At the same time, we can not forget that the different representation methods have to be comparable, convertible. Thus, the new representation methods have to be comparable with the "classical" probability theory-based methods.
The aim of this section is to establish the analysis of this topic and to investigate possible solutions for these problems. Subsection 1.6.2 deals with the "classical", probability theory based representation and its limits. Subsection 1.6.3 describes the new, promising fuzzy based representation methods, while Subsection 1.6.4 deals with mixed data models where different

modeling methods and thus different uncertainty representation methods are used within one (possibly huge) system. Different conversion methods between probability based and fuzzy data models are compared and some suggestions are made how to reduce the conversion error. Subsection 1.6.5 analyzes the comparability, convertibility of the new methods and the "classical" techniques. The qualification of the data is investigated together with proposing new uncertainty measures for the quantification of uncertainty of fuzzy variables.

1.6.2 Uncertainty representation based on probability theory

In practical applications, the probability theory based uncertainty representation is wide-spread used for the expression of the uncertainty of the information. The advantages of this method are the uniformity, popularity, and the profound theoretical background [27]. The standards and recommendations regarding the expression of measurement uncertainty are also linked to this method.

Traditionally, two main types of measurement errors were considered: the so-called random and the systematic errors. The effects of random errors can be reduced by statistical methods, while the systematic ones call for deterministic corrections. Simultaneously in the recently wide-spread standards ([28]) a different approach has appeared: the so-called 'A' and 'B' types of uncertainty. The uncertainty is expressed in both cases as variance or standard deviation. For category 'A' the evaluation is based only on the statistical evaluation of the measurement, while for category 'B' also a priory information, like calibration data, can be considered.

Sometimes it is necessary to define the measurement uncertainty by an interval, which includes a great part of the - hypothetical - distribution of the measured data (see [29]). The width of this interval (in the standards: extended measurement uncertainty) is computed from the standard deviation (in the standards: standard measurement uncertainty) with the help of a so-called extension factor:

$$U_x = k * d_x , \tag{1.25}$$

where U_x stands for the half-width of the interval, and d_x denotes the standard deviation. The extension factor k depends on the confidence level and the type of the distribution. In case of non-linear transformations, the actual distribution of data is not or only with difficulties obtainable, so the extension factor and the width of the confidence interval are not or only approximately computable. On the other hand, when computing the confidence intervals the resultant standard deviation is used, thus it is an important question, how accurate the computing of the resultant standard deviation can be in non-linear systems.

In case of nearly linear systems, there are known methods, based on the first-order Taylor expansion of the system, for the expression of the resultant uncertainty. If *f()* maps the input variables $x_1, x_2, ..., x_N$ into the output variable *y*, the resultant uncertainty can be assessed according to the followings:

$$d_y^2 \cong \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i}\right)^2 d_i^2 + 2\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} r_{i,j} , \tag{1.26}$$

where d_i stands for the standard deviation of x_i, and $r_{i,j}$ denotes the covariance between x_i and x_j.

For the examination of the probability theory based uncertainty representation the MATLAB package [30] was used. By simulating different non-linear characteristics the computed standard deviation of the output variable (1.26) and the variance got from the simulations were compared.

Theorem 1.2 ([S17], [S117], [S55], [S60]):

The classical, probability based uncertainty representations given by the measurement standards [28] cannot be used with high confidence for expressing the resultant uncertainty if the

characteristics of the system is non-monotone and changes rapidly on the interval, which contains a great part of the - hypothetical - distribution of the measured data.

Proof:

The opposite of Theorem 1.2 can be disproved by counter-examples. For this, we used the MATLAB package and analyzed the error propagation in case of different non-linear mapping function. We have analyzed approximately fifty non-linear and non-monotonous characteristics. As input values, random variables were used (generated by the random generator of the MATLAB toolbox) and the output were measured and analyzed by statistical methods.
Here we include two typical (one monotonous and one non-monotonous) results. As a conclusion, we could prove that if the requirement in Theorem 1.2 does not held, the formulas of the standards do not give accurate results, in fact sometimes they are not at all suitable for the computing the resultant uncertainty (Figs. 1.31-1.32).

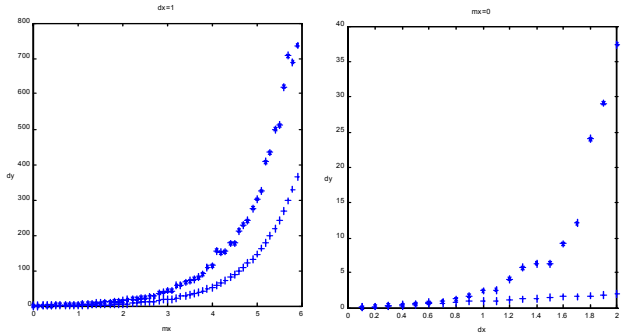


Fig. 1.32 The effect of the exponential transformation on the variance and the standard deviation.
++++++ : values computed by (1.26); ********: values got by the simulation

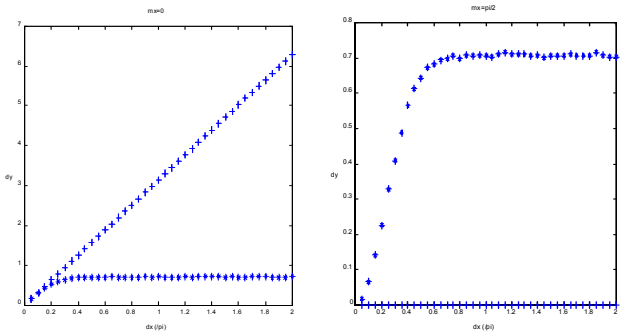


Fig. 1.33 The effect of the sinusoidal transformation on the variance and the standard deviation.
++++++ : values computed by (1.26); *******: values got from the simulation

For the problems, a possible solution can be the improving of the formulas - e. g. by the aid of the higher-order derivatives. But this would make the computing more complex and not necessary result in better result e. g. in case of systems, where also the higher-order derivatives change fast or are non-monotone. So it is worth considering, using other modeling techniques for the representation of the uncertainty in non-linear systems.
Another, not negligible aspect is, that recent complex systems, which are often based on “soft” computing methods, raise new problems in the area of uncertainty representation. These systems are often based on modeling methods, which are completely different from the probability theory, so the “classical” tools for uncertainty representation are not usable. Furthermore, transformations realized by complex systems are often not describable by simple functions, so

the derivative can not or only approximately be computed. In these systems, the representation of the uncertainty during the data processing has not been yet solved and, thus, the authenticity and accuracy of the resultant data are also not known.

1.6.3 Uncertainty representation based on fuzzy theory ([S17], [S117], [S55])

Fuzzy theory [31] is often used to solve modeling and computing tasks, thus, with the fuzzy based uncertainty representation we can solve two problems. First, there is a possibility, that we can improve the overall accuracy of the computing of the resultant uncertainty in non-linear systems. Secondly, we can easily represent and compute the uncertainty in fuzzy based systems, where probability theory can be used only with difficulties. Another great advantage of the fuzzy representation is that it is close to human thinking, and the effect of parameter changes can be easily estimated, so the fuzzy based systems are usually easy to survey. Furthermore, some kinds of uncertainty (e. g. incomplete data or subjective information) can be expressed far better by fuzzy sets, then by probabilistic or statistical methods.
In case of fuzzy representation of the information, data is not expressed by a definite value ("measured data") and the degree of the connected uncertainty (variance, standard deviation, confidence-interval); nor by an - approximate - probability distribution. It is expressed by a membership function, which gives the degree in which the elements of the universe belong to the set, containing the value(s) of the measured data. This method can be used to express non-statistical type uncertainty, e. g. originated from subjective data, as well.
To express the data we can use fuzzy numbers, i. e. fuzzy sets defined on $\Re$ (the universe of real numbers), which are normalized (at least one of its elements attains the "1" membership grade) and convex:

$$\mu(\lambda r + (1-\lambda)s) \geq \min(\mu(r), \mu(s)) \qquad \forall r, s \in R\,;\ \lambda \in [0,1] \tag{1.27}$$

If during the information processing fuzzy data representation is used and we manage to define a degree of uncertainty to the membership function, the uncertainty becomes expressible and measurable at any point of the system. The fuzzy data representation is solved in some soft computing (e.g. fuzzy, fuzzy-neural), systems. In case of "classical" data-processing methods the membership function of the output variable y is computable from the membership functions of the input variables x_1, x_2, ..., x_N with the help of the extension principle [32]:

$$\mu(y) = \sup\{\min(\mu_1(x_1), \mu_2(x_2), ..., \mu_N(x_N)) \mid \ y = f(x_1, x_2, \ldots, x_N)\ \} \tag{1.28}$$

If we want to change to "classical" data representation (e.g. at the end of the data processing we need a definite output value), the fuzzy numbers have to be defuzzificated. Different defuzzification methods are known, here the center of gravity (CoG) method [33] is used, because of its analogy to probability theory and because this method may possibly express the information involved in the membership function the most faithfully. With the center of gravity method, the defuzzificated value can be computed accordingly

$$x^* = \frac{\int x\mu(x)dx}{\int \mu(x)dx}. \tag{1.29}$$

In the literature (see e.g. [33]) there are several measures for the expression of the uncertainty of fuzzy sets (e. g. U-uncertainty), however these measures are based on discrete domains, where no distance is defined or considered between the discrete values. Thus, though these measures can be very useful at the expression of the uncertainty in case of diagnostic tasks, they are not suitable for the expression of measurement uncertainty.

During our examinations several candidates, avoiding the above problem, were examined to express the fuzzy type uncertainty (e.g. quadratic deviation from the defuzzificated value). In the followings two of them will be proposed, which proved to be the most promising.

Width of α-cut

The uncertainty represented by a fuzzy number can be expressed by the width of the interval, where the membership function is higher or equal then a given α value. Since the membership function is convex, this interval can't be "holed".

$$u_{1,\alpha} = x_2 - x_1, \text{ where } x_1 = \inf\{x \mid \mu(x) \geq \alpha\} \text{ and } x_2 = \sup\{x \mid \mu(x) \geq \alpha\}. \quad (1.30)$$

Integral of the membership function

The area under the curve can also express the uncertainty, since wider, higher membership functions result in a higher value. Theoretically there can be cases, when two datasets with subjectively different uncertainty get the same value, but if we use only fuzzy numbers, these cases can be excluded.

$$u_2 = \int_{-\infty}^{+\infty} \mu(x)dx. \quad (1.31)$$

Beyond being analogous with the subjective opinion (e.g. data represented by wider membership function must have a higher uncertainty value), the measure of uncertainty must also be comparable with the probability theory based representation, and it must behave similarly to the "classical" measures during different transformations. Partly because the probability theory based representation is widely used and it is favorable for the users, the new measure of uncertainty should be comparable, convertible with the "classical" techniques. Furthermore, we must ensure the "communication" between systems based on different modeling techniques, which means the existence of some kind of "rule of conversion" between the different measures of uncertainty.

In this section, two aspects of the question of comparability, convertibility are examined:

1. Relations between the fuzzy measures of uncertainty and the standard deviation
2. Relations between the fuzzy measures of uncertainty and the standard deviation after different transformations

In the practice, the normal distribution is often used, as supposed distribution of the variables, so random variables with normal distribution and the corresponding fuzzy numbers are considered for the comparisons.

Relations between the fuzzy measures of uncertainty and the standard deviation

The membership function of a fuzzy number, corresponding to a normal random variable with an expected value of $\bar{x}$ and standard deviation of σ, can be computed over a discrete universe as [34] (see Fig. 1.34):

$$\mu\left(\frac{x_i + x_{i-1}}{2}\right) = \sum_{k=1}^{N} \min\{\lambda_i, \lambda_k\}, \quad (1.32)$$

where $\lambda_i = \int_{x_{i-1}}^{x_i} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right)dx$, and $x_0 = -\infty$, $x_N = +\infty$, $x_i = \bar{x} - 3\sigma + \frac{(i-1)6\sigma}{N-2}$, $i = 1,\ldots N-1$.

With the help of these formulas we can examine the relation between the standard deviation and the uncertainty of the corresponding fuzzy number (Fig. 1.35). It is preferred, to have a linear relationship between the uncertainty measures, because then they can be converted easily to each other. This expectation is fulfilled in both cases (u_1 and u_2).

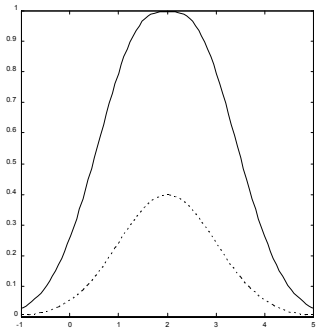


Fig. 1.34 A normal distribution function (dashed line) and the membership function (continuous line) of the corresponding fuzzy number

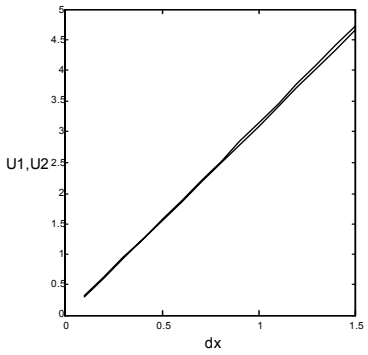


Figure 1.34 Relation between the standard deviation and the uncertainty of the corresponding fuzzy number. ——: u_1 (α -cut, α=0,5); - - - -: u_2 (integral)

Relations between the fuzzy measures of uncertainty and the standard deviation after different transformations

The linear transformations are a special and important group of transformations. It is known, that in case of a linear transformation, given by the formula

$$y = ax + b, \tag{1.33}$$

it is true for the standard deviation of the variables *x* and *y*, that

$$d_y = |a| d_x . \tag{1.34}$$

This conforms to the subjective expectation, as well, thus it is necessary, that the new measures of uncertainty also fulfill this formula. Since this transformation (if $a \neq 0$) is a one-to-one mapping, we get from the extension principle (1.28):

$$\mu_y(y) = \mu_x(\frac{y-b}{a}) . \tag{1.35}$$

Width of α-cut

From formula (1.35)

$$\mu_x(x_1) \geq \alpha \Leftrightarrow \mu_y(ax_1 + b) \geq \alpha . \tag{1.36}$$

Thus, if the α-cut of the input fuzzy number is $[x_1, x_2]$, with width $x_2 - x_1$, the α-cut of the output variable will be $[ax_1 + b, ax_2 + b]$ (if a>0) or $[ax_2 + b, ax_1 + b]$ (if a<0) with a width of $|a|(x_2 - x_1)$.

Integral of the membership function
With the $y = ax + b$ substitution we get

$$u_2(y) = \int_{-\infty}^{+\infty} \mu_y(y)dy = |a| \int_{-\infty}^{+\infty} \mu_x(x)dx = |a|u_2(x) \tag{1.37}$$

A further expectation can be, that there must be a simple relationship between the standard deviation and the uncertainty of the corresponding fuzzy number in case of other, nonlinear transformations, as well. It is preferred, if this relation is also linear, with the same or similar factor. This factor must be the same or similar to the factor computed from the input variables.
In this case, the examinations were carried out by simulations, using the MATLAB Package. The fuzzy uncertainties were estimated by '*ad*', where *d* is the standard deviation of the corresponding random variable. The results are summarized in Tables 1.3 and 1.4: '*l*' is the "optimal" factor of the linear estimation (where $\sum(u_i - ld)^2$ is minimal), and '*e*' is the relative quadratic error ($e = \Sigma(\frac{u_i - ld}{ld})^2$).
The fuzzy uncertainties can also be estimated with '$l_0 d$', where l_0 is the "optimal" factor for the input variables (u_1: l_0=3.1224; u_2: l_0=3.1606). In this case, there is a bigger relative quadratic error $e_0 = \Sigma(\frac{u_i - l_0 d}{l_0 d})^2 \geq e$. It is preferred, if the '*l*' factors are nearly the same in case of different transformations, and '*e*' and 'e_0' are small.
As it appears from the results, in most cases, there may exist a linear conversion between the standard deviation and the fuzzy uncertainty ($u_i = ld$) with a relatively small error. In some cases, there are higher errors (e. g. addition, multiplication). This fact needs further examinations.

Table 1.3 Fuzzy uncertainties in case of different non-linear mappings

	x m_x=1	x+y m_x=1,m_y=2 d_x=0.1	x+y m_x=1,m_y=2 d_x=d_y	x*y m_x=2,m_y=1 d_x=0.1	x*y m_x=5,m_y=1 d_x=0.1	1/x m_x=10	x^2 m_x=1	x^2 m_x=5
u_1 a	3.1224	3.3712	4.3708	3.6604	4.0753	3.0774	2.6592	3.0948
e	0	0.1278	0	0.0987	0.0483	0.0028	0.1337	0.0012
e_0	0	0.2976	1.2237	0.5753	0.9067	0.0035	0.1964	0.0016
u_2 a	3.1606	3.444	4.4504	3.7747	4.1165	3.1677	2.8861	3.1297
e	0	0.1244	0	0.0696	0.0454	0.0021	0.0316	0.0019
e_0	0	0.3226	1.2237	0.6051	0.888	0.002	0.0787	0.0026

Table 1.4 Fuzzy uncertainties in case of different non-linear mappings

	$\sqrt{x}$ m_x=10	$\sqrt{x}$ m_x=25	log(x) m_x=10	log(x) m_x=25	e^x m_x=1	e^x m_x=2	sin(x) m_x=0	sin(x) m_x=1	sin(x) m_x=π/2
u_1 a	3.1226	3.1323	3.0868	3.1431	15001	1.5255	3.0262	2.9412	2.3609
e	0.0048	0.0067	0.007	0.0043	2.2799	2.1312	0.0616	0.0358	0.4742
e_0	0.0048	0.0067	0.0063	0.0048	5.8949	5.7440	0.0483	0.0704	1.8262
u_2 a	3.1683	3.145	3.1468	3.1818	2.3317	2.3623	2.8294	2.746	2.281
e	0.0019	0.0007	0.0028	0.002	0.5841	0.51110	0.0661	0.0617	0.0388
e_0	0.0017	0.001	0.0022	0.0018	0.5967	0.5975	0.1279	0.1794	1.3697

1.6.4 Mixed data models ([S17], [S117], [S60])

Although there are several information processing methods based on the different homogenous data models, mixed data models often need to be used with the increasing complexity of the problems and with the appearance and spreading of heterogeneous, connected systems,
Complex, difficult problems need the partitioning of the task and the system. It can be found, that different parts of the problem need different computing methods, which are possibly based

on different data models. For example, after some "classical" pre-processing methods, a fuzzy based pattern recognizing algorithm can follow. In this case, the different data models follow each other in time, inside one part of the system one uniform data model is used, but some conversion method is needed between the different parts.
Different data models can be in use parallel with each other, as well. E.g., in case of a system obtaining its inputs from different sources, it may be possible that the inputs are variables given in different data representation models. For example, inputs originated from measurement procedures or some "classical" information processing methods, are given as probability variables, while other input, originating from fuzzy production systems or subjective information sources are given as fuzzy variables. In this case, some methods for the treating of these different data models are needed.
A possible solution can be the use of fuzzy random variables [36]. Fuzzy random variables are fuzzy-valued random variables, and can be described by a set of $v=\{(F_i,p_i), i=1,2,...n\}$, where F_i is a fuzzy set, over the universe of real numbers and p_i is the probability of the event, labeled F_i. The expected value of a fuzzy random variable is a fuzzy variable

$$v = \sum_i p_i F_i \tag{3.38}$$

An exact, real-valued result can be obtained by the defuzzification of the expected value.
In general term, fuzzy and random variables can also be treated as special fuzzy random variables. In the first case, the probability of the given fuzzy variable is considered to be 1 while in the second case, the real numbers could be treated as special fuzzy variables (singletons), where the membership function is 1 only in one point, and 0 otherwise.
With that extension, any operation on fuzzy and random variables can be carried out and the obtained results will be fuzzy random variables. For example, if $g(x,y)$ maps the x random and y fuzzy input variables into the output variable v, then the result will be the $v=\{(F_i,p_i), i=1,2,...n\}$ fuzzy random variable, where $F_i = g^*(x_i, y)$ (g^* is the fuzzy version of $g()$, got from the extension principle), and p_i is the original probability of x_i (Fig. 1.36).
The problem with the fuzzy random variables is that the calculations are very time consuming. In the above example, g^* must be executed n-times and in the next phase when the outputs of the procedure are fuzzy random variables, the operations must be carried out n^2-times. This can cause an exponential explosion during the information processing.

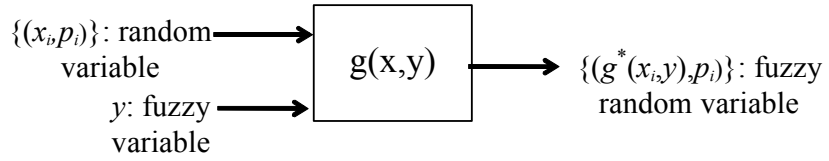


Fig. 1.36 Computation with random and fuzzy variables

Another solution is, if instead of using fuzzy random variables, all input variables are converted into one uniform data model, which could be either the fuzzy or the probability based data model. In this case, information processing can be carried out in that uniform data model, so the computation is not so time-consuming.
Conversion methods
Conversion between different data models is needed, in case of complex, heterogeneous systems, where the different parts of the system are based on different data models. On the other hand, the conversion of the inputs into one uniform data model can be a solution in cases of input, given in different data models, as well.
Conversion can be treated as calculation of the $\mu()$ membership function from the $f()$ density function. Another, different approach is that the conversion is made between the

$p=(p_1,p_2,\ldots,p_n)$ probability distribution and the $r=(r_1,r_2,\ldots,r_n)$ possibility distribution, where $p_i \geq p_{i+1}$, $r_i \geq r_{i+1}$ and $r_1=1$. The possibility distribution can be treated as the discrete sampling of the $\mu()$ membership function, but the values are ordered not by x_i, but by $r_i=\mu(x_i)$. Several conversion methods between fuzzy and probability based data models can be found in the literature (e. g. [37], [38], [39]). The simplest conversion method is the scaling ([38], [39]), where

$$\mu(x)=\frac{f(x)}{\max(f(x))} \quad \text{or} \quad r_i=\frac{p_i}{p_1} \quad \text{and} \tag{3.39}$$

$$f(x)=\frac{\mu(x)}{\int \mu(x)dx} \quad \text{or} \quad p_i=\frac{r_i}{\Sigma r_i}. \tag{3.40}$$

Another, more complicated method can be the following [38], [39]

$$\mu(x)=\int \min\{f(y),f(x)\}dy, \text{ or } r_i=\sum_{j=1}^{n}\min\{p_i,p_j\} \text{ and} \tag{3.41}$$

$$p_i=\sum_{j=i}^{n}\frac{(r_j-r_{j+1})}{j}. \tag{3.42}$$

Conversion can also be based on the correspondence between the confidence intervals of the random variable and the α-cut of the fuzzy variable [38], [39]

$$\mu(x_1)=\mu(x_2)=\int_{-\infty}^{x_1} f(x)dx+\int_{x_2}^{\infty} f(x)dx, \text{ or } r_i=\sum_{j=i}^{n} p_j. \tag{3.43}$$

where $[x_1,x_2]$ is a confidence interval. Instead of the inverse of this conversion, usually the previous conversion method (3.42) is used.
The different conversion methods give different results (see Fig. 1.37), so two main questions arise:

- Into which data model to convert the data?
- By which conversion method?

In some cases, the answer for the first question can be predetermined (for example, calculation

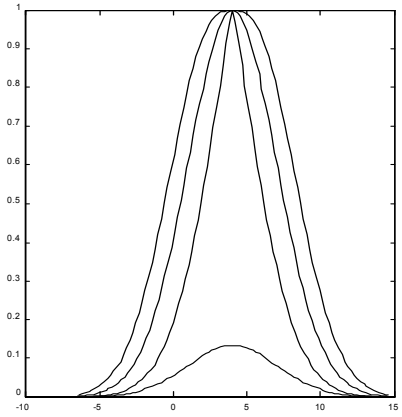


Figure 1.36 The results of different conversion methods.
——: density function of a normal random variable (*m*=4, *d*=3); –·–·–: membership function (3.39); ·····: membership function (3.41); – – – –: membership function (3.43)

methods and their data model is given and only the conversion of input into the proper data model is required), but if the calculation methods and data models are not yet settled, the uniform data model can be freely chosen.
For the finding answers for these questions, several (appr. 50) examinations were carried out by making simulations using the MATLAB package. The above-described five conversion methods were examined.
The examinations were carried out in case of the basic arithmetic operators (addition, multiplication, division), thus the input variables/results can be extended to rational functions. One of the input variable was a normal random variable and the other was a triangular fuzzy variable. In case of division, the cases of fuzzy and random denominators were separately examined.
First, input was converted into one, uniform - fuzzy or probability based - data model and results were calculated based on this data model. After that, the results were compared to the reference value, which was computed by using fuzzy random variables according to the following steps:

1. Discrete approximation of the input random variable: $\{(x_i,p_i), i=1,2,...n\}$.
2. Calculation of the $F_i = g^*(x_i, y)$ values, where $g^*()$ is the fuzzy extension of the *g()* mapping, describing the transformation.
3. The result is a fuzzy random variable: $\{(F_i,p_i), i=1,2,...n\}$.
4. Calculation of the (fuzzy) expected value of the result: $\sum_i p_i F_i$.
5. The reference value is the defuzzificated expected value.

A part of the resultant error originates from the discrete approximations (step 1), but it is considerably smaller, then the error originated from the conversion itself.
The following observations can be made (Table 1.5 shows some of the results):

- Usually conversion into random variables gives better results. Nevertheless, in many cases, the use of fuzzy variables can be not or only at great costs avoided. Fuzzy based systems simply need fuzzy input and some kind of inaccuracy, uncertainty can not be correctly represented by random variables.
- If the output is highly non-linear function of one of the input variables (e.g. in case of division, where the denominator is less, then 1), the conversion of that input causes bigger errors. So, in this case, conversion into the data model of this "non-linear" input gives better results. The cause of this phenomenon can be that in case of linear or nearly linear functions, if the original density/membership functions are symmetric, then results have also symmetric or nearly symmetric density/membership functions. In case of symmetric or nearly symmetric density/membership functions, the expected value/defuzzified value will be around the center of the distribution/fuzzy set, independently of the exact shape of the distribution/membership functions. Therefore in these cases, conversion can not cause too high error (all of the examined conversion methods preserve the symmetric property).
- Conversion methods into random variable give similarly "good" results, there can not be made a distinction among them.
- From conversion methods into fuzzy variable, the third (3.43) gives the best results, while the second (3.41) the worst.

Consequently, if the uniform data model can be chosen freely (e.g., if information processing and the data model used by it is not given), then it is suitable to choose the data model of the "less linear" input of the system, as uniform data model. If there are no great differences in the linearity, then the probability based data model can be a good choice.
In many cases, the data model of the system is given, e.g., expert knowledge must be used or the lack of information must be represented, therefore fuzzy data model must be used. A similar case is, when a highly non-linear module follows, where the use of fuzzy data model can be more suitable and we choose it in order to reduce the number of conversions. In these cases, the

Table 1.5 Difference from the reference value. a) Addition of fuzzy and random variables (the expected value and the defuzzificated values of the inputs are 10); b, Multiplication of fuzzy and random variables (the expected value and the defuzzificated value of the inputs are 10); c) Division of random variable by fuzzy variable (the expected value of the dividend is 10, and the defuzzificated value of the divisor is 0.5); d, Division of fuzzy variable by random variable (the defuzzificated value of the dividend is 10 and the expected value of the divisor is 0.5); h1-h3: absolute errors in case of conversion into fuzzy variables, according to (3.39), (3.41), and (3.43), respectively; h4-h5: absolute error in case of conversion into random variables, according to (3.40), and (3.42), respectively; dx: the standard deviation of the random input

1.a		dx=0.1	dx=0.2	dx=0.3
h1	df=0.3	0.0006	0.0009	0.0001
	df=0.6	0.0007	0.0005	0.0008
h2	df=0.3	0.00007	0.0008	0.0006
	df=0.6	0.0007	0.0008	0.0011
h3	df=0.3	0.0008	0.0009	0.0749
	df=0.6	0.0006	0.0008	0.0004
h4	df=0.3	0.0025	0.0021	0.0004
	df=0.6	0.0009	0.0016	0.0006
h5	df=0.3	0.0000	0.0011	0.0014
	df=0.6	0.0003	0.0010	0.0047

1.b		dx=0.1	dx=0.2	dx=0.3
h1	df=0.3	0.0443	0.0741	0.1131
	df=0.6	0.0769	0.1467	0.2243
h2	df=0.3	0.0499	0.0887	0.1221
	df=0.6	0.0862	0.1711	0.2424
h3	df=0.3	0.0349	0.0887	0.0749
	df=0.6	0.0611	0.1711	0.1819
h4	df=0.3	0.0032	0.0026	0.0017
	df=0.6	0.0052	0.0078	0.0083
h5	df=0.3	0.0054	0.0056	0.0061
	df=0.6	0.0067	0.0067	0.0064

1.c		dx=0.1	dx=0.2	dx=0.3
h1	df=0.3	0.1541	0.3631	0.5679
	df=0.48	0.4318	1.0166	1.6032
h2	df=0.3	0.1752	0.4050	0.6353
	df=0.48	0.4885	1.1223	1.7625
h3	df=0.3	0.1211	0.2965	0.4733
	df=0.48	0.3504	0.8530	1.3670
h4	df=0.3	0.8704	3.8430	3.8278
	df=0.48	15.8224	15.9831	16.0178
h5	df=0.3	4.3603	4.3657	4.3331
	df=0.48	17.1521	17.2090	17.2090

1.d		dx=0.1	dx=0.14
h1	df=0.3	2.6339	14.9361
	df=0.6	2.7887	23.6055
h2	df=0.3	4.0444	26.9831
	df=0.6	4.2486	41.0495
h3	df=0.3	1.3041	6.0385
	df=0.6	1.4439	8.9964
h4	df=0.3	0.0106	0.0542
	df=0.6	0.0177	0.1309
h5	df=0.3	0.0037	0.1378
	df=0.6	0.0113	0.1441

third conversion method can be used to make lower conversion error. If a faster, simpler conversion method is needed, the first one, scaling can be a good choice.
We would like to remark that a recently proposed new fuzzy alternative, the use of type-2 fuzzy sets (see e.g. [40]) seems to be also a promising possibility, however it needs further investigations.

1.6.5 Qualification of results in case of mixed data models ([S17], [S117], [S55], [S60])

The qualification can easily be solved if we convert every data into one, uniform data model. In this case, qualification can be carried out according to the uniform data model. If the uniform data model is the probability theory based data model (e.g. the system calculated with random variables), the resultant standard deviation can be calculated either from the distribution function of the output or with the help of some approximate method.
If the uniform data model is the fuzzy data model, we must define a "measure of uncertainty", which can be calculated from the membership function. The measure of uncertainty has to fulfill the following requirements: (1) it must be analogous with the subjective opinion; (2) it must be comparable with the probability theory based uncertainty measures (e.g. standard deviation); (3) it must behave similarly to the "classical" measures during different transformations; (4) we must ensure communication between systems based on a "rule of conversion" between the different measures of uncertainty; and, finally (5) comparability and convertibility are important for human users, who are at home in probability theory based uncertainty measures and expect similar behaving from fuzzy uncertainty measures, as well.
Based on the results of Subsection 1.6.3, two methods can be proposed for quantifying uncertainty of fuzzy data:

1. Width of α-cut:

$$u_{1,\alpha} = x_2 - x_1 \text{, where } x_1 = \inf\{x \mid \mu(x) \geq \alpha\} \text{ and } x_2 = \sup\{x \mid \mu(x) \geq \alpha\}. \tag{3.44}$$

In some aspects this uncertainty measure is analogous with the width of a confidence interval, a higher α value means a higher level of confidence.

2. Integral of the membership function

$$u_2 = \int_{-\infty}^{+\infty} \mu(x)dx. \tag{3.45}$$

This second method can be recommended, if the fuzzy variable is not normalized and/or is not convex.
As a summation of the above results we can conclude that if we represent the uncertainty with the width of α-cut of a fuzzy number or with the integral of the membership function, we get a suitable method which behaves similar to the standard deviation and the classical and fuzzy based uncertainty measures are easily convertible.
In case of inputs of different data models, one solution could be the use of fuzzy random variables. Although, calculations with fuzzy random variables are very time-consuming, so a faster and more general solution can be the use of conversion methods. With them, all input can be converted into one, uniform data model and calculations can be carried out using this uniform data model. With the use of these methods, conversion between the parts of the complex, heterogeneous systems, based on different data models can be solved.
Examinations also showed that usually the conversion into random variables causes less error, except in cases, when the output is strongly non-linear function of one of the input variables when it is recommended to keep the original data model of that input. In case of conversion into fuzzy variable, there can be stated a "goodness" order of the conversion methods.

Part II New Methods in Digital Image Processing

Enhancement of noisy image data is a very challenging issue in many research and application areas. In the last few years, non-linear filters, feature extraction, high dynamic range (HDR) imaging methods based on soft computing models have been shown to be very effective in removing noise without destroying the useful information contained in the image data. In Chapter II new image processing techniques are introduced in the above mentioned fields, thus contributing to the variety of advantageous possibilities to be applied. The main intentions of the presented algorithms are (1) to improve the quality of the image from the point of view of the aim of the processing, (2) to support the performance, and parallel with it (3) to decrease the complexity of further processing using the results of the image processing phase.

2.1 Introduction

With the continued growth of multimedia and communication systems, the instrumentation and measurement fields have seen a steady increase in the focus on image data. Images contain measurement information of key interest for a variety of research and application areas such as astronomy, remote sensing, biology, medical sciences, particle physics, science of materials, etc. Developing tools and techniques to enhance the quality of image data plays a very relevant role in any case. Enhancement of noisy images, however, is not a trivial task. The filtering action should distinguish between unwanted noise (to be removed) and image details (to be preserved or possibly enhance). Soft computing, and especially fuzzy systems based methods can effectively complete this task outperforming conventional methods. Indeed, fuzzy reasoning is very well suited to model uncertainty that typically occurs when both noise cancellation and detail preservation (enhancement) represent very critical issues. As a result, the number of different approaches to fuzzy image processing has been progressively increasing [1].

In this chapter we deal with different areas of image processing and introduce new soft computing (fuzzy) supported methods. In Section 2.2 corner detection is addressed. Two new fuzzy based corner detection methods represent our contribution to the field.

Section 2.3 deals with useful information extraction. “Useful” information means that the information is important from the further processing point of view and the, from this aspect non-important (in other situations possibly significant) image information is handled as noise, i.e. is filtered out. In this section, we present a new method for separating the primary and non-primary edges in the images.

Sections 2.4 and 2.5 are devoted to high dynamic range imaging. 5 novel approaches are detailed for reproduction of images distorted by the high dynamic range of illumination. Finally, Section 2.7 shows illustrative examples.

2.2 Corner detection

Recently, the significance of feature extraction, e.g. corner detection has increased in computer vision, as well in related fields. Corner detection helps to determine the most characteristic points of an object and thus to reconstruct it. Corners are also useful in pattern recognition. In this chapter, a new corner detection technique is introduced, which is based on fuzzy reasoning and applies a special local structure matrix. Furthermore, by introducing a new attribute

associated to the corners, the method efficiently supports further processing, e.g. point correspondence matching in stereo images or 3D reconstruction of schemes.

2.2.1 Introduction

Corner detection plays an important role in computer vision, pattern recognition [2], in shape and motion analysis [3] as well as in 3D reconstruction [4], [5] of a scene. Motion is ambiguous along an edge and unambiguous at a corner [S108]. In most cases, shapes can approximately be reconstructed from their corners. 3D reconstruction from images is a common issue of several research domains.

More and more applications are using computer generated models. In many cases, models of existing scenes or objects are desired. Creating photorealistic 3D models of a scene from multiple photographs is a fundamental problem in computer vision and in image based modeling. The emphasis for most computer vision algorithms is on automatic reconstruction of the scene with little or no user interaction. The basic idea behind 3D model reconstruction, from a sequence of un-calibrated images, can be defined in several steps [S107]: first, we need to relate the images in the whole sequence then extract information based on pixel correspondences to be able to apply methods of epipolar geometry.

In real-life image sequences, certain points are much better suited for automated matching than others. The environments of these points contain significant intensity variations and are therefore easy to differentiate from others. The correspondence between such points of interest has to be done by some kind of matching procedure. A possible approach to select points of interest is corner detection. Corners in a scene are the end points of the edges. As we know, edges represent object boundaries and are very useful in 3D reconstruction of a scene.

There are two important requirements for the selection of interesting points. First, points corresponding to the same scene point should be extracted consistently over the different views. Secondly, there should be enough information in the environment of the points, to be able to automatically match the corresponding points [s116]. Corner points are good candidates from both points of view, because they are usually "easily" detectable and identifiable and, furthermore, may have a "characteristic" environment, which all increase the chances for matching the corresponding points in other images [s9].

There are several known corner detection algorithms for the estimation of the corner points. These detectors are based on different algorithm-specific principles. It is known that there are corner detectors, whose functionality is based on a so-called feature orientation matrix $\underline{\underline{L}}(x,y)$, which utilizes the local structure matrix $\underline{\underline{L}}_s(x,y)$ consisting of the first partial derivatives of the intensity function:

$$\underline{\underline{L}}(x,y) = G(x,y) * \underline{\underline{L}}_S(x,y) = G(x,y) * \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}, \qquad (2.1)$$

where $G(x,y)$ corresponds to the a Gaussian smoothing function (see (2.9)) and * stands for the convolution operation. (The idea behind is that corners are local image features characterized by locations where the variations of the intensity function *I(x,y)* are high both in directions *x* and *y* (i.e. both partial derivatives I_x and I_y are large) anyhow the main axes of the coordinate system are chosen. Examples of it are the Harris feature point detector [6] and Förstner's method [7].

Harris' method evaluates a comparison: the measure of the corner strength

$$R_H = \det(\underline{\underline{L}}(x,y) - k(\mathrm{trace}(\underline{\underline{L}}(x,y))), \qquad (2.2)$$

is compared to an appropriately chosen constant threshold. If R_H exceeds the threshold, the point is taken as a corner. Here, $\mathrm{trace}(\underline{\underline{L}}(x,y)) = \lambda_1 + \lambda_2$, λ_1, λ_2 stand for the eigenvalues of matrix

$\underline{\underline{L}}(x,y)$, and k denotes a parameter effecting the sensitivity of the method (typical values for k are $k \in [0.04 - 0.2]$).
Förstner determines the corners as the local maxima of the beneficial function $H(x, y)$

$$H(x,y) = \frac{\det(\underline{\underline{L}}(x,y))}{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}. \tag{2.3}$$

A further well-known corner detector is the SUSAN (Smallest Univalue Segment Assimilating Nucleus) detector based on brightness comparison [8]**.** The algorithm does not depend on image derivatives; it uses the brightness values of the pixels. The first step of the algorithm is to place a circular mask around the pixel in question (the nucleus). After this, the method calculates the number of pixels within the circular mask which have similar brightness values to the nucleus. (These pixels define the so-called USAN.). The next step is to produce the corner strength image by subtracting the USAN size from a given geometric threshold. The possible false positives can be neglected by finding the USAN's centroid and its contiguity. The so called USAN area reaches a minimum (SUSAN), when the nucleus lies on a corner point. This method is more resistant to image noise than the previous ones.
The above, most well known algorithms all apply the following idea: the processed image point is detected as a corner when the calculated value of a certain feature (which is characteristic for a corner) exceeds a given, *constant* threshold. The effectiveness of these methods from corner detection point of view is acceptable. However, modern signal/image processing methods need to fulfill certain new requirements as well.
A new requirement set against signal and image processing algorithms (or in more general, against all pre-processing algorithms) is that they have to give an efficient support to further processing and to autonomous operation of the whole procedure [S9], [S119]. (This requirement has been defined by the author and has been accepted by the international research community.) The above summarized standard methods usually fail to satisfy these requisites.
In the following three subsections, simple procedures (noise elimination, smoothing and creation of the local structure matrix) are summarized, which serve as building blocks of the new introduced corner detectors. This is followed by the presentation of the methods themselves in 2.2.5, and their comparison to existing algorithms in 2.2.6.

2.2.2 Noise Elimination

Before starting to search for the corner points of an image, it is necessary to eliminate the noise. For this purpose we use FIRE filters, a special fuzzy system characterized by an IF-THEN-ELSE structure and a specific inference mechanism proposed by Russo [9]. Different noise statistics can be addressed by adopting different combinations of fuzzy sets and rules. In the followings, we will present a simple FIRE filter removing impulse noise.
Let $I(\mathbf{r})$ be the pixel luminance at location $\mathbf{r}=[x,y]$ in the noisy image, where x is the horizontal and y the vertical coordinate of the pixel. Let $I_0=I(\mathbf{r}_0)$ denote the luminance of the input sample having position $\mathbf{r}_0$ ($\mathbf{r}_0=[x_0,y_0]$) and being smoothed by a FIRE fuzzy filter. The input variables of the fuzzy filter are the amplitude differences defined by:

$$\Delta I_j = I_j - I_0, j = 1,...,8 \tag{2.4}$$

where $I_j=I(\mathbf{r}_j)$, j=1,…,8 values are the luminance values of the neighboring pixels of the actually processed pixel $\mathbf{r}_0$ (see the left side of Fig. 2.1). Let K_0 be the luminance of the pixel having the same position as $\mathbf{r}_0$ in the output image. This value is determined by the following relationship:

$$\textit{new value} = \textit{old value} + \textit{correction},$$

$$K_0 = I_0 + \Delta I \tag{2.5}$$

where ΔI is determined in (2.8).

Let $W = \bigcup_{i=1}^{9} W_i$ be defined by a subset of the eight neighboring pixels around $\mathbf{r}_0$ belonging to a 3x3 moving window (see the right hand side of Fig. 2.1). Let the rule base deal with the pixel patterns $W_1, \ldots, W_9$. Value K_0 can be calculated, as follows [9], [1]:

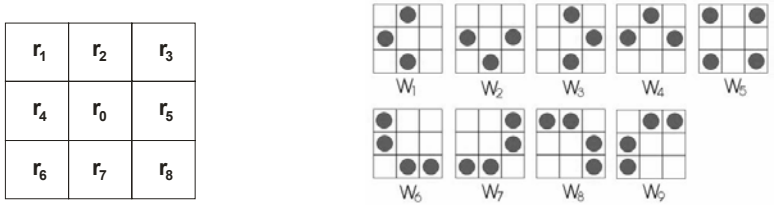


Fig. 2.1 The neighboring pixels of the actually processed pixel $\mathbf{r_0}$ (left) and the pixel patterns (right)

$$\lambda = MAX\left\{MIN\left\{m_{LP}(\Delta I_j) : r_j \in W_i\right\}; i = 1,\ldots,9\right\} \tag{2.6}$$

$$\lambda^* = MAX\left\{MIN\left\{m_{LN}(\Delta I_j) : r_j \in W_i\right\}; i = 1,\ldots,9\right\} \tag{2.7}$$

$$\Delta I = (L-1)\Delta\lambda, \qquad K_0 = I_0 + \Delta I \tag{2.8}$$

where $\Delta\lambda=\lambda-\lambda^*$, L is the maximum of the gray level intensity, m_{LP} and m_{LN} correspond to the membership functions large negative and large positive, and $m_{LP}(I)=m_{LN}(-I)$ (see Fig. 2.2). The filter is recursively applied to the input data.

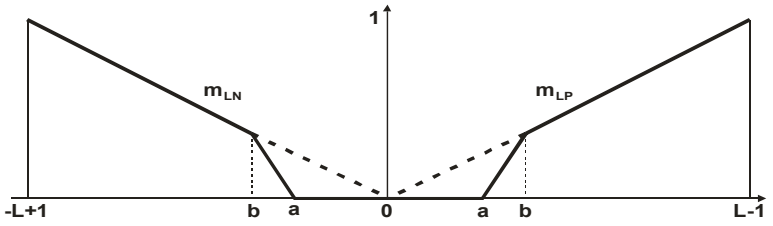


Fig. 2.2 Membership functions m_{LN} (large negative) and m_{LP}, (large positive), a and b are parameters for the tuning of the sensitivity to noise of the filtering

Here we would like to remark that the fuzzy sets above are suitable for removing impulse noise which is the most typical noise type in case of images. Although, FIRE filters can be used for removing other types of noise, as well, however in these cases different fuzzy sets have to be applied. A further advantageous feature of these filters is that the fuzzy sets can be combined, i.e. different noise statistics can be addressed simultaneously.

2.2.3. Gaussian Smoothing

The algorithm uses a convolution kernel of size NxN that represents the shape of a Gaussian hump. This kernel has special properties detailed below. A circularly symmetric Gaussian hump has the form of

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.9}$$

where x and y stand for the 2D coordinates of a point and σ is a shaping parameter of the distribution (see Fig. 2.3) [10]. The main idea of Gaussian smoothing is to use this 2D distribution as a 'point-spread' function which can be achieved by convolution. Since digital images are stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution [10]. An example of this distribution can be seen in Table 2.1.
The idea of Gaussian smoothing is to use this 2-D distribution as a „point-spread" function achieved by convolution.

0.0003	0.0023	0.0062	0.0062	0.0023	0.0003
0.0023	0.0168	0.0458	0.0458	0.0168	0.0023
0.0062	0.0458	0.1244	0.1244	0.0458	0.0062
0.0062	0.0458	0.1244	0.1244	0.0458	0.0062
0.0023	0.0168	0.0458	0.0458	0.0168	0.0023
0.0003	0.0023	0.0062	0.0062	0.0023	0.0003

Table 2.1 Gaussian 6x6 convolution kernel with σ=1

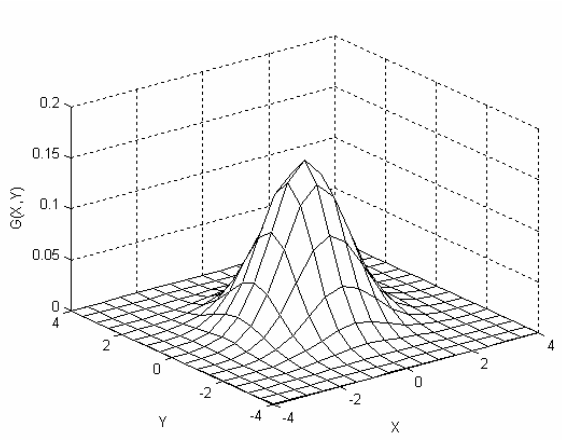


Fig. 2.3 2D Gaussian distribution function with σ=1

The convolution is implemented as follows:

```
for(yo=n; y0<=ymax-n; yo++)
for(xo=n; xo<=xmax-n; xo++)
        {
        newvalue=0
        for(y=-n; y<=n; y++)
        for(x=-n; x<=n; x++)
                newvalue=newvalue+g(x,y)·f(x+xo,y+yo)
                newvalue=newvalue/S
         },
```

where x_o, y_o stand for the identifiers of the element on which we want to execute convolution (in our case these correspond to the 2D coordinates of the analyzed pixel), x and y denote the relative positions of the kernel points, S is the sum of the kernel values, $g(x,y)$ represent the weighting factors of the convolution kernel, and f stands for the function which has to be smoothed.

2.2.4 Determination of the local structure matrix

The local structure matrix $\underline{\underline{L_S}}(x,y)$ is composed of the first derivatives of the intensity function $I(x,y)$ (see (2.1)). The calculation of the first derivatives of $I(x, y)$ can be solved by applying the following convolution masks at each image point:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ for determining } \frac{\partial I}{\partial x}, \text{ and } \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ for determining } \frac{\partial I}{\partial y}.$$

For increasing the effectiveness of the corner detection algorithm it is proposed to smooth each of the entries of matrices I_x^2, I_y^2, I_xI_y, in (2.1) by applying a Gaussian convolution kernel, see 2.2.3.

2.2.5 The new corner detection methods

In this section two new corner detection algorithms are proposed that on one hand out perform the previously used corner detectors, while on the other hand fulfill also the previously mentioned new requirements.
Method 2.1 improves the Harris corner detection method by introducing a fuzzy measure for the determination of points being corners.

Method 2.1 ([S108], [S123]): The steps of the corner detection method are
Step 1. Noise smoothing by Russo's fuzzy filters (subsection 2.2.2)

Step 2. Determination of the local structure matrix $\underline{\underline{L_S}}(x,y)$ (subsection 2.2.4)

Step 3. Application of a convolution mask for the determination of the elements of Förstner's feature orientation matrix $\underline{\underline{L}}(x,y)$ ((2.1) and subsection 2.2.3).

Step 4. Determination of the values of the beneficial function $H(x, y)$ ((2.3)).

Step 5. Determination of the fuzzy membership values of the pixels representing the strength of being corner. For this, fuzzy reasoning is introduced which is applied to the calculated values $H(x, y)$. By the score of the membership function (see Fig. 2.4) of fuzzy set "corners" $m_c(H)$, we can determine a weighting factor, which characterizes the rate of the corner's membership. The value of the membership function is 1 for those image points for which the calculated value H equals or is larger than a given threshold value. With the help of parameters p, q we can modify the shape of the membership function and, thus change the sensitivity of the detection.
Step 6. The output of the detection is yielded by the following relation:

$$z_{i,j} = (L-1) * m_c(H), \qquad (2.10)$$

where $z_{i,j}$ represent the gray-level intensity values of the output image, L stands for the largest intensity value (e.g. for 8 bit gray-scale images L=255), and H denotes the calculated $H(x,y)$ values.

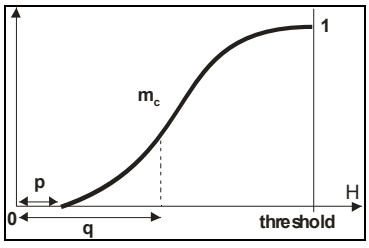


Fig. 2.4 Membership function of fuzzy set "corner" (m_c). Axis **H** is the universe of the calculated $H(x,y)$ values

Step 7. Finally, if the detected corners are neighbors then we have to keep only the corner with the largest calculated value $H(x,y)$. The others should be ignored to avoid multiple detection of an, in practice, single corner.

Method 2.2 starts off the basics of the fuzzy based corner detection algorithm described in Method 2.1. In addition, it applies an image smoothing procedure in the preprocessing phase, furthermore its performance is improved by introducing a fuzzy based technique assigning new attributes to the detected corner candidates (showing the membership value of being a 'real' corner). This latter property of the detector is very advantageous for the matching of corresponding points in stereo image pairs and thus it results in a better output.

Method 2.2 ([S9], [S125]):
Step 1. Noise smoothing by Russo's fuzzy filters (Subsection 2.2.2)

Step 2. Gaussian smoothing of the filtered image (Subsection 2.2.3). This new step improves the performance of the detection significantly. In corner detection, besides the noise, a further problem can occur because the digital image is stored as a collection of discrete pixels. An edge is represented as a series of points possibly resulting in small brakes in the edge which, in many of the cases, causes that false corners appear during the detection. In Fig. 2.5, the left image illustrates how a line looks like (producing false corners) when the resolution of the image is finite. For improving the performance of the corner detection algorithm, the false corners should be eliminated before applying the corner detector. For this purpose a Gaussian smoothing algorithm can be implemented, which is usually used to 'blur' images and to remove unimportant details and noise. In Fig. 2.5 the right image shows how a line after smoothing appears in the image.

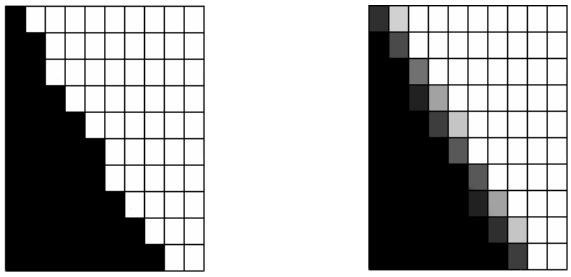


Fig. 2.5 Edge representation without smoothing (left) and after applying a smoothing algorithm (right)

Step 3. determination of the local structure matrix $\underline{\underline{L}}_S(x,y)$ (Subsection 2.2.4)

Step 4. Application of a convolution mask for the determination of the elements of Förstner's feature orientation matrix $\underline{\underline{L}}(x,y)$ ((2.1) and subsection 2.2.3). $\underline{\underline{L}}(x,y)$ can also be derived from locally approximating the autocovariance function of a real valued stochastic signal $I(x,y)$ (generated by a stochastic process) in the origin [11].

Step 5. Determination of the values of the beneficial function $H(x, y)$.

Step 6. Determination of the membership values of the pixels representing the strength of being corner.
In most of the cases, we can not unambiguously determine whether the analyzed image point is a corner or not based only on a certain concrete threshold value. Therefore, in the proposed algorithm fuzzy techniques are applied for the calculation of the values (corners) significantly increasing the rate of correct corner detection. The higher the calculated H is, the higher the membership value representing that the analyzed pixel is a corner becomes. After fuzzifying the H values into fuzzy sets and applying a fuzzy rulebase we can evaluate the "degree of corner-ness" of the analyzed pixels. This attribute of the pixels can advantageously be used in further

processing, e.g. when searching for the corresponding corner points in stereo image pairs, as well [S25]. (Point correspondence matching is an indefinite step of automatic 3D reconstruction. The consideration of an additional feature, i.e. the similarity of the degree of corner-ness of the projections of a certain point in different pictures taken from near camera positions, can highly increase the reliability of the decision.)
The antecedent and consequent fuzzy sets of the detector are illustrated in Figs. 2.6 and 2.7, respectively. In Fig. 2.6 (antecedent fuzzy sets) the horizontal axis represents the universe of the ***H*** values with three fuzzy sets, C_{WEAK}, C_{MEDIUM}, and C_{STRONG} corresponding to points being WEAK, MEDIUM, and STRONG corners, respectively. Parameters H_k (k=1,2,3,4,5) serve for the shaping of membership functions $\mu_{C_{WEAK}}$, $\mu_{C_{MEDIUM}}$, $\mu_{C_{STRONG}}$ by which the sensitivity of the described detector can be tuned.

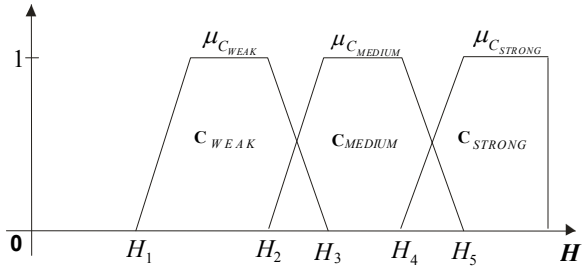


Fig. 2.6 Illustration of antecedent fuzzy sets C_{WEAK}, C_{MEDIUM}, and C_{STRONG} of universe ***H***. The values H_k (k=1,2,3,4,5) serve for shaping membership functions $\mu_{C_{WEAK}}$, $\mu_{C_{MEDIUM}}$, $\mu_{C_{STRONG}}$, i.e. for tuning the sensitivity of the detector

In Fig. 2.7 (consequent fuzzy sets) the horizontal axis is the axis of universe ***I*** (output intensity) also with three fuzzy sets, I_{LOW}, I_{MEDIUM}, and I_{HIGH}. If the pixel is not at all a corner (none of the fuzzy rules are fired) then its intensity will be set to zero, while in other cases the output intensity showing the degree of cornerness will be evaluated by the aggregation of the following fuzzy rulebase:

If ($H(x,y)$, C_{WEAK}) then ($I(x,y)$, I_{LOW}),
If ($H(x,y)$, C_{MEDIUM}) then ($I(x,y)$, I_{MEDIUM}),
If ($H(x,y)$, C_{STRONG}) then ($I(x,y)$, I_{HIGH}),

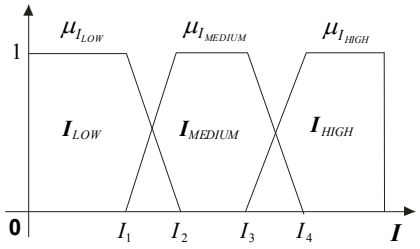


Fig. 2.7 Illustration of consequent fuzzy sets I_{LOW}, I_{MEDIUM}, and I_{HIGH} of universe ***I*** (output intensity). Values I_k (k=1,2,3,4) serve for shaping the membership functions $\mu_{C_{LOW}}$, $\mu_{C_{MEDIUM}}$, and $\mu_{C_{HEIGHT}}$

which means that if the $H(x,y)$ value is member of the fuzzy set C_{WEAK}, C_{MEDIUM}, or C_{HIGH} then the output intensity of the pixel is low, medium, or high, respectively. Let μ (.) be the

membership function of the consequent fuzzy set generated as the superposition of the rule consequents. As defuzzification algorithm we use the center of gravity method, thus the intensity value of a pixel in the output image is obtained by

$$I_o(x,y) = \frac{\sum_{i=1}^{L} \mu(I_i) I_i}{\sum_{i=1}^{L} \mu(I_i)}, \qquad (2.11)$$

where $I_o(x,y)$ denotes the intensity value of the pixel in the output image at position $[x,y]$ and L stands for the maximum of the intensity.

2.2.6 Comparison of different corner detection methods

While the new algorithms are for sure advantageous for further processing and automation, we also performed a comparison to demonstrate their effectiveness. We have made several (appr. 35) simulations and tested the four methods by running them on different pictures partly taken from the literature (like the famous "Lena" photo) and partly on photos chosen by us because we thought them characteristic from corner detection point of view. For simplicity, we have processed gray scale images, with maximum intensity $L = 255$.
Before starting with corner detection, we have applied the same noise smoothing (typically a FIRE filter with a=66 and b=100) in each case. As an example of the results, we include here a "typical" running result in Table 2.2, where the parameters of the different corner detectors were set as follows:

- *Method 2.2:* smoothing: by 2x2 Gaussian hump; fuzzy set: for the comparative runs, we have applied only one fuzzy set with threshold t=161 and tg α=1/544 (see Fig. 2.8) and the membership value corresponds to the strength of being a corner. This simplification can be accepted here because the aim of the illustration is only to show the performance of the corner detection and not to use it for further processing, e.g point correspondence matching. The threshold is set as in case of the Förstner's method to make the comparison easier.
- *Förstner's method*: threshold=161.
- *Harris corner detector*: k=0.15, threshold=5000.
- *SUSAN corner detector:* brightness threshold=10 (the maximum difference in grey levels between two pixels which allows them to be considered part of the same "region" in the image), geometric threshold=37 pixel fixed mask.

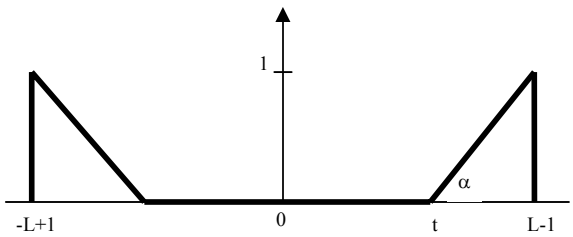
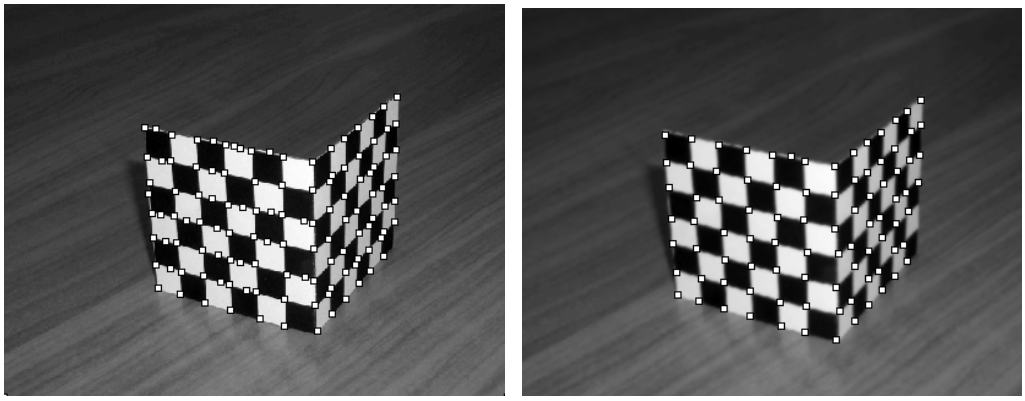


Fig. 2.8 Fuzzy set "corner"

As illustration, we include two very simple examples to show the effectiveness of the new method. For more details and examples, see [S9].

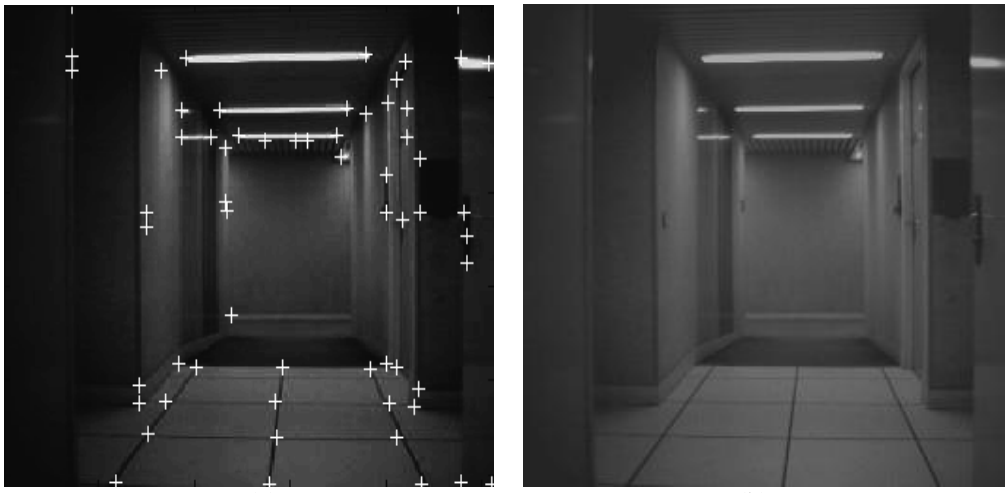
Fig. 2.9 presents results got using the new corner detectors (a) by and (b) without image smoothing. The comparison in this figure illustrates very well the improvement of the results after applying smoothing. (Here we would like to remark the following: In the right hand side image you can find a detected corner not located in a grid point which at first glance could be thought as false detection. However, at closer look we have found that during the cutting out of the check pattern we have made a corner by the scissors, i.e. the detection is correct.)



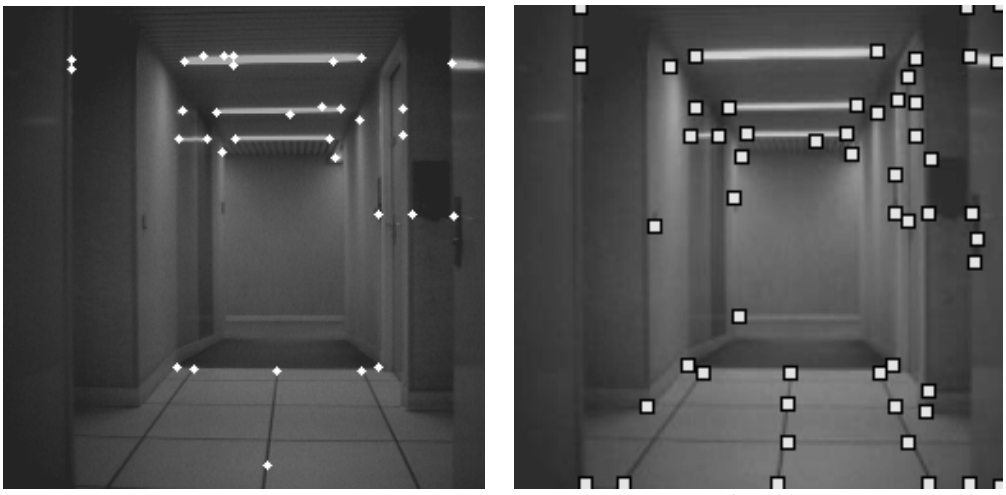
(a) (b)

Fig. 2.9 Detected corners in the fuzzy filtered image (a) using fuzzy based detector without image smoothing, (b) using the same detector but with image smoothing

The next example serves for the comparison of different corner detection algorithms. In case of all methods, the same fuzzy filter was applied for noise removal. Fig. 2.10 (a) shows a part of a corridor with several lamps and doors. In Fig. 2.10 (b) the corners detected by the introduced new fuzzy supported algorithm (method 2.2) can be seen. By analyzing the results we can see that all the corners are detected and no false corner was found. Figs. 2.10 (c)-(e) illustrate the results obtained by the Förstner's, Harris, and SUSAN corner detection algorithms, respectively.



(a) (b)



(c) (d)

(e)

Fig. 2.10 Corridor – (a) Original photo, corner detection applying (b) the proposed new fuzzy based corner detection algorithm, (c) Förstner's corner detector, (d) the Harris corner detector (k=0.001), (e) SUSAN detector

Table 2.2: Comparison of different corner detection methods

Methods \ Corners	Correct [%]	False [%]	Non detected [%]
Fuzzy based corner detector (2x2 Gaussian hump, a=100, b=161, tg β=1/544)	84	0	16
Förstner's method (threshold=161)	78,7	0	21.3
SUSAN corner detector (brightness threshold=10, geometric treshhold=37 pixel fixed mask)	52	4,7	48
Harris corner detector (k=0.15, threshold=5000)	71	15.3	29

2.3 "Useful" information extraction

Recently, in digital image processing a large amount of research has been focused on information retrieval and image understanding. Typical examples are searching for similar objects/images in large databases and understanding the objects in images. The main point of these tasks is to extract the most characteristic features of the objects in the images, like edges, corners, characteristic textures, etc. Another very important aspect can be the separation of the "significant" and "unimportant" parts of these features, i.e. the enhancement of those features which carry primary information and to filter out the part, which represents information of minor importance. By this, the complexity of the searching and/or interpreting algorithms can be decreased while the performance is increased. This chapter describes a new edge processing method which is able to extract the "primary" ones, i.e. those edges which can advantageously be used in sketch based image retrieval algorithms.

2.3.1 Introduction

The recent tremendous growth in computer technology parallel with the appearance of new advances in digital image processing has brought a substantial increase in the storage and aims of digital imagery. On one hand, this means the explosion of database-sizes, while on the other hand the increasing complexity of the stored information calls for new, intelligent information managing methods. To address this challenge, a large number of the digital image processing algorithms which have been introduced in the past years apply soft computing and/or intelligent techniques. All of these algorithms aim some kind of (intelligent) feature extraction supporting the further, more advanced processing, like object recognition, image understanding, image information retrieval, etc. in a single photo or in (large) data bases.

A typical problem of the above type is searching for similar objects/images in large databases. Usually, this process is very time consuming, thus manual searching is not acceptable. A large amount of effort was put on the automation of the procedure. As a result, numerous methods of different kinds were developed.

Some of the methods are based on the description of the images using text attributes, enabling the organization of images by topical or semantic hierarchies to facilitate easy navigation and browsing based on standard boolean queries [12], [13]. Automatically generating descriptive texts for a wide spectrum of images is not feasible, most text-based image retrieval systems require manual annotation of images. Obviously, annotating images manually is an expensive task for large image databases, and is often subjective, context-sensitive, and incomplete [12], [13]. Because of this reason, searching procedures based on image content analysis have been developed, which can select the images more effectively as the text based retrieval methods.

The possibly most interesting and important step in image retrieval is the extraction of the "useful" features from the images. There are several characteristic attributes of the images (e.g. the edges and corners) which carry useful information and can be of help during the extraction of the primary information by appropriate techniques.

The edges in an image can advantageously be used when comparing two images and searching for similar objects [s11]. An image usually contains a lot of different edges, among which there are texture edges and object contour edges. From the point of view of image retrieval, only the latter ones are important because they carry the primary information about the shape of the objects. By considering both types of edges during the search/comparison, the complexity/ time need of the procedure might dramatically, and the (probably high number of) non-important details (edges) might lead to false decisions. As a consequence, it is of key importance to separate the "significant" and "unimportant" subsets of the edges, i.e. to enhance the ones which correspond to the object boundaries and thus carry primary information, but to filter out the others which represent information of minor importance

In this chapter a new primary edge extraction method is introduced, which applies surface deformation combined with fuzzy edge detection technique. In 2.3.2 a standard method of

surface smoothing is described, while in 2.3.3 an existing edge detection method due to Russo is summarized. The novel method of the author is then presented in 2.3.4.

2.3.2 Surface smoothing

Let S_t be the surface describing an image to be processed, i.e. $S_t = \{(x, y, z); z = I(x,y,t)\}$, where variables x and y represent the horizontal and vertical coordinates of the pixels, z stands for the luminance value, which is the function of the pixel coordinates and of time t. Smoothing is performed by image surface deformation. Such a process preserves the main edges (contours) in the image. The surface deformation process satisfies the following differential equation [14]:

$$\frac{\partial I_t}{\partial t} = k\underline{n}\,, \tag{2.12}$$

where k corresponds to the „speed" of the deformation along the normal direction $\underline{n}$ of the surface S_t. In our case, value k is represented by the mean curvature of the surface at location [x, y], i.e. the speed of the deformation at a point will be the function of the mean curvature at that point. The mean curvature is defined as

$$k = \frac{k_1 + k_2}{2}\,, \tag{2.13}$$

where k_1 and k_2 stand for the principal curvatures. Starting from equation (2.13), the following partial differential equation can be derived (Because of the limitations on the volume, we skip the details of the deduction. For details, see [15]):

$$k = \frac{(1+I_y^2)I_{xx} - 2I_xI_yI_{xy} + (1+I_x^2)I_{yy}}{2\left(1+I_x^2+I_y^2\right)^{3/2}}\,. \tag{2.14}$$

Here I_x, I_y, I_{xx}, I_{xy}, I_{yy} stand for the partial derivatives with respect to the variables indicated as lower indices. Starting from equation (2.12) the surface at time $t+\Delta t$ (for small Δt) can be calculated as follows [14]:

$$I(x,y,t+\Delta t) = I(x,y,t) + k\sqrt{1+I_x^2(x,y,t)+I_y^2(x,y,t)}\Delta t + o(\Delta t) \tag{2.15}$$

where $o(\Delta t)$ represents the error of the approximation.
Fig 2.11 illustrates the virtual process of the surface deformation along the time.

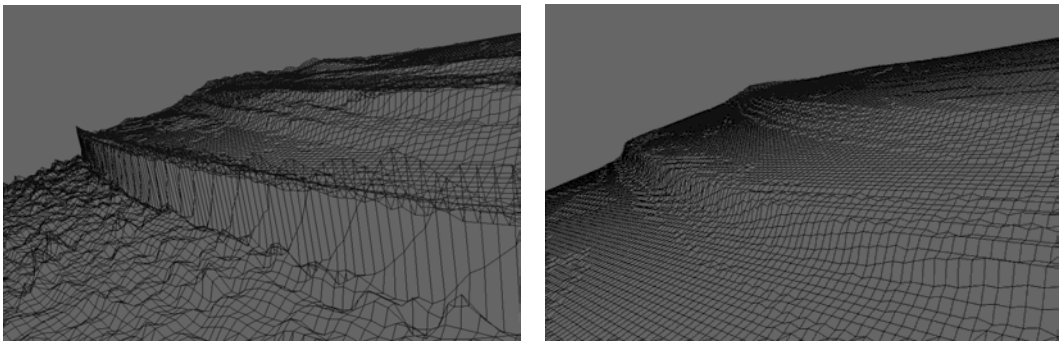


Fig. 2.11 Illustration of an image surface before (left) and after (right) the deformation

2.3.3 Edge detection

In the followings, the fuzzy edge detection method of Russo [1] is summarized. The idea of the method is very similar to that of the noise smoothing algorithms presented in Subsection 2.2.2.

Let $z_{0,x,y}$ be the pixel luminance at location $[x,y]$ in the original image. Let us consider the group of neighboring pixels which belong to a 3x3 window centered on $z_{0,x,y}$. The output of the edge detector is yielded by the following equation:

$$z^{p}_{x,y} = (L-1)MAX\{m_{LA}(\Delta v_1), m_{LA}(\Delta v_2)\}$$
$$\Delta v_1 = \left| z_{0,x-1,y} - z_{0,x,y} \right| \qquad (2.16)$$
$$\Delta v_2 = \left| z_{0,x,y-1} - z_{0,x,y} \right|$$

where $z^{p}_{x,y}$ denotes the pixel luminance in the edge detected image and m_{LA} stands for the used membership function (see Fig. 2.12). $z_{0,x-1,y}$ and $z_{0,x,y-1}$ correspond to the luminance values of the left and upper neighbors of the processed pixel at location $[x,y]$. L-1 equals to the maximum luminence value (e.g. 255). For more details see [1].

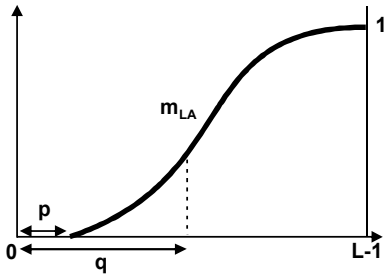


Fig. 2.12. Fuzzy membership function m_{LA} of "edge". L-1 equals to the maximum intensity value, p and q are tuning parameters

2.3.4 The new primary edge extraction method ([S11], [S139])

Assume that we have an image and we want to extract the edges corresponding to the object contours. This can be done with the help of the new primary edge extraction method detailed below.

Prepocessing

Step 1. Surface smoothing (Subsection 2.3.2): As the first step, it is necessary to remove the unimportant details from the image. After smoothing the image, only the most characteristic contours are kept.

Step 2. Constructing the edge map: The edge map of the original (unsmoothed) image is constructed using the fuzzy based edge detection method described in Subsection 2.3.3. Such an edge map contains all the possible edges.

After preprocessing, the new edge separation follows according to Method 2.3

Method 2.3 ([S11], [S139]):

The new method aims to extract the most characteristic edges of images. This is done by a simultaneous analysis of the smoothed image and the edge map of the original image: in case of each of the edge points a small environment of the point is taken in the original image and using the smoothed image the variance of the color components inside this environment is analyzed. If the variance is below a predefined threshold value then the edge point is removed while otherwise it is considered as a useful, primary edge point. The procedure is performed as follows:

Step 1. For each edge point taken from the edge map of the original image, the environment of the point is analyzed in the smoothed image. The analysis is realized by calculating the mean squared deviation of the color components (in case of grayscale images the gray-level component) in the environment of the selected edge point.

Let $\mathbf{p}=[p_x, p_y]$ be an edge point in the original image and let **M** denote a rectangular environment of **p** with width w and height h. The mean squared deviation is calculated as follows:

$$d = \frac{\sum_{i=p_x-w/2}^{p_x+w/2} \sum_{j=p_y-h/2}^{p_y+h/2} (\mu - I(i,j,t_{stop}))^2}{hw}, \tag{2.17}$$

where t_{stop} represents the duration of the surface deformation.
In case of grayscale images, μ denotes the average gray level inside the environment **M**. For color images, the whole process should be done for each component separately and in this case μ corresponds to the average level of this color component inside the environment **M**.

Step 2. If the calculated deviation exceeds a predefined threshold value, then the edge point is considered as useful edge. As result, an image containing only the most characteristic edges is obtained.

For illustrating the behavior of the method, in Figs. 2.13-2.16 an example is presented showing the useful edge extraction procedure. For more examples and details, see [S11], [S139]. As shown in Fig. 2.16, many of the details disappear after the processing and only the characteristic edges of the car are left. This helps filtering out the non-important details and enhancing the most significant features/objects in images, thereby making image retrieval, object recognition, etc. easier.

Fig. 2.13. Original image taken of a car

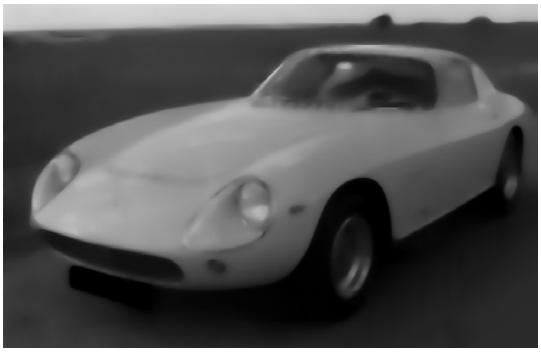


Fig. 2.14. Smoothed image using surface deformation based on mean curvature

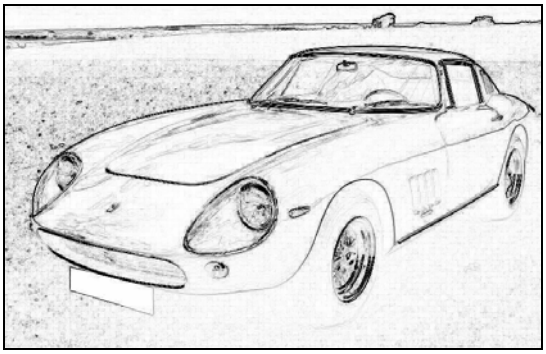


Fig. 2.15. Edge map of the original image

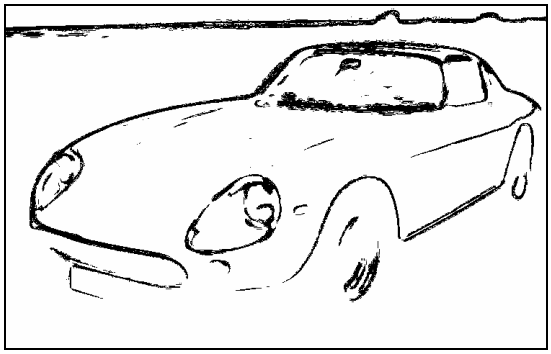


Fig. 2.16. Edges after applying the proposed information enhancement method

2.4 High Dynamic Range (HDR) Imaging

The high dynamic range of illumination may cause serious distortions and problems in the view and further processing of digital images. Important information can be hided in the highly or extreme lowly illuminated parts. This chapter deals with the reproduction of such images and introduces two anchor based and one tone reproduction pre-processing algorithms which may help in developing the hardly or non viewable features and content of the images.

2.4.1 Introduction

Digital processing can often improve the visual quality of real-world photographs, even if they have been taken with the best cameras by professional photographers in carefully controlled lighting conditions. This is because visual quality is not the same as the accurate reproduction of the scene. In image processing, most of the recently used methods apply preprocessing procedures to obtain images which guarantee - from the point of view of the concrete aims – better conditions for the processing. For example, if we eliminate the noise from the images to be processed we can obtain much better results then else or applying different feature extraction methods, like edge and corner detection, may significantly help in pattern recognition.

There are many kinds of image properties to which the certain methods are more or less sensitive [1]. It is also known that in most of the cases certain regions of images have different features while at the same time the parameters of the processing methods are usually functions of the image features. This fact may make difficult to find the optimal parameters for the applied image processing procedure or to correctly interpret the results.

The light intensity at a point in the image is the product of the reflectance at the corresponding object point and the intensity of the illumination at that point. The amount of light projected to the eyes (luminance) is determined by a number of factors: the illumination that strikes visible surfaces, the proportion of light reflected from the surface, and the amount of light absorbed, reflected, or deflected by the prevailing atmospheric conditions (such as haze or other partially transparent media) [16]. Only one of these factors, the proportion of light reflected (lightness) is associated with an intrinsic property of surfaces and hence is of special interest to the visual system. If a visual system only made a single measurement of luminance, acting as a photometer, then there would be no way to distinguish a white surface in dim light from a black surface in bright light. Yet, humans can usually do so and this skill is known as lightness constancy [17]. The constancies are central to perception. An organism needs to know about meaningful world properties, such as color, size, shape, etc. These properties are not explicitly available in the retinal image and must be extracted by visual processing. A gray patch appears brighter when viewed against a dark background and darker when viewed against a bright background. This effect, known as “simultaneous contrast” is one of many brightness effects that are commonly attributed to simple visual processes, such as the lateral inhibition that occurs in the retina, whereby cells in one region inhibit cells in adjacent regions.

In this chapter we deal with a further, relatively new research topic in image processing, namely with the reproduction of images when the high dynamic range of the lightness causes distortions in the appearance and contrast of the image in certain regions e.g. because a part of the image is highly illuminated looking plain white or another is in darkness. It may cause serious problems in the processing and analysis of the view. (Just consider the problem when you are leaving a dark tunnel and would like to be sure that nothing is in front of you, i.e. it is safe to enter or drive to the bright sunshine.) Using such a reproduction algorithm in preprocessing phase of the images, we can make the information hided in the picture attainable, we can avoid information loss, and we can improve the performance of further processing and analysis.

The chapter is organized as follows: Section 2.4.2 deals with the background of tone reproduction, in Section 2.4.3 the principles of anchoring theory are summarized. Section 2.4.4 is devoted to the segmentation of images, while Sections 2.4.5 and 2.4.6 discuss new HDR imaging techniques due to the author: in Section 2.4.5 two new anchoring based reproduction algorithms are described; in Section 2.4.6 another new and promising concept, the so called tone mapping function based algorithm is introduced. In Section 2.4.7 a simple example illustrates the performance of the techniques.

2.4.2 Background

Everybody knows the phenomena: if we switch off the lights at night, at first we cannot see anything at all. But the eye adapts to the new lighting situation and after one minute we begin to detect first objects and after about twenty minutes most features are visible again (although they without color). The same happens if the illumination suddenly changes from dark to bright, but this time the process of adaptation is less noticeable, because it happens much faster: in the first two seconds more than 80 percent of the adaptation is done [18]. The task of a tone reproduction operator is to present the global illumination solution for a scene on the display (monitor, print, etc.) such that it closely matches the impression of an observer of the corresponding real world's scene. The tone reproduction operator has to face two major problems when fulfilling this task: First, it must mimic how the eye would see the real world and the displayed image and find a match between the two impressions. As the second task, it must compress the high dynamic range of the real world to the small dynamic range of the display (e.g. for standard monitors 1 - 120 cd/m2).

The tone reproduction problem was first defined by photographers. Often their goal is to produce realistic "renderings" of captured scenes and they have to produce such renderings while facing the limitations presented by slides or prints on photographic papers. Many common practices were developed over the 150 years of photographic practice [19]. In computer graphics the dynamic range of a scene is expressed as the ratio of the highest scene luminance to the lowest scene luminance. Photographers are more interested in the ratio of the highest and lowest luminance regions where details are visible. This can be viewed as a subjective measure of dynamic range.

Fig. 2.17 illustrates the scale of a high dynamic range image. This scale can be divided into two main parts: the first contains displayable luminance values while the second one contains such high luminance values, which are not displayable by today's devices. Images may contain different parts with different illumination characteristics. Real images often contain regions, which are highly illuminated and much less illuminated parts, as well. The highly illuminated regions can contain non-displayable luminance values. From the point of view of the observer, the image data falling into such luminance intervals are lost. Thus, to avoid the image information loss, it is necessary to map these high luminance values to the displayable range.

There are several known methods in the literature addressing these problems. Each of them tries to compress the high dynamic range of luminance values to the displayable range. Just to mention two of them, Kawahito's method [20] is based on multiple exposure time signals while Reinhards in [21] applies a so called zone system. Recently, the authors of this paper have also introduced new fuzzy supported anchor based and tone reproduction algorithms in [S29], [S31]. We would also like to mention the work of Ukovich et.al. [22], in which a comprehensive tool for the qualitative and visual evaluation of the different techniques is presented.

In this Section three new methods are introduced for anchoring and tone reproduction. The first two are based on anchoring theory and image segmentation. During the development of the second method, the author started from the segmentation technique described in the first method [S29] and kept the fuzzy merging technique introduced also in [S29]. However, the technique was improved by a new anchor estimation algorithm and a new operator for the determination of the display luminance of the output pixels.

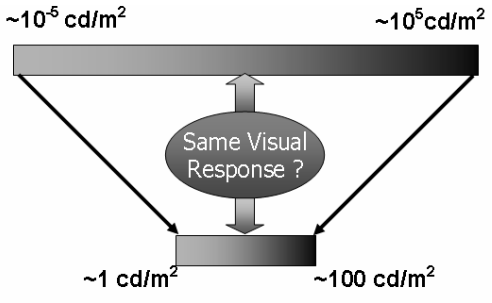


Fig. 2.17 Illustration of the mapping the high dynamic scale to a displayable range

The basis of the third method presented here is the application of a new, modified form of the so called tone mapping function. All three introduced techniques result in an improvement of the preservation of the visual information and in the quality of the reproduction of the input pictures.

2.4.3 Anchoring theory

Although the ambiguous relationship between the luminance of a surface and its perceived lightness is widely understood, there has been little appreciation of the fact that the relative luminance values are scarcely less ambiguous than absolute luminance values [23]. Consider, e.g. a pair of adjacent regions in the retinal image whose luminance values stand in a five to one ratio. This five to one ratio informs the visual system only about the relative lightness values of the two surfaces, not their specific or absolute lightness values. It informs only about the distance between the two gray shades on the gray scale, not about their specific location on that scale. There is an infinite family of pairs of gray shades that are consistent with the five to one ratio. For example, if the five represents white then the one represents middle gray. But the five might represent middle gray as well, in which case the one will represent black. Indeed, it is even possible that the one represents white and the five represents an adjacent self-luminous region. So the solution is not even restricted to the scale of the gray surface. To derive specific shades of gray from relative luminance values in the image, one needs an anchoring rule. An anchoring rule defines at least one point of contact between luminance values in the image and gray scale values along our phenomenal black to white scale. Lightness values cannot be tied to absolute luminance values because there is no systematic relationship between absolute luminance and surface reflectance, as noted earlier. Rather, lightness values must be tied to some measure of relative luminance. The relative lightness of two regions in an image can remain fully consistent with the luminance ratio between them, even though their absolute lightness levels depend on how the luminance values are anchored [23]. The most frequently used rules for anchoring are the followings:

Highest Luminance Rule: The value of white is assigned to the highest luminance in the display and serves as the standard for darker surfaces [24].

Average Luminance Rule: The average luminance rule derives from the adaptation level theory and states that the average luminance in the visual field is perceived as middle gray. Thus, the relative luminance values have to be anchored by their average value to middle gray [23].

There is a tendency of the highest luminance to appear white in the human vision system and also a tendency of the largest area to appear white [17]. Therefore, the highest luminance rule was redefined based on this experimental evidence. As long as there is no conduct, i.e. the highest luminance covers the largest area, the highest luminance becomes a stable anchor. However, when the darker area becomes larger, the highest luminance starts to be perceived as self-luminous. Thus, the anchor is chosen as a weighted average of the luminance proportionally to the covered area.

2.4.4 Fuzzy set theory based segmentation of images into frameworks

The anchoring rule, described in the previous section, cannot be applied directly to complex images in an obvious way. The main concept is based on the decomposition of the image into so called frameworks or segments and then on the application of the anchoring rule in the individual segments, separately. However, the borders of the segment are not unambiguous, thus as an improvement, in the following, a new method is presented for the segmentation, which takes into consideration of this ambiguousness and applies fuzzy reasoning in the segmentation.

Method 2.4 ([S29], [S31], [S121], [S122]):
Step 1. By the segmentation we have to find the so called centroids of the segments using, e.g. the K-means clustering algorithm [25]. The K-means algorithm is initialized by values ranging from the minimum to maximum luminance in the image with a luminance step equal to one order of magnitude and we execute the iterations until the algorithm converges. We operate on a histogram in the 10 based logarithm of luminance. After the K-means algorithm has finished, the centroids we have got are merged according to the following criteria: when the difference between two centroids is below a certain threshold, e.g. one then these centroids have to be merged. This is also done iteratively. In each iteration step the two closest centroids are merged together and the new centroid value is calculated as the weighted average of the two merged centroids proportional to their area:

$$c_{ij} = \frac{c_i a_i + c_j a_j}{a_i + a_j} \tag{2.18}$$

where c_i and c_j are the values corresponding to the centroids with indices i and j, a_i and a_j stand for the number of pixels in the ith and jth frameworks, i.e. they represent the number of pixels corresponding to the ith and jth centroid.

Step 2. It is not an ambiguous step to assign a border to the certain frameworks. We can get better results if we look at the frameworks as fuzzy sets, which means that to each pixel a fuzzy membership value is assigned to define the membership of the pixels belonging to the frameworks [S29]. Thus, as next step we have to estimate the membership functions corresponding to the frameworks. For this, we suggest to use the centroids determined in the previous step by the K-means clustering algorithm.
Let μ_i be the membership function corresponding to the ith centroid (framework) defined as follows (see Fig. 2.18):

$$\begin{aligned} &\mu_i(x,y) = 0 && I(x,y) \le c_{i-1} \\ &\mu_i(x,y) = \frac{I(x,y) - c_{i-1}}{c_i - c_{i-1}} && c_{i-1} < I(x,y) < c_i \end{aligned}, \tag{2.19}$$

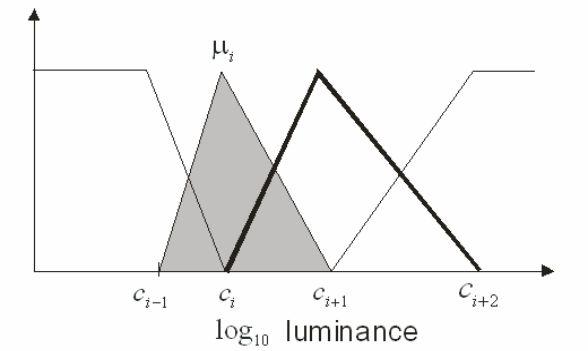


Fig. 2.18 Illustration of the membership function μ_i, which defines the membership of the luminance values in the ith framework

$$\mu_i(x,y) = \frac{c_{i+1} - I(x,y)}{c_{i+1} - c_i} \qquad c_{i+1} \geq I(x,y) \geq c_i,$$
$$\mu_i(x,y) = 0 \qquad I(x,y) \geq c_{i+1} \tag{2.20}$$

The c_i values are the centroids estimated by the K-means clustering algorithm and $I(x,y)$ stands for the luminance value in the log10 space of the pixel at position $[x, y]$.

Step 3. Besides to develop as much details of the images as possible, we also want to keep the "character" of the image. This means that during the determination of the output intensity of the pixel, we don't simply take into account the membership values showing the measure of belonging to the different frameworks. We also apply a so-called articulation factor [26] associated to each framework and modify the membership values by these articulation factors in order to ensure that the "big" frameworks (i.e. having wide intensity ranges) will have a more characteristic influence on the output intensity than the "small" frameworks, i.e., frameworks with low variance have less influence on the global lightness. Thus, as the next step we have to determine an articulation factor for each framework independently. Frameworks with wider intensity range will have a greater weighting factor in the determination of the final luminance value of the output pixel. A framework whose dynamic range is higher than one order of magnitude has a maximum articulation and as the dynamic range goes down to zero, the articulation reaches the minimum:

$$F_i = 1 - e^{-\frac{(\max I_i - \min I_i)}{2s}}. \tag{2.21}$$

Here F_i denotes the articulation factor of the ith framework, max I_i and min I_i correspond to the maximum and minimum values of the intensity within the ith framework, respectively, and s stands for a parameter by which the influence of F_i on the final membership value can be set. After the articulation factors have been estimated, we have to multiply these factors by the corresponding μ_i values to get the final membership value in the frameworks (see (2.21)).

$$\mu_i(x,y) := \mu_i(x,y) F_i, \tag{2.22}$$

where i=1..n, n is the number of frameworks.

2.4.5 Anchor based new algorithms

In the followings two new anchor based algorithm are introduced.

Method 2.5 ([S29], [S121], [S122]):

Step 1. Segmentation of the image into frameworks according to Method 2.4.

Step 2. Anchor estimation: The anchors within each framework are estimated separately based on the so-called log-average luminance of an image.
The log-average luminance is calculated by finding the geometric mean of the luminance values of all pixels [27]. The log-average luminance can usefully be used as the approximation of the anchor of a concrete framework in a scene. This quantity is computed by

$$\bar{I}_i = \frac{1}{N_i} \exp\left(\sum_{x,y \in frm_i} \log\left(\delta + I(x,y)\right) \right), \tag{2.23}$$

where $I(x,y)$ stands for the luminance of pixel $[x,y]$, N_i is the total number of pixels in the image and δ is a small value to avoid singularity occurring if black pixels are present in the image. frm_i denotes the ith framework.

$\bar{I}_i$ is then used to scale the luminance I to the middle grey zone. For this we define a scaling factor for each framework, as follows:

$$L_i = \frac{a_i}{\bar{I}_i}, \tag{2.24}$$

where $\bar{I}_i$ is the estimated anchor for the ith framework and a_i is the key-value to indicate whether the ith framework is subjectively light ("high key") or dark ("low key"). The standard value is 0.18 for normal-key frameworks, but the user can set a_i to 0.09 and 0.045 or to 0.36, 0.72 or 1.0 if the scene makes it necessary.

Step 3. Merging the frameworks. The global lightness of the pixels is computed by merging the frameworks using the following fuzzy rulebase:

If $(I(x,y),\text{fw}_1)$ Then $L_G=L_1$,
If $(I(x,y),\text{fw}_2)$ Then $L_G=L_2$,
If $(I(x,y),\text{fw}_3)$ Then $L_G=L_3$,
…
If $(I(x,y),\text{fw}_n)$ Then $L_G=L_n$,

where $(I(x,y), \text{fw}_i)$ denotes that the luminance $I(x,y)$ is the member of the ith framework (fw_i) with nonzero membership value. i=1..n, n stands for the number of frameworks. L_i represents the estimated anchor corresponding to the ith framework while L_G stands for the global lightness, which is used for shifting the original luminance values to achieve a displayable range of luminance. This value is got after evaluation of the above defined fuzzy rulebase, i.e. as the weighted sum of the local lightness values, where the membership values serve as weighting factors.

Step 4. Finally, a simple global operator is applied to obtain the display luminance:

$$I_d = \frac{I(x,y)L_G}{1+I(x,y)L_G} \tag{2.25}$$

The other anchor based algorithm presented here keeps the segmentation technique described in Subsection 2.4.4 and the fuzzy merging algorithm introduced in Method 2.6 (Step 3). On the other hand, a new anchor estimation algorithm is suggested and a new operator for the determination of the display luminance of the output pixels:

Method 2.6 ([S31], [S126]):

Step 1. Segmentation of the image into frameworks according to Method 2.4.

Step 2. Anchor estimation: The anchors within each framework are estimated separately based on the so-called highest luminance rule. This means that we need to find the luminance value that would be perceived as white. Although, we apply the highest luminance rule, we cannot directly use the highest luminance in the framework as an anchor. Seemingly, there is a relation between what is locally perceived as white and the relative magnitude of its area. If the highest luminance covers a large area it becomes a stable anchor. On the other hand, if the highest luminance is largely surrounded by darker pixels, the light pixels have a tendency to appear white (i.e. lighter than in reality). This is called self-luminance. The opposite is also true, i.e. if the lowest luminance is largely surrounded by light pixels, the (small region of) dark pixels have a tendency to appear black (i.e. darker than in reality). We can decrease the effect (the highest luminance to appear as self-luminous). When estimating the local anchor it is advised to remove a certain amount, e.g. 5% of all pixels in the framework's area that have the highest luminance and then take the highest luminance of the rest of the pixels as the anchor (the 5% is an experimental factor) [26].

Step 3. Merging the frameworks, as is described in Method 2.5, except that the new operator applied in Step 4 gives a different interpretation to the determined lightness values:

If $(I(x,y),\mathrm{fw}_1)$ Then $L_M=L_1$,
If $(I(x,y),\mathrm{fw}_2)$ Then $L_M=L_2$,
If $(I(x,y),\mathrm{fw}_3)$ Then $L_M=L_3$,
…
If $(I(x,y),\mathrm{fw}_n)$ Then $L_M=L_n$,

where $(I(x,y),\mathrm{fw}i)$ denotes that the luminance $I(x,y)$ is the member of the ith framework (fw$i$) with nonzero membership value, i=1..n, n stands for the number of frameworks. L_i represents the estimated anchor corresponding to the ith framework while L_M stands for the lightness modification value, which is used for shifting the original luminance values to achieve a displayable range of luminance. This value is obtained after evaluation of the above defined fuzzy rulebase, i.e. as the weighted sum of the local lightness values, where the membership values serve as weighting factors. Finally,

Step 4. Determination of the display luminance of the output pixels: The lightness values of the output image $I_{res}(x,y)$ are obtained according to

$$I_{res}(x,y) = I(x,y) - L_M \,. \tag{2.26}$$

2.4.6 A new Tone mapping function based algorithm

The method presented in this section solves the task of mapping the high dynamic range of luminance values to a displayable one by maintaining the complexity so low, that real time processing can be achieved, as well.

Method 2.8 ([S31], [S126]):

The main idea lies on defining a simple mapping function, which maps the wide range of luminance values to a displayable one.
Besides it, the simple and easily evaluable mapping function (having low complexity) can be combined with a nonlinear scaling of the vertical axis (the axis of displayable luminance values), as well thus extending the mapping possibilities. The nonlinear vertical axis on one hand enables to keep the image data, which should not be modified, invariable while on the other hand the high or less illuminated areas can be corrected without having any influence on the areas containing correct image data. Furthermore, the nonlinear mapping function makes possible to compress regions where unimportant or sparse information is stored thus offering a way to keep wider parts of the displayable or viewable domain for the important (dense) regions. The importance of the regions can be measured easily and automatically by the magnitude of "strong" intensity changes (e.g. density of the edges) within the region which is characteristic for the density of the represented amount of (seen or hided) information in this region (see Section 2.5). The displayable luminance region can be allocated proportional to this measure, i.e. if we have an important (dense) high dynamic region, we can modify the originally corresponding region (assigned according to the log function) proportional to the characteristic information dense measure. The mapping will keep the relativity of the luminance, i.e. lighter regions will remain lighter while darker regions darker. Fig. 2.19 illustrates a possible mapping function and a simple nonlinear vertical axis of the displayable luminance values. The nonlinearity of the vertical axis is influenced by a set of linear functions. By changing the linear functions the nonlinear characteristics of the vertical axis can also be modified.
The mapping function in Fig. 2.19 has the form of:

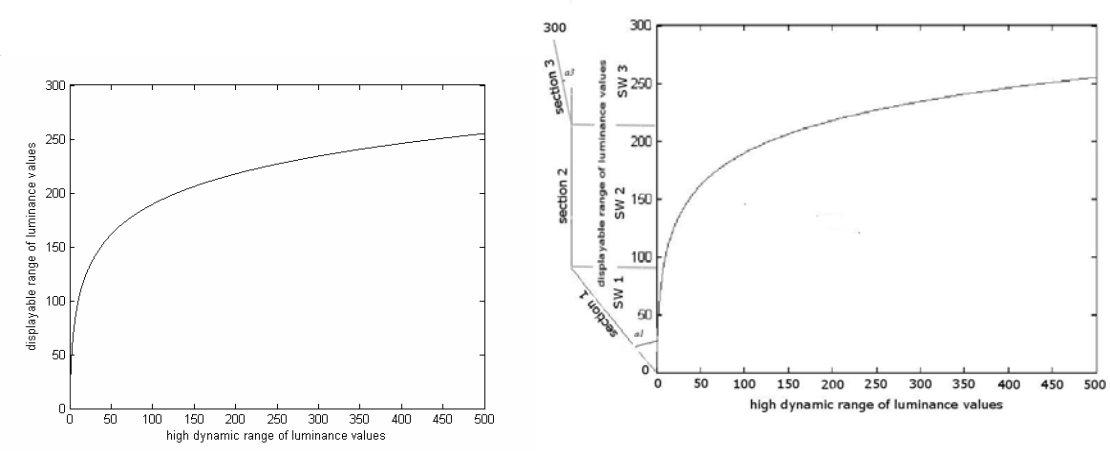


Fig. 2.19 Example of the most frequently used logarithmic mapping function (left) and the proposed new complex function: the logarithmic mapping combined with nonlinear (or piecewise linear) vertical axis (bottom)

$$\begin{aligned}
&\text{if } \ 0 \le \log(L_w) \le SW1 && \text{then } \ L_d = \log(L_w)/\cos\alpha_1 \\
&\text{if } \ SW1 < \log(L_w) \le (SW1 + SW2) && \text{then } \ L_d = \log(L_w)/\cos\alpha_2 \\
&\text{if } \ (SW1 + SW2) < \log(L_w) \le (SW1 + SW2 + SW3) && \text{then } \ L_d = \log(L_w)/\cos\alpha_3
\end{aligned}$$

$$SW1/\cos\alpha_1 + SW2/\cos\alpha_2 + SW3/\cos\alpha_3 = L_{d\max} \qquad (2.27)$$

where L_w stands for a wide range luminance value, SWi represents the width of the ith section (logarithmic scale), L_d denotes the displayable luminance value (luminance value in the resulted output image, with upper limit $L_{d\max}$) and α_i is the angle between the side of the ith section of the axis and the original vertical axis.

2.4.7 Illustrative example

In the followings, a simple example, an image taken of a satellite, is presented for illustrating the effectiveness of the proposed new methods. For simplicity, we process grayscale images where the displayable range is between 0 and 255. For more examples and details, see [S29] and [S31].

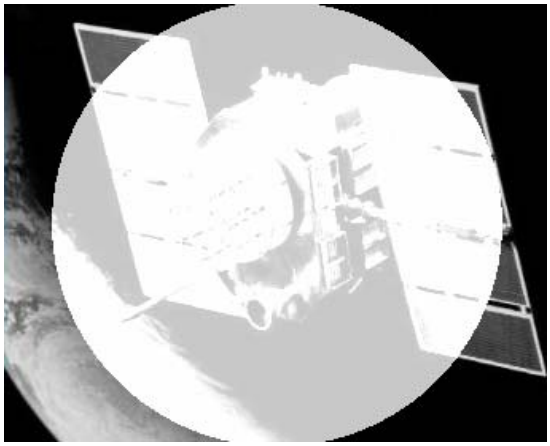


Fig. 2.20 Image of a satellite

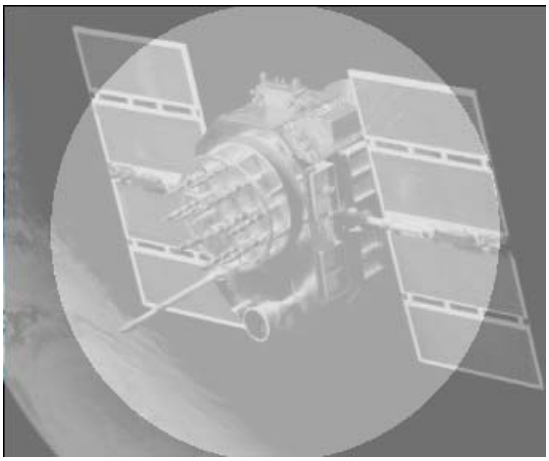


Fig. 2.21 Processed image using Method 2.6

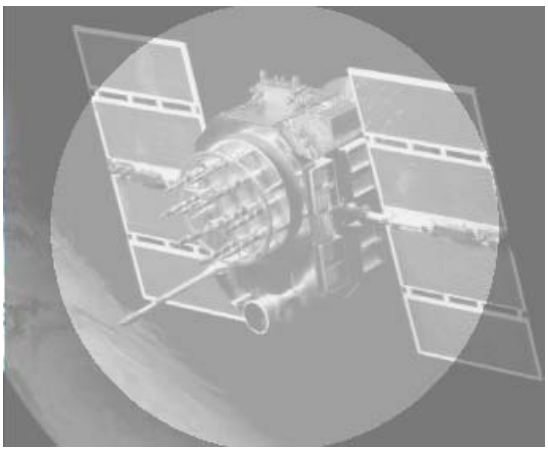


Fig. 2.22 Processed image using Method 2.7

Fig. 2.23 Processed image using logarithmic mapping function with linear vertical axis

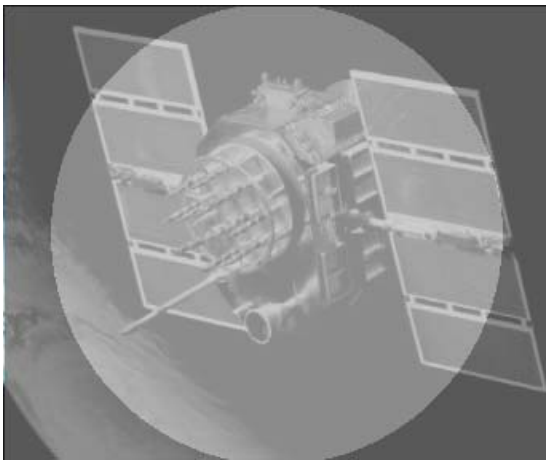


Fig. 2.24 Processed image using Method 2.8

In Fig. 2.20 the original image can be seen where a part of the scene is highly illuminated and the details can badly be recognized. Fig. 2.21 shows the output after using the fuzzy anchor based tone reproduction algorithm presented in Method 2.5, while in Fig. 2.22 the result after applying Method 2.6 can be followed. Fig. 2.23 illustrates the processed image when applying a simple logarithmic mapping function with linear vertical axis. Finally, in Fig. 2.24, the processed image is got by the complex tone mapping function based algorithm (logarithmic mapping and vertical axis) described in Method 2.7.

2.5 Multiple exposure time HDR image synthetization

In many of the image processing tasks, like object recognition and categorization, the color information may have a primary role. In case of complex searching and image understanding tasks, the application of gray scale images is usually not effective enough. The primary aim is to keep and/or enhance the color information, as well because it is useful for categorizing the detected object(s) more precisely, thus the color information may significantly improve the reliability of the decisions.

In this section a new tone reproduction algorithm is introduced which may help in developing hardly or non-viewable features and content of color images. The method is based on the synthetization of multiple exposure images, from which the dense part, i.e. regions having the maximum level of detail are included in the output image. The Red, Green, and Blue color components of the pixels are handled separately and in the output, the corresponding (modified) color components are blended. As a result, a high quality color HDR image is obtained, which contains both all of the detail and the color information. Using HDR color images, the performance of information enhancement, object and pattern recognition, scene reconstruction, etc. algorithms can significantly be improved.
The section is organized as follows: In 2.5.1 the basic concept of the proposed new gradient based multiple exposure time synthesization algorithm is presented. Section 2.5.2 describes how we can measure the level of detail in an image region. Section 2.5.3. is devoted to image synthesization.

2.5.1 Introduction

When the dynamic range of a scene is high, taking just one photo by using a normal camera is not enough for producing a high quality image containing all the information. In such cases, several pictures are needed to capture all the scene details. These images should be merged together in such a way, that all the involved information is preserved.
If the scene contains regions with high luminance values, then it is necessary to take a picture with low exposure time for the visualization of the details in the highly illuminated region. On the other hand, if the scene contains very dark areas, then the exposure time should be possibly much higher. Approaching from the opposite site, if we have images with different exposures we have to decide somehow which exposure contains the maximum level of information in case of a certain region.

2.5.2 Measuring the Level of the Color Detail in an Image Region

There are existing methods, which use statistical elements for measuring the amount of information in images/image regions. Others apply the histogram of the luminance values of the processed region. We propose a new measurement method: to measure the level of the details in a region based on the sum of gradient magnitudes of luminance in that region. The higher sum of the gradient values is in a region corresponds to higher amount of details. This quantity can be computed easily by handling and measuring the RGB components of the region separately and at the end by summing up the component results. The complexity of this approach is significantly lower than that of the other ones, and as we demonstrate below, it provides good results.

Theorem 2.1: The level of detail in an image/image region can be measured by summing up the intensity changes of the RGB components.

Proof:

The amount of information in an image is strongly related to the number and complexity of the objects in the image. The boundary edges of the objects carry the primary information about the object's shape. Thus, image content information can be represented by the characteristic features, like corners and edges in the image, i.e. the number of characteristic pixels is proportional to the amount of information in the image.
Let $I^R(x,y)$, $I^G(x,y)$, and $I^B(x,y)$ be the R, G, and B intensity components of the pixel at location $[x, y]$ in the image to be processed. Let us consider the group of neighboring pixels which belong to a 3x3 window centered on $[x, y]$. For calculating the gradients of the intensity functions in horizontal ΔI_x and vertical ΔI_y directions at position $[x, y]$, the intensity differences of the RGB components between the neighboring pixels are considered. For simplicity, we show the

expressions for only one (let's say the R) component (the same has to be evaluated in case of the other two, G and B components):

$$\begin{aligned} \Delta I_x^R &= \left| I^R(x+1,y) - I^R(x,y) \right| \\ \Delta I_y^R &= \left| I^R(x,y-1) - I^R(x,y) \right| \end{aligned}. \tag{2.28}$$

For the further processing the maximum of the estimated gradient values should be chosen, which solves as the input of the normalized linear mapping function P defined as follows:

$$P(v) = v / I_{\max}, \tag{2.29}$$

here $I_{\max}$ stands for the maximum intensity value. (For 8 bit RGB scales it equals 255.)
Let **R** be a rectangular image region of width rw and height rh, with upper left corner at position $[x_r, y_r]$. The R component level of the detail inside of region **R** is defined as

$$M_D^R(\mathbf{R}) = \sum_{i=0}^{rw} \sum_{j=0}^{rh} P\left(\max\left(\Delta I_x^R(x_r+i, y_r+j), \Delta I_y^R(x_r+i, y_r+j)\right)\right) \tag{2.30}$$

The sum of the three, R, G, and B component levels of detail gives the level of detail in region **R**

$$M_D(\mathbf{R}) = M_D^R(\mathbf{R}) + M_D^G(\mathbf{R}) + M_D^B(\mathbf{R}) \tag{2.31}$$

As higher is the calculated M_D value as detailed the analyzed region is.

2.5.3 HDR image synthesization

Consider that we have N color images of a static scene obtained at different exposures by using a stationary camera. The aim of the image synthesization method introduced in this subsection is to combine the images into a single one in such a way that all input information is included in the output image without producing noise.
For extracting all of the details involved in a set of images of the same scene made with different exposures, it is required to introduce a factor for characterizing the level of the detail in an image region and then we will be able to choose the most information dense parts of the input images to include in the output image. The measure proposed in the previous subsection (see Theorem 2.1) is a serious candidate for this purpose. In the followings a new gradient based multiple exposure time synthesization algorithm is detailed which uses the above detail level measurement method and solve the HDR image synthesization task.

Method 2.9 ([S32], [S], [S137]):

Step 1. Image segmentation: The first step of the processing is to divide the pictures into n rows and m columns, which yields nxm rectangular image regions. The regions in the images are of the same size with height rh and width rw (see Fig. 2.25). (In Fig. 2.25 you can see a 3x3 division.) Let $\mathbf{R}_{ijk}$ denote the region in the ith row and jth column of the image with index k. Let rx_{ij}, ry_{ij} denote the horizontal and vertical coordinates of the center of the region in the ith row and jth column. $I^R_k(x,y)$, $I^G_k(x,y)$, $I^B_k(x,y)$ stand for the RGB intensity values of the pixel at position (x,y) in the image with index k, where $k = 1, \ldots, N$ and N stands for the number of images to be processed, each of them taken with different exposures.

Step 2. Measuring the level of detail: For each image, the level of the detail has to be estimated inside every region $\mathbf{R}_{ijk}$ (Method 2.8 described in Subsection 2.5.2). This information helps us to select the most detailed region among the corresponding image segments (indexed by the same i and j values) which is included into the output scene. This step is repeated for each image segment, i.e. for $i = 1 \ldots n, j = 1 \ldots m$.

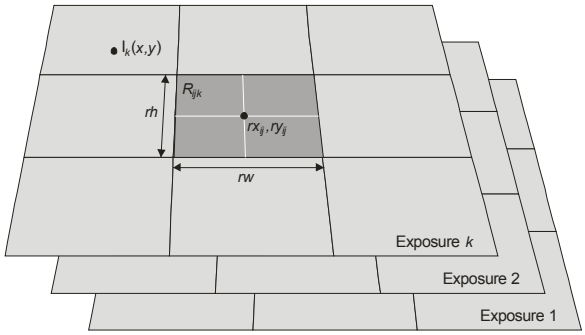


Fig. 2.25 Image regions after the segmentation

Let **D** denote the matrix of regions with the highest level of detail. Let d_{ij} be the element in the ith row and jth column of **D**, which stands for the index of the image, which has the most detailed region in the ith row and jth column, i.e.

$$M_D\left(R_{ijl}\right) > M_D\left(R_{ijk}\right) \mid k = 1...,l-1,l+1,...,N; l = d_{ij}\,. \tag{2.32}$$

Step 3. Merging the most informative segments: The next step is to merge the most detailed ($\mathbf{R}_{ijl}$) R, G, and B regions together, where $l=d_{ij}$ $i=1..n$, and $j=1..m$. Merging the selected regions together results in three images (a red, a green, and a blue) which contain every detail involved in the N input images.

Step 4. Smoothing: Unfortunately, the resulted images usually contain sharp transitions along the borders of the regions. These sharp transitions should be eliminated. A Gaussian blending function having the form of

$$G_{ij}(x,y) = \frac{e^{-\left(\frac{(x-rx_{ij})^2}{2\sigma_x^2}+\frac{(y-ry_{ij})^2}{2\sigma_y^2}\right)}}{\sum_{p=1}^{m}\sum_{q=1}^{n} e^{-\left(\frac{(x-rx_{pq})^2}{2\sigma_x^2}+\frac{(y-ry_{pq})^2}{2\sigma_y^2}\right)}} \tag{2.33}$$

can be applied advantageously, for this purpose. Here i and j stand for the row and column indices of the region over which the Gaussian hump $G_{ij}(x,y)$ is centered (see Fig. 2.26), m denotes the total number of columns and n the total number of rows in the input images. σ_x and σ_y stand for the standard deviation of the 2D Gaussian function. The values rx_{pq} and ry_{pq} represent the coordinates of the center of the region in the pth column and qth row, $1 \le r \le n$, $1 \le s \le m$.

Let U be a function defined as

$$U(x,y) = \begin{cases} 1 & (x,y) \in \mathbf{R}_{rs} \quad \mid \quad \left|rx_{rs} - rx_{ij}\right| \wedge \left|ry_{rs} - ry_{ij}\right| \le \varepsilon \\ 0 & \text{else} \end{cases}. \tag{2.34}$$

Function $U(x,y)$ is used for cutting the Gaussian function, i.e. for eliminating the influence of those segments, who's center points fall outside a predefined ε environment of the actually processed pixel.

Combining the Gaussian smoothing function $G_{ij}(x,y)$ and the cutting function $U(x,y)$, the output image can be evaluated according to

$$
\begin{aligned}
I_{out}(x,y) &= \langle I^R_{out}(x,y), I^G_{out}(x,y), I^B_{out}(x,y) \rangle, \\
I^R_{out}(x,y) &= \sum_{i=1}^{n}\sum_{j=1}^{m} G_{ij}(x,y)U(x,y)I^R_{d_{ij}}(x,y), \\
I^G_{out}(x,y) &= \sum_{i=1}^{n}\sum_{j=1}^{m} G_{ij}(x,y)U(x,y)I^G_{d_{ij}}(x,y), \\
I^B_{out}(x,y) &= \sum_{i=1}^{n}\sum_{j=1}^{m} G_{ij}(x,y)U(x,y)I^B_{d_{ij}}(x,y).
\end{aligned}
\tag{2.35}
$$

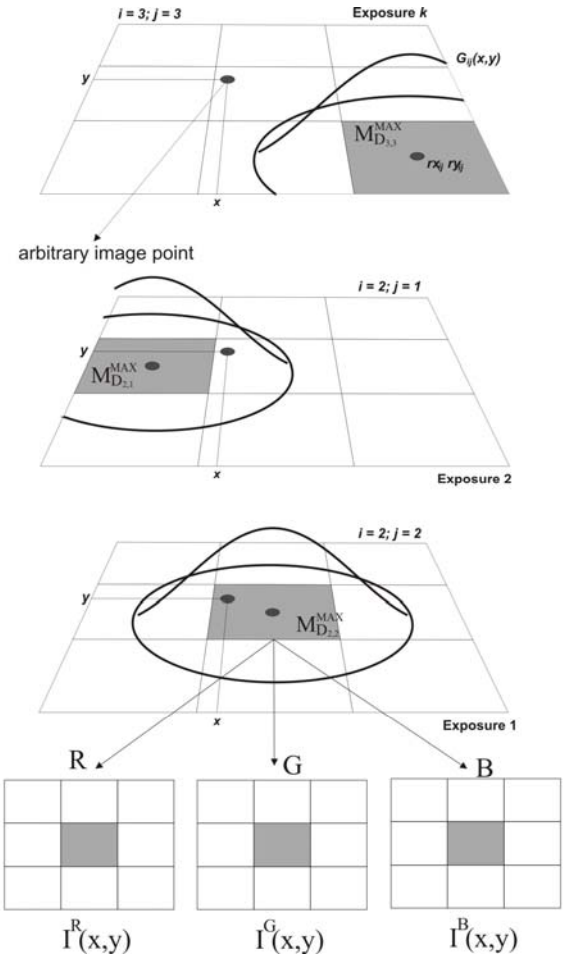


Fig. 2.26 Illustration of the Gaussian humps over the maximum level of detail regions used in the blending of the three (Red, Green, and Blue) output component images

The output color and intensity can be influenced by changing the size of the regions and the standard deviations of the Gaussian functions. E.g., as smaller is the standard deviation of the Gaussian function as higher influence the regions with low detail level onto the result have.
In Fig. 2.26 in case of exposure 1 the region in the 2nd row and 2nd column has the maximum level of detail compared to the other images, in case of exposure 2 the region in the 2nd row and 1st column while in case of exposure *k* the region in the 3rd row and 3rd column is the most detailed. The Gaussian functions are centered at (rx_{ij}, ry_{ij}) of the maximum level regions.
For increasing the speed of the processing, during the merging we can formulate groups of the adjacent regions originally belonging to the same exposure time image. Then for blending, a single Gaussian smoothing function can be applied over each of the groups, with center point falling into the center of gravity of the group. The cutting function *U* can also be defined in such

a way that the values of the blending function exceed zero over the whole image domain.

2.5.4 Illustrative example

The effectiveness of the proposed algorithm is illustrated by an example. The photos of the example are taken by Lou Haskell [28]. As input, three different exposure time color images are used. The width and height of the regions are chosen to 10 x 10 pixels. Deviations σ_x and σ_y equal to 60. For increasing the speed of the processing, during the merging we formulated groups of the adjacent regions originally belonging to the same exposure time image. A single Gaussian smoothing function is applied over each of the groups for the blending, with center point falling into the center of gravity of the group. The cutting function U is defined in such a way that the values of the blending function exceed zero over the whole image domain.

Fig. 2.27 Images of a scene taken by different exposure times, photo: Lou Haskell [28]

Fig. 2.28 Picture processed by easyHDR [28]

Fig. 2.29 Picture processed by the gradient based multiple exposure time synthesization algorithm (Method 2.9)

In Fig. 2.27 three photos of the same scene can be seen taken by different exposure times. For comparison, Fig. 2.28 presents a picture processed with easyHDR [28] while Fig. 2.29 shows the result got by the proposed new method.

Part III Automatic 3D reconstruction and its application in vehicle system dynamics

3D model reconstruction plays a very important role in computer vision as well as in different other engineering fields. The determination of the 3D model from multiple images is of key importance. One of the primary difficulties in autonomous 3D reconstruction is the (automatic) selection of "significant" points which carry information about the shape of the 3D bodies i.e. which are characteristic from the model point of view. Another problem to be solved is the point correspondence matching in different images. An automatic solution for finding the point correspondences would open new possibilities for the automation of many methods, currently done under manual assistance. The automation of 3D reconstruction may grease the skids to the development of new intelligent systems. The results of such a procedure could advantageously be applied among others in car-crash analysis, robot guiding, object recognition, supervision of 3D scenes, etc. In this chapter, besides describing an automatic 3D reconstruction method, we focus on its possible application in the field of vehicle system dynamics where we use it for car-crash analysis.

3.1 Introduction

Feature matching is a key element in many computer vision applications, for example in stereo vision, motion tracking, and identification. The most significant problem in stereo vision is how to find the corresponding points in two, ('left' and 'right') images, referred to as the correspondence problem. In the field of computer vision several applications require to reconstruct 3D objects from images taken form different camera positions. In recent time the interest in 3D models has dramatically increased [1], [2]. The emphasis for most computer vision algorithms is on automatic reconstruction of the scene with little or no user interaction [3].

Stereo techniques can be distinguished by several attributes, e.g., whether they use area-based or feature-based techniques (see explanation below), if they are applied to static or dynamic scenes, if they use passive or active techniques, or if they produce sparse or dense depth maps.

The extremely long computational time needed to match stereo images is still the main obstacle on the way to the practical application of stereo vision techniques. In applications such as robotics, where the environment being modeled is continuously changing, these operations must also be fast to allow a continuous update of the matching set, from which 3D information is extracted [4]. The correspondence search in stereo images is commonly reduced to significant features as computing time is still an important criterion in stereo vision.

There exist several stereo vision techniques, from which the most popular are the area-based and the feature-based method. The first kind of the mentioned techniques finds corresponding points based on the correlation between the corresponding areas in the left and right images [5]. First, a point of interest is chosen in one of the images. A correlation measure is then applied to search for a corresponding point with a matching neighborhood in the other image. Area-based techniques have the disadvantage of being sensitive to photometric variations during the image acquisition process and are sensitive to distortions, which are caused first of all by the changing viewing position.

Feature-based stereo techniques, on the other hand, match features in the left image to those in the right image. Features are selected as the most prominent parts in the image, such as, for example, edge points or edge segments, corner points etc. Feature-based techniques have the advantage of being less sensitive to photometric variations and of being faster than the area-based stereo method, because there are fewer candidates for matching corresponding points [6]. In this chapter a new approach is presented which combines the two techniques and, thus obtains better results. Based on the method, the automatic 3D reconstruction from images becomes also reality.
The alternative approach proposed in this chapter avoids most of the problems mentioned above. The object which has to be modeled is recorded from different viewpoints by a camera. The relative position and orientation of the camera and its calibration parameters are automatically retrieved from image data. For the reconstruction we use characteristic features, like edges and corner points of the objects. The complexity of the technique is kept low on one hand by filtering out the points and edges carrying non-primary information (i.e. the so-called texture edges and points) while on the other hand by applying recent methods of digital image processing (see e.g. [7]-[10]) combined with intelligent and soft (e.g. fuzzy) techniques.
The introduced autonomous 3D reconstruction and its algorithms can be applied advantageously at many fields of engineering. Here, we will show a possible application in vehicle system dynamics: the usage in car-crash analysis.
The chapter is organized as follows: In Section 3.2 the new automatic point correspondence matching method is detailed. Section 3.3 deals with the automation of the camera calibration while the description of the new automatic 3D reconstruction technique can be found in Section 3.4. Section 3.5 presents CASY, the car-crash analysis system: an intelligent application of the introduced algorithms.

3.2 Matching the corresponding feature points in stereo image pairs

Feature matching is commonly referred to as the correspondence problem. The problem is how to automatically match corresponding features from two images, while at the same time not assigning matches incorrectly. In the followings, an automatic point corresponding algorithm is presented.
The common approach for corners is to take a small region of pixels around the detected corner (referred to as a correlation window) and compare this with a similar region around each of the candidate corners in the other image. Each comparison yields a score, a measure of similarity. The match is assigned to the corner with the highest matching score. The most popular measure of similarity is the cross-correlation. Most matching algorithms include constraints to complement the similarity measure. These may take the form of constraints on which corners are selected as candidate matches: a maximum disparity, or corners which agree with some known relationship between the images (such as the epipolar geometry). Constraints such as uniqueness or continuity may also be applied after candidate matches have been found.

Theorem 3.1. The complexity of point correspondence matching can be reduced based on the combination of fuzzy techniques and epipolar constraints. It is sufficient to analyze as candidate corresponding points in the other image only the points carrying the same type of primary information (like corner and edge points) lying on the epipolar line in fuzzy sense ([S11], [S25]).

Proof:

Epipolar geometry [11] defines the connection between two images of a scene taken from different camera positions. Consider a point **M** in the 3D space. Consider the case of two perspective images of the 3D scene illustrated by Fig. 3.1. The 3D point **M** projects to point $\mathbf{m}_1$

in the left image and $\mathbf{m_2}$ in the right one. Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be the centers of projection of the left and right cameras respectively.

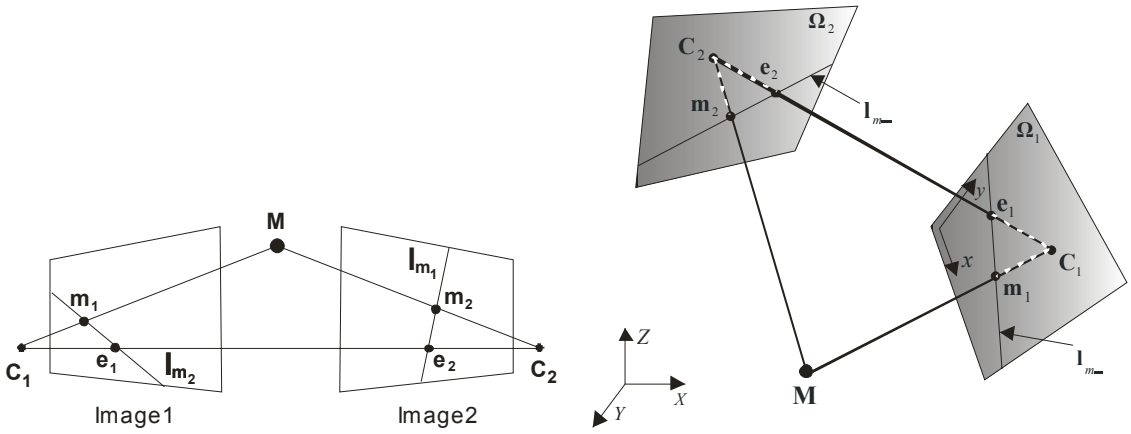


Fig. 3.1 Illustration of epipolar geometry

Points $\mathbf{e_1}$ and $\mathbf{e_2}$ are the so-called *epipoles*, and they are the intersections of the line $[\mathbf{C_1C_2}]$ with the two image planes. The plane formed by the three points $[\mathbf{C_1MC_2}]$ is called *epipolar plane*. The lines $[\mathbf{m}_1\mathbf{e}_1]\equiv\mathbf{l_{m2}}$ and $[\mathbf{m_2e_2}]\equiv\mathbf{l_{m1}}$ are called *epipolar lines* of $\mathbf{m}_2$ and $\mathbf{m}_1$, respectively. Point $\mathbf{m_2}$ is constrained to lie on the epipolar line $\mathbf{l_{m1}}$ of point $\mathbf{m_1}$. This is called *epipolar constraint*. Epipolar line $\mathbf{l_{m1}}$ is the intersection of the epipolar plane mentioned above with the second image plane Image2. This means that image point $\mathbf{m_1}$ can correspond to any 3D point on the line $[\mathbf{C_1M}]$ and that the projection of $[\mathbf{C_1M}]$ in the second image Image2 is the line $\mathbf{l_{m1}}$. All epipolar lines of the points in the first image pass through the epipole $\mathbf{e}_2$ and form thus a pencil of planes containing the baseline $[\mathbf{C_1C_2}]$. The above definitions are symmetric, i.e. the point of $\mathbf{m_1}$ must lie on the epipolar line $\mathbf{l_{m2}}$ of point $\mathbf{m_2}$.

Let $\mathbf{m_1} = (x, y, z)^t$ be the homogeneous coordinates of a point in the first image and $\mathbf{e_1} = (u, v, w)$ be the coordinates of the epipole of the second camera in the first image. The epipolar line through $\mathbf{m_1}$ and $\mathbf{e_1}$ is represented by the vector

$$\underline{\mathbf{l}}_{\mathbf{m2}} = (a,b,c)^t = \underline{\mathbf{m}}_{\mathbf{1}} \times \underline{\mathbf{e}}_{\mathbf{1}} \tag{3.1}$$

The mapping $\mathbf{m_1}\rightarrow\mathbf{l_{m2}}$ is linear and can be represented by a 3x3 rank 2 matrix $\underline{\underline{C}}$:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} yw - zv \\ zu - xw \\ xv - yu \end{pmatrix} = \begin{pmatrix} 0 & w & -z \\ -w & 0 & u \\ z & -u & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.2}$$

The mapping of epipolar line $\mathbf{l_{m2}}$ from Image1 to the corresponding epipolar line $\mathbf{l_{m1}}$ in Image2 is a collineation defined on the 1D pencil of lines through $\mathbf{e_1}$ in Image1. Let $\underline{\underline{A}}$ be one such collineation: $\mathbf{l_{m1}} = \underline{\underline{A}}\,\mathbf{l_{m2}}$.

Since $\underline{\underline{A}}$ has eight degrees of freedom and we only have five constraints, it is not fully determined. Nevertheless, the *fundamental matrix* $\underline{\underline{F}} = \underline{\underline{AC}}$ is fully determined [12][13]. We get

$$\mathbf{l_{m1}} = \underline{\underline{AC}}\,\underline{\mathbf{m}}_{\mathbf{1}} = \underline{\underline{F}}\,\underline{\mathbf{m}}_{\mathbf{1}} \tag{3.3}$$

It is a fact that all epipolar lines in the second image pass through $\mathbf{e_2}$ for all transferred $\mathbf{l_{m1}}$

$$\mathbf{e}_2^T.\mathbf{l_{m1}} = 0\,. \tag{3.4}$$

$\underline{\underline{F}}$ defines a bilinear constraint between the coordinates of the corresponding image points. If $\mathbf{m_2}$ is the point in the second image corresponding to $\mathbf{m_1}$, it must lie on the epipolar line $\mathbf{l_{m1}}$.

$$\underline{\mathbf{l}}_{\mathbf{m1}} = \underline{\underline{F}}\underline{\mathbf{m}}_{\mathbf{1}} \tag{3.5}$$

and hence

$$\underline{\mathbf{m}}_{\mathbf{2}}^{T}.\underline{\mathbf{l}}_{\mathbf{m1}} = 0\,. \tag{3.6}$$

The epipolar constraint can therefore be written as

$$\underline{\mathbf{m}}_{\mathbf{2}}^{T}\underline{\underline{F}}\underline{\mathbf{m}}_{\mathbf{1}} = 0 \tag{3.7}$$

For 3D reconstruction the edge and corner points are the most characteristic features because they carry the primary information about the boundaries, i.e. the shape of the objects. Edges can be decomposed to sections ending in corners, where the sections are either piecewise linear pieces which can be represented by their endpoints or are curved. In the first case the section can unambiguously be determined by its endpoints, i.e. the determination of the 3D coordinates of the corner points are sufficient for the 3D reconstruction which need the point correspondence matching of the corners. In the latter case, the common section of the curve and an epipolar line defines a finite set of points. If the angle between two camera positions is relatively low then the points keep their main character, i.e. to corner points in Image1 also corner points correspond in Image2 and edge points correspond to edge points.
This means that to determine the corresponding points to the characteristic corner or edge points of an object in Image1, we have to analyze only points which are corner or edge points lying in fuzzy sense on the corresponding epipolar line (i.e. near the epipolar line) (see Fig. 3.2). Thus, the number of candidate points is reduced significantly. This can be especially effective if before starting with point correspondence matching, we apply the surface smoothing procedure described in Section 2.3, which may cause a small "climbing" of the points.

3.2.1 Linear solution for the fundamental matrix

Each point match gives rise to one linear equation in the unknown entries of matrix $\underline{\underline{F}}$. The coefficients of this equation can easily be written in terms of the known coordinates of $\mathbf{m_1}$ and $\mathbf{m_2}$. Specifically, the equation corresponding to a pair of points $\mathbf{m_1}$ and $\mathbf{m_2}$ will be

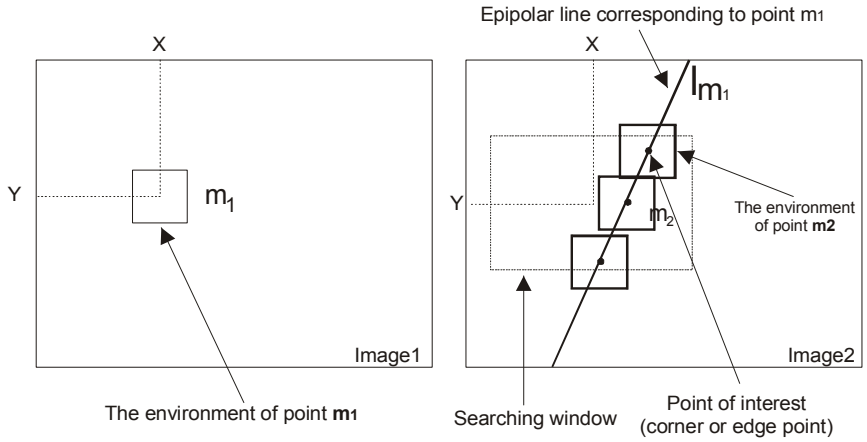


Fig. 3.2 Illustration of the point matching technique: in the left image a chosen corner is illustrated, while in the right image the candidate corner points can be seen

$$xx'f_{11} + xy'f_{12} + xf_{13} + yx'f_{21} + yy'f_{22} + yf_{23} + x'f_{31} + y'f_{32} + f_{33} = 0 \tag{3.8}$$

where the coordinates of $\mathbf{m_1}$ and $\mathbf{m_2}$ are $(x, y, 1)^t$ and $(x',y',1)^t$, respectively. Combining the equations obtained for each match gives a linear system that can be written as $\underline{\underline{A}}\underline{w} = \underline{0}$, where $\underline{w}$ is a vector containing the 9 coefficients of $\underline{\underline{F}}$ (f_{ij}) and each row of $\underline{\underline{A}}$ is built of the coordinates $\mathbf{m_1}$ and $\mathbf{m_2}$ of a single match. Since $\underline{\underline{F}}$ is defined only up to an overall scale factor, we can restrict the solution for $\underline{w}$ to have norm 1. We usually have more than the minimum number (8) of points, but these are perturbed by noise so we will look for a least squares solution:

$$\min_{\|\underline{w}\|=1} \left\| \underline{\underline{A}}\underline{w} \right\|^2 \tag{3.9}$$

As $\left\| \underline{\underline{A}}\underline{w} \right\|^2 = \underline{w}^T \underline{\underline{A}}^T \underline{\underline{A}}\underline{w}$, we have to find the eigenvector associated with the smallest eigenvalue of the 9x9 symmetric, positive semidefinite normal matrix $\underline{\underline{A}}^T\underline{\underline{A}}$. However, this formulation does not enforce the rank constraint, so a second step must be added to the computation to project the solution $\underline{\underline{F}}$ onto the rank 2 subspace. This can be done by taking the Singular Value Decomposition (SVD) of matrix $\underline{\underline{F}}$ and setting the smallest singular value to zero. Basically, SVD decomposes any real valued matrix $\underline{\underline{F}}$ in the form of

$$\underline{\underline{F}} = \underline{\underline{Q}}\,\underline{\underline{D}}\,\underline{\underline{R}} \tag{3.10}$$

where $\underline{\underline{D}}$ is diagonal and $\underline{\underline{Q}}$ and $\underline{\underline{R}}$ are orthogonal matrices. Setting the smallest diagonal element of $\underline{\underline{D}}$ to 0 and reconstituting gives the desired result.

3.2.2 New similarity measure for image regions

The images in which we have to find the corresponding feature points are taken from different camera positions. If the angle of the camera positions is relatively small, we have greater chance to match the mentioned feature points, because of the small deformation of image pixels between two views. In this case the corresponding points can be found with high reliability in each image. Feature point mentioned in this section can be either corners or edge points. Matches are found by evaluating the similarity between image regions and selecting the match of the pair of regions with the highest similarity (see Fig.3.2).
There are many similarity measure definitions known in the literature (see [14]). In this chapter, we introduce a new measure of similarity which is based on the combination of cross-correlation and a fuzzy measure:

Definition 3.1. Fuzzy cross-correlation similarity measure for image points ([S25], [S113], [S116])

$$M_s = \frac{\sum F_m(x,y) I_L(x,y) I_R(x,y)}{\sum F_m(x,y) I_L(x,y)^2 \sum F_m(x,y) I_R(x,y)^2}, \tag{3.5}$$

where I_L and I_R are the intensity functions of the input images (left and right image) and F_m stands for the fuzzy measure corresponding to the pixel with coordinates x,y. F_m can be calculated, as follows:

$$F_m(x,y) = MIN\{\mu_A(x), \mu_B(y)\}, \tag{3.6}$$

μ_A and μ_B are the membership functions in universes X and Y representing the closeness of the points in the environment to the analyzed corner point-candidate (see Fig. 3.3).

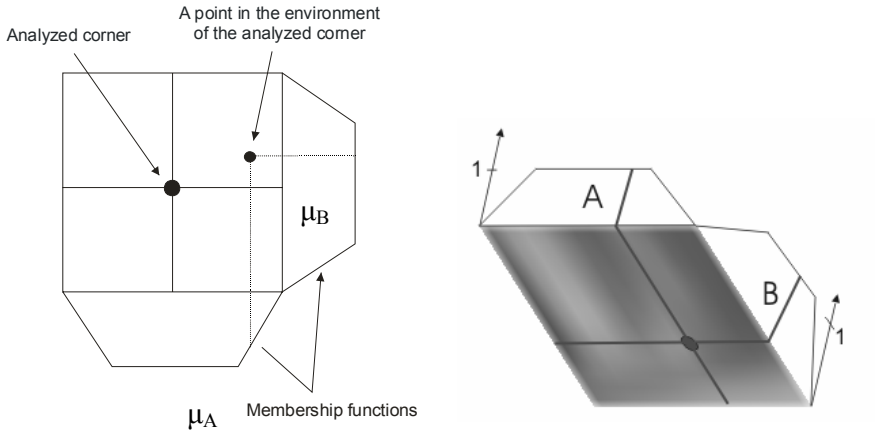


Fig.3.3 Fuzzy membership functions μ_A and μ_B of closeness used in (3.5) and (3.6)

3.2.3 Automatic image point matching

Based on the results of Subsections 3.2.1 and 3.2.2, automatic image point matching can be achieved according to Method 3.1 without any human intervention.

Method 3.1 ([S25], [S11], [S113], [S116]):

Step 1. Image preprocessing (noise filtering)

Step 2. Determination of the most characteristic image points (fuzzy corner detection, fuzzy edge detection, primary edge extraction, see Part II)

Step 3. Determination of the corresponding epipolar line of the processed corner/edge point

Step 4. Search for the candidate points. We know that the corresponding point will lay (in fuzzy sense) on the epipolar line and is also a corner/edge point (see Figure 3.2).

Step 5. Determination of the most probably corresponding point by the fuzzy similarity measure given in Subsection 3.2.2. It is worth remarking that the new corner detection methods proposed in 2.2.5 assign a new attribute, the fuzzy strength of being a corner, to the corner points. If the angle between the two camera positions is small, then the points keep this attribute in the images. Thus, the reliability of the matching can be improved by comparing this feature as well.
Step 6. The procedure has to be repeated for all characteristic corner points.

3.2.4 Experimental results

There are several known 3D reconstruction softwares in the market, however all of them need manual assistance in the point correspondence matching step, i.e., the user has to give manually the corresponding characteristic points in the images. On the contrary, the new method presented in the previous sections is able to solve this task automatically, without any human intervention. In the followings, one simple example is presented for illustrating the usability of the technique, more result can be found in works [S11], [S25], [S113], [S116].
Fig.3.4 illustrates two images taken of a car with different camera positions. In each of them an epipolar line corresponding to the pointed image pixel can be followed. In the right hand side figure, epipolar line LA corresponds to image point A of the left hand side figure while epipolar line LB corresponds to image point B. The corresponding image point of A is image point B and inversely: the point which corresponds to point B is point A.
In Fig. 3.5 the matching procedure can be followed. Here, the position of the car is approximately similar in both images, which results in that the search can be limited to a certain area of the image (the bigger, yellow searching window around the point to be matched) and we do not have to analyze all corner points on the corresponding epipolar line. After finding the

candidate corner points falling into the searching window, the determined image regions around the points (the smaller, red window) are compared by evaluating the similarities according to Definition 3.1. The point with the most similar region is chosen as corresponding point.

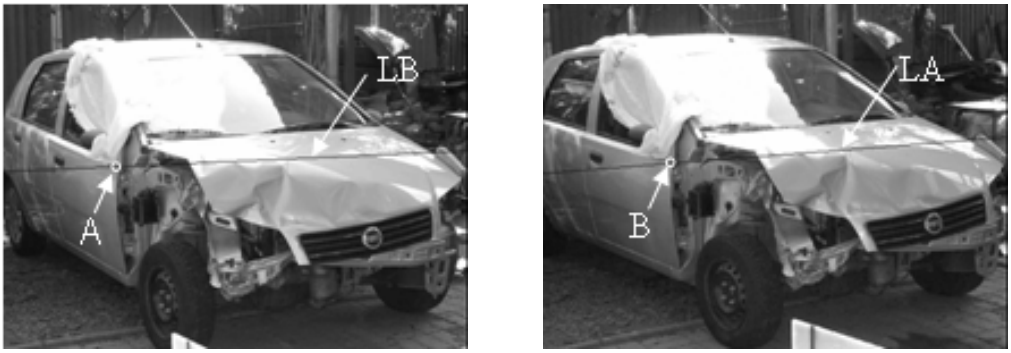


Fig.3.4. The epipolar lines (LA, LB) of two corresponding feature points (A,B)

Fig. 3.5 Illustration for the determination of the corresponding pixels (Method 3.1)
Here, since the position of the car is approximately similar in the images, the search can be limited to a certain area of the image (the bigger, yellow searching window around the point to be matched). The smaller, red window represents the image regions around the points where the similarities are evaluated.

According to our experience, the reliability of the matching is over 98% in case of characteristic points of crashed cars like in Figures 3.4 and 3.5. Problematic cases cover situations where the scene is composed of identical parts, which are located "parallel" to the epipolar line. Even in this case, by matching another point from the neighborhood (which epipolar line will not be "parallel" to the locations) and taking into account a possible distance interval of the points, the false matching can be neglected and the reliability can be increased to theoretically 100 %.
Fig. 3.6 illustrates several samples and a test sample which is compared to the others. Fig 3.7 shows the similarity measures calculated using the new fuzzy based technique (solid line) and the method without fuzzy reasoning (dashed line). Higher levels of similarity correspond to smaller calculated similarity measure values (see e.g. samples 1 and 5). The scene of sample 5 is very similar to that of the test sample but the illumination is quite different. Although, the significant differences appear in the farther environment of the image point to be matched which is taken into consideration through the fuzzy weighting by the new method. As a result, the fuzzy based method finds in sample 5 higher similarity than the other technique.

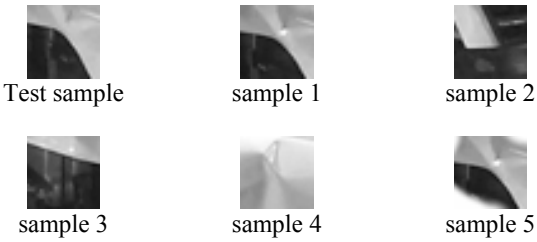


Fig. 3.6 Illustration for the application of the new similarity measure. The five samples are compared to the test sample using the fuzzy based similarity measure (Definion 3.1)

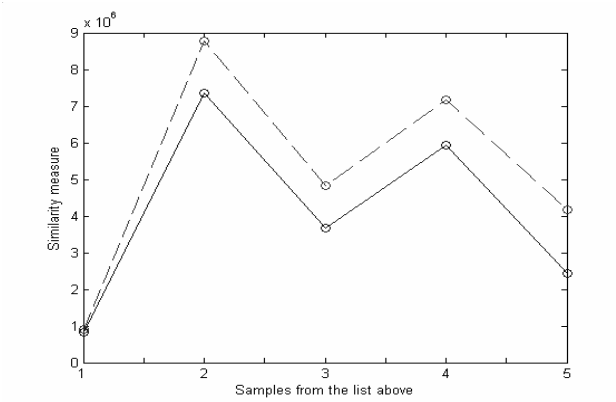


Fig.3.7 The corners of the graphs represent the samples which are compared to the test sample illustrated in Fig. 3.6. The solid line illustrates the results using the fuzzy based similarity measure, while the dashed line shows the results using the same similarity measure without fuzzy reasoning

3.3 Camera Calibration by Estimation of the Perspective Projection Matrix

In the followings, we propose a novel, autonomous method for camera calibration.

Method 3.2: Automatic calibration of uncalibrated cameras ([S25], [S11], [S113], [S116]):

Proof of the correct operation of the algorithm:

There exists a collineation, which maps the projective space to the camera's retinal plane: 3D to 2D. Then the coordinates of a 3D point $\mathbf{M} = [M_X, M_Y, M_Z]^T$ (determined in an Euclidean world coordinate system) and the retinal image coordinates $\mathbf{m} = [m_x, m_y]^T$ (see Fig. 3.8) are related by the following equation:

$$\begin{bmatrix} m_x W \\ m_y W \\ W \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & 1 \end{bmatrix} \begin{bmatrix} M_X \\ M_Y \\ M_Z \\ 1 \end{bmatrix} \tag{3.7}$$

where W is a scale factor, $\mathbf{m} = [m_x, m_y, 1]^T$ and $\mathbf{M} = [M_X, M_Y, M_Z, 1]^T$ are the homogeneous coordinates of points **m** and **M**, and $\underline{\underline{P}}$ is a 3 × 4 matrix representing the collineation 3D to 2D. One parameter of $\underline{\underline{P}}$ can be fixed ($l = 1$). $\underline{\underline{P}}$ is called the perspective projection matrix. Values *a,*

b, c, d, e, f, g, h, i, j ,k are the elements of the projection matrix $\underline{\underline{P}}$. The optical axis passes through the center of projection (camera) **C** and is orthogonal to the retinal plane. The focal length f_l of the camera is also shown, which is the distance between the center of projection and the retinal plane. If only the perspective projection matrix $\underline{\underline{P}}$ is available, it is possible to recover the coordinates of the optical center or camera [12]. It is clear that

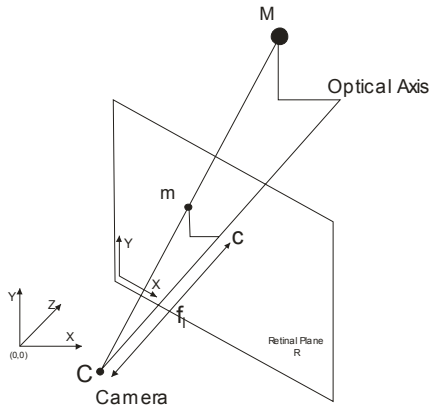


Fig. 3.8 Perspective projection – illustration of points M=[X,Y,Z] and its projection m=[x,y] in the retinal plane R.

$$W = iM_X + jM_Y + kM_Z + 1 \tag{3.8}$$

From (3.7) we can compute the coordinates of point **m** (m_x, m_y), as follows:

$$m_x = \frac{aM_X + bM_Y + cM_Z + d}{W} \tag{3.9}$$

$$m_y = \frac{eM_X + fM_Y + gM_Z + h}{W} \tag{3.10}$$

$$M_X a + M_Y b + M_Z c + d + 0e + 0f + 0g + 0h - M_X m_x i - M_Y m_x j - M_Z m_x k = m_x \tag{3.11}$$

$$0a + 0b + 0c + 0d + M_X e + M_Y f + M_z g + h - M_x m_y i - M_y m_y j - M_z m_y k = m_y \tag{3.12}$$

All together we have eleven unknowns, the elements of the projection matrix that means that we need six points to determine the projection matrix. After substitutions and equivalent transformations we get the following equations and matrices:

$$\underline{\underline{A}}\,\underline{q} = \underline{b} \tag{3.13}$$

where matrix $\underline{\underline{A}}$ and vectors $\underline{q}$ and $\underline{b}$ are described thereinafter (see (3.16), (3.18), and (3.19)).

$$\underline{\underline{A}}^T\,\underline{\underline{A}}\,\underline{q} = \underline{\underline{A}}^T\,\underline{b} \tag{3.14}$$

From (3.13) the projection matrix can be obtained:

$$\underline{q} = (\underline{\underline{A}}^T \underline{\underline{A}})^{-1} \underline{\underline{A}}^T \underline{b} \tag{3.15}$$

$$\underline{\underline{A}} = \begin{bmatrix} M_{1X} & M_{1Y} & M_{1Z} & 1 & 0 & 0 & 0 & 0 & -m_{1x}M_{1X} & -m_{1x}M_{1Y} & -m_{1x}M_{1Z} \\ 0 & 0 & 0 & 0 & M_{1X} & M_{1Y} & M_{1Z} & 1 & -m_{1y}M_{1X} & -m_{1y}M_{1Y} & -m_{1y}M_{1Z} \\ & & & & & & \vdots & & & & \\ M_{nX} & M_{nY} & M_{nZ} & 1 & 0 & 0 & 0 & 0 & -m_{nx}M_{nX} & -m_{nx}M_{nY} & -m_{nx}M_{nZ} \\ 0 & 0 & 0 & 0 & M_{nX} & M_{nY} & .M_{nZ} & 1 & -m_{ny}M_{nX} & -m_{ny}M_{nY} & -m_{ny}M_{nZ} \end{bmatrix} \tag{3.16}$$

The first two lines of matrix $\underline{\underline{A}}$ correspond to points $\mathbf{M_1}$ and $\mathbf{m}_1$, the second two lines correspond to points $\mathbf{M_2}$ and $\mathbf{m_2}$, etc. With the help of these points we can compute the elements of the projection matrix $\underline{\underline{P}}$

$$\underline{\underline{P}} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & 1 \end{bmatrix}. \tag{3.17}$$

Vector $\underline{q}$ is composed of the elements of projection matrix $\underline{\underline{P}}$

$$\underline{q} = \begin{bmatrix} a & b & c & d & e & f & g & h & i & j & k \end{bmatrix}^T. \tag{3.18}$$

The elements of vector $\underline{b}$ are the coordinates of points $\mathbf{m_i}$, where i=1...n

$$\underline{b} = \begin{bmatrix} m_{1x} & m_{1y} & m_{2x} & m_{2y} & . & . & . & m_{nx} & m_{ny} \end{bmatrix}^T. \tag{3.19}$$

From the following deduction follows that we need six known points to determine the projection matrix. It can be solved if we place a known 3D object with minimum six easily localizable points (not located in one plane) to the camera's field of vision and fix the origin of the above coordinate system to one of the base corners of the object. Then, the automatic calibration can be performed. This object can be e.g. a cube-shaped frame with known edge-length where the corners of the cube can easily be determined.

3.4 3D reconstruction of objects starting of 2D photos of the scene

3.4.1 3D reconstruction

After solving automatic point correspondence matching and camera calibration, automatic 3D reconstruction simplifies to the following algorithm:

Method 3.3 ([S10], [S30], [S39], [S127], [S130], [S132]):

Step1. Preprocessing (noise elimination edge detection, primary edge extraction, see Part II)

Step 2. Determination of the 3D coordinates of the extracted characteristic corner/edge points.

This step means the determination of the corner and edge correspondences, which is followed by the camera calibration (determination of the Perspective Projection Matrix). In the knowledge of

the 3D coordinates and the correspondences of the significant points the spatial model of the car body can easily be built.

Step 3. Building the 3 model.

3.4.2 Experimental results

In the followings a simple example is shown for the 3D reconstruction procedure. For more examples and details see [S10], [S30], [S127], [S130], [S132].
Fig. 3.9 shows an unknown object to be modeled in 3D. The cube-shaped frame (with edge-length=20 cm) for the automatic calibration is also included.

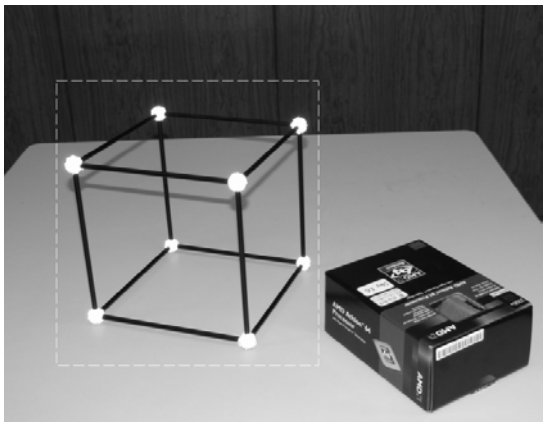


Fig. 3.9 An unknown object and the cube-shaped frame (with edge-length=20 cm) for the automatic calibration

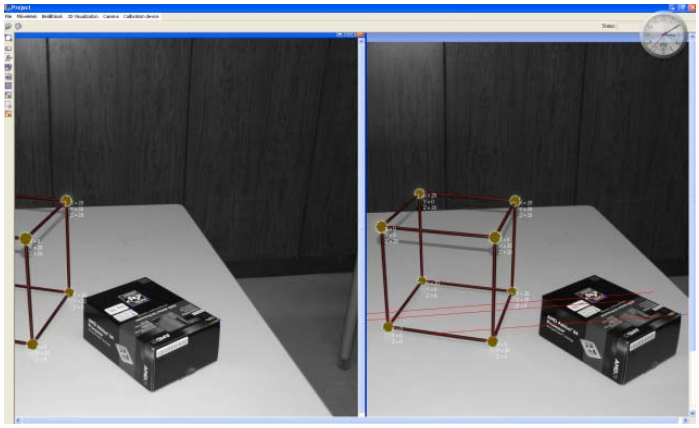


Fig. 3.10 Two images of the object taken from different camera positions

In Fig. 3.10 two pictures taken from different camera positions can be followed. Here, we are after the camera calibration step, as the coordinates of the calibration object have already been determined. Fig. 3.11 presents the reconstructed 3D model of the unknown object.

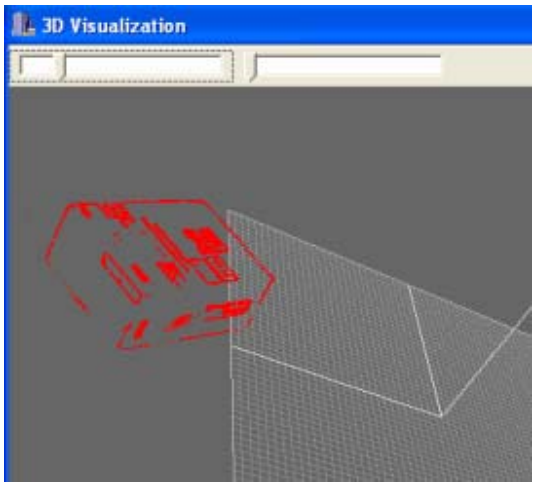


Fig. 3.11 The reconstructed 3D model of the object

3.5 The intelligent car-crash analysis system

3.5.1 Introduction

Crash and catastrophe analysis has been a rather seldom discussed field of traditional engineering in the past. In recent time, both the research and theoretical analyses have become the part of the everyday planning work (see e.g. [15]). The most interesting point in crash analysis is that even though the crash situations are random probability variables, the deterministic view plays an important role in them. The stochastic view, statistical analysis, and frequency testing all concern past accidents. Crash situations, which occur the most frequently (e.g. the characteristic features of the crash partner, the direction of the impact, the before-crash speed, etc.) are chosen from these statistics and are used as initial parameters of crash tests. These tests are quite expensive, thus only some hundred tests per factory are realized annually, which is not a sufficient amount for accident safety. For the construction of optimal car-body structures, more crash-tests were needed. Therefore, real-life tests are supplemented by computer-based simulations which increase the number of analyzed cases to 1-2 thousands. The computer-based simulations – like the tests – are limited to precisely defined deterministic cases. The statistics are used for the strategy planning of the analysis. The above mentioned example clearly shows that the stochastic view doesn't exclude the deterministic methods [16].
Crash analysis is also very helpful for experts of road vehicle accidents, since their work requires simulations and data, which are as close to the reality as possible. By developing the applied methods and algorithms we can make the simulations more precise and thus contribute towards the determination of the factors causing the accident.
The results of the analysis of crashed cars, among which the energy absorbed by the deformed car body is one of the most important, are of significance at other fields, as well. They carry information about the deformation process itself, as well, having a direct effect on the safety of the persons sitting in the car. Thus, through the analysis of traffic accidents and car crash tests we can obtain information concerning the vehicle which can be of help in modifying the structure/parameters to improve its future safety. There is an ever-increasing need for more correct techniques, which need less computational time and can more widely be used. Thus, new modeling and calculating methods are highly welcome in deformation analysis.
The techniques of deformation energy estimation used until now can be classified into two main groups: The first one applies the method of finite elements [17]. This procedure is accurate enough and is suitable for simulating the deformation process, but this kind of simulation

requires very detailed knowledge about the parameters of the car-body and its energy absorbing properties, which in most of the cases are not available. Furthermore, if we want to get enough accurate results, its complexity can be very high.
The other group covers the so called energy grid based methods, which start from known crash test data and from the shape of the deformation, or from the maximal car-body deformation [18][19]. The distribution of the energy, which can be absorbed by the cells is considered just in 2D and the shape of the deformation is described by a 2D curve. This curve is the border of the deformation visible from the top view of the car-body. Of course, there are a lot of cases, when the shape of the deformation can not be described by 2D curves. Furthermore, in many of the cases considering the energy distribution in 2D is not enough precise, because the absorbing properties of the car-body change along the vertical axis as well, which is not considered by these methods.
Starting from these reasons, in this section a new energy estimation method is presented, which avoids the mentioned limitations of the standard methods. First, the energy distribution is considered in 3D. Secondly, for describing the shape of the deformation, spline surfaces are used, with the help of which complex deformation surfaces can be modeled easily. Furthermore, for decreasing the computational cost and time, soft computing techniques are used, which also enable to achieve more accurate results. Last, the deformation surface is obtained from the digital photos of the car body by 3D reconstruction.
We will show how we can construct a system capable to automatically build the 3D model of a crashed car as well as to determine the energy absorbed by the car-body deformation and the speed of the crash.
Section 3.5 is organized as follows: In Subsection 3.5.2 the intelligent car crash analysis system is discussed briefly. Subsection 3.5.3 shows how to evaluate the shape of the deformation and based on it the deformation energy from digital pictures. Section 3.5.4 presents extraction of the 3D car body model from digital images. Section 3.5.5 discusses how to determine the direction of impact and the absorbed energy. Section 3.5.6 shows an example to illustrate the effectiveness of the presented methods

3.5.2 The concept of the car-crash analysis system ([S10], [S39], [S40])

The block structure of the proposed new car crash analysis system can be followed in Fig. 3.12. It contains four well defined sub-blocks. The first (image processing) is responsible for the pre-processing of the digital photos (noise elimination/filtering, edge detection, corner detection, primary edge extraction) and for the 3D modeling (including the point correspondences and the 3D model building). The second part of the system (comparison of models) calculates the volumetric change of the car body from the deformed and the original 3D models of the car. Parallel with it, an expert system (Expert system) determines the direction of the impact. Based on the direction of impact and volumetric change a hierarchical fuzzy-neural network system determines the absorbed energy and the energy equivalent speed of the car.
The methods applied in the first block are summarized in Part II and the previous sections of Part III. The volumetric change can be determined after fitting the deformed and undamaged car-body models. Here we utilize the fact that if the car hits a wall then the backside of it will remain undamaged and this part of the models can be fitted based on minimization of the mean squares error of the deviation of the 3D shapes. The ideas and main steps of the other blocks are detailed in the next sections.

3.5.3 Determination of the direction of impact, the absorbed energy and the equivalent energy equivalent speed ([S10], [S30], [S39], [S40])

The spatial model of the deformed car-body serves as input to the “Expert system” block, as well. This module applies an expert system and produces the direction of impact. For this we use the so called “energy-centers” of the undamaged and deformed car bodies and the direction is

estimated from the direction of movement of the energy-center of the car-body part attached in the deformation.

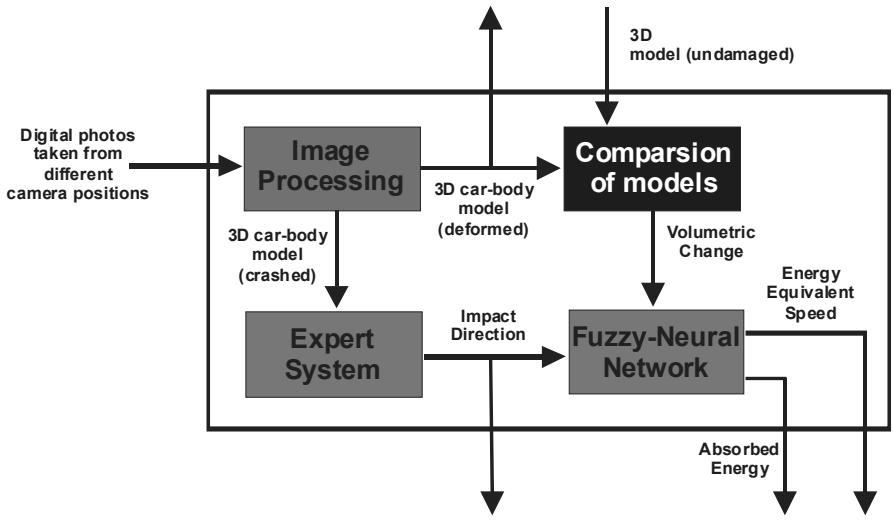


Fig. 3.12 Block-structure of the intelligent car crash analysis system

Definition 3.2 Energy-center of a body ([S30], [S127]):

3D bodies can be represented by elementary 3D cells of the body. These cells can be got by cutting the body to elementary pieces along all 3 dimensions. If we deform the body, during the deformation the different 3D cells of the body absorb a certain amount of energy. The energy-center can be determined by weighting the cells by the corresponding energy values.

The above technique usually gives acceptable estimation for the direction of impact. Although, problem may occur when the speed of the car is very low or when the angle of the direction of impact and the angle, between the main axis of the car (which goes through the energy-center and points "forward") and the central axis of the attached surface, are very different. In these latter cases a more sophisticated new method, surface fitting can be used for the determination of the direction of impact.

Method 3.4 (these results have been developed in the frame of the GVOP -3.3.1-05/1.2005-05-0160/3.0 project under the leadership of the author)

Surface fitting applied in the determination of direction of impact is based on the fitting of the 3D surfaces of the attached objects created by the deformation during the crash.

Step 1. The fitting is solved by minimizing the LMS error of the differences between the two surfaces. The fitting is done in 3D. As a result, the positions of the two vehicles or objects can be reconstructed with acceptable accuracy (as an illustration, see Fig. 3.13).
The collision can be decomposed into two parts. The major part of the deformation during collision belongs to the type of plastic collision, resulting in lasting deformation in the body, while a small part is elastic collision resulting in reversible deformation and thus error in the energy estimation. According to our experiences, the error caused by the elastic deformation can be tolerated, it usually is below 10%.

Step 2. The direction of impact is estimated as the angle between the longitudinal axes of the two objects/vehicles attached in the impact.

Here we would like to remark that Method 3.4 can be applied both if the car hits a wall and if two vehicles are crashed.

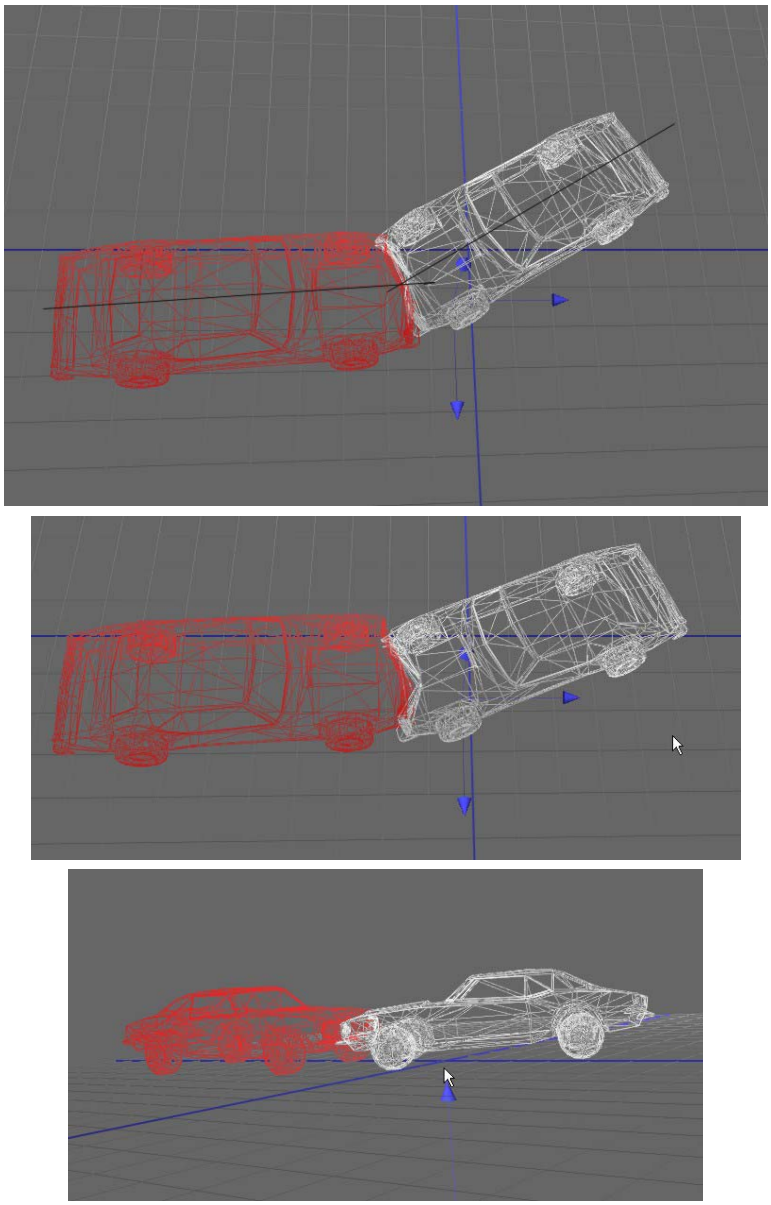


Fig. 3.13 Illustration for the surface fitting

The absorbed energy and the equivalent energy equivalent speed (EES) is evaluated from the volumetric difference and from the direction of impact by an intelligent hierarchical fuzzy-neural network system (lower right hand side block in Fig. 3.12). Several classes of cases are formulated according to the main types of the crash and the car and different sets of NNs are used in the different classes. A concrete set of NN is suitable for modeling the problem of a certain class and the elements of the set are smaller size NNs operating as local models over a subspace of the state space of the class.

Before choosing the correct set of neural networks we have to pre-determine the category of the analyzed vehicle and the main character of the crash: Cars are classified into car categories according to their weights. Concerning the character of the crash we have formulated 12 different cases which follow the system used in car factories during the crash tests (frontal full impact, frontal offset impact, side impact, corner impact, rear impact, …), see also Fig. 3.14.

For increasing the reliability of the modeling and for decreasing the teaching phase of the neural networks we have applied a fuzzy weighted locally approximated neural network system for the neural network based modeling of the absorbed energy.

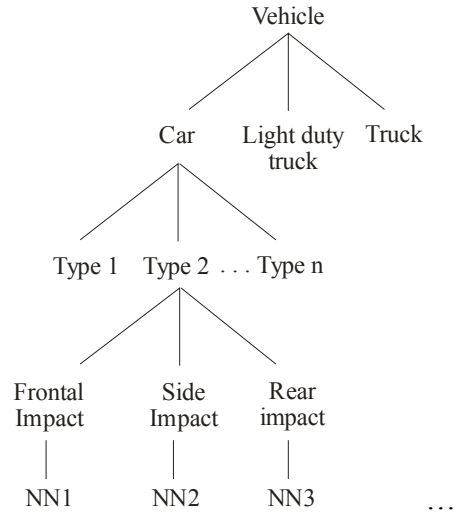


Fig. 3.14 Hierarchical structure of the pre-classification in the EES determination

Definition 3.3 Fuzzy weighted locally approximated neural network system ([S10], [S30], [S39], [S40], [S127]):

For the neural network based modeling of a system, the n-dimensional state space of the problem is decomposed into subspace domains which can "easily" be modeled by neural networks, i.e. does not contain many local minima which could "trap" the neural networks. Over each domain an elementary neural network is developed and trained separately. Since the transitions between the neighboring domains are unambiguous and to ensure the smooth transition between the domains, a fuzzy system is used for the determination of the actual (fired) local neural networks to be applied, with a rulebase as

$$R_{i1,..,in}: \quad \text{If } x_1 \text{ is } A_{1i1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{nin} \text{ then } use\ NN_{i1,\ldots,in} \tag{3.20}$$

where $R_{i1,..,in}$ stands for the $i1,\ldots,in$-th rule, x_j (j=1…n_j) denotes the j-th input variable, A_{jij} is the i_j-th fuzzy set in input universe x_j ,and $NN_{i1,\ldots,in}$ represents the local neural network model of domain $i_1,\ldots,i_n$.

Applying the above NN system, for each class of EES determination a different set of neural networks is developed. For the determination of the elementary neural networks (local models) the surface is divided into domains. The elementary neural networks are trained over these surface domains. The fuzzy system for the determination of the fired local neural networks consists of rules as

R_{ij}: If *the direction of impact* is *Di* and *the volumetric change* is *Vj* then *use NNij*

For the training of the neural networks simulation and crash test data is used. The training data include the volumetric change, the direction of the impact (input data) and the corresponding deformation energy (output data). The surface is usually symmetric (to the longitudinal axes of the vehicle) in which case it is enough to deal with its half part.
After the determination of the absorbed energy, the equivalent energy equivalent speed can be determined according to e.g. [20].

3.5.4 Experimental results

A software package and an experimental analysis system (CASY) has been developed for the intelligent analysis of crashed cars in the frame of the GVOP -3.3.1-05/1.2005-05-0160/3.0

project under the leadership of the author. At current stage, the system is able to automatically determine the 3D model, the absorbed energy, the direction of impact, and the energy equivalent speed based on digital photos made of the crashed car from different camera positions, if the car hits a (massive) wall. The methods and algorithms of Parts II and III are built in the system.
In the followings, an example is shown illustrating the operation of the system. The example is taken from a car crash test, with the following parameters:

Vehicle / Mass of the vehicle: Audi 100 / 1325 kg
Real direction of impact: 0 Degree
Real EES of the vehicle: 55 km/h
Volumetric change (evaluated): 0.62 m3
Absorbed deformation energy (evaluated): 171960 Joule

The absorbed energy surface (Fig. 3.15) belonging to this car-class is symmetric (to the longitudinal axes of the vehicle) so only half of the surface has to be dealt with. The used NNs modeling this surface shown in Fig. 3.15 has been taught based on the simulation data of a similar class, but Mercedes car. The approximation of domains D1 and D2 can be solved by simple feed-forward backpropagation NNs with one hidden layer and three hidden neurons. The NNs are used to determine the deformation's energy and EES. During the teaching period of the system, the determined EES values were compared to known test results and the parameters of the expert system were modified to minimize the LMS error. The accuracy of the system is influenced by the accuracy of the crash test data.

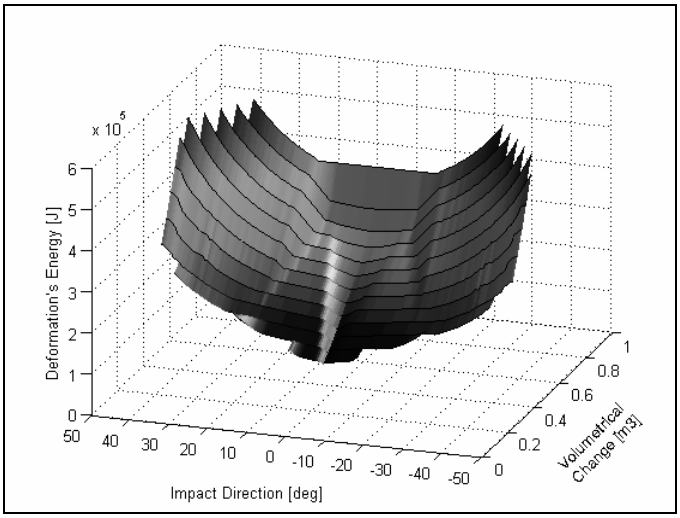


Fig. 3.15 Relation among the direction of impact, volumetric change, and the deformation energy based on simulation data (Mercedes 290)

The half of the surface is decomposed into two parts along the impact direction (Fig. 3.16). The fuzzy sets for the weighting of the NNs are shown in Fig. 3.17. (Although, in this case this simple decomposition results in good performance, here we would like to remark that in general the mapping is more complex and it can be advantageous to define more domains along both inputs to keep the complexity of the used NNs low.)
In Figs. 3.18 and 3.19 steps of the processing of two of the thirty used viewpoints can be followed. The resulted 3D models of the damaged and undamaged car-bodies are shown by Figs. 3.20 and 3.21, respectively, while in Fig. 22 the corresponding energy cells can be followed.
The results of the analysis are summarized in Table1. For comparison, the results got by the 2D method are also attached. (The 2D method uses a 2D deformation curve for identifying the deformation and a 2D energy grid. The error of this method is usually much higher than is in this simple example, because in most of the cases the deformation shape can not accurately be

identified in 2D. Furthermore, the 2D method is not able to model the differences in the energy absorption properties along the vertical axis. Although, we included this example to demonstrate that new method out performs the known 2D method even in these cases when the deformation is “simple” and “visible” from above. The proposed new method is able to handle such cases as well and uses 3D distribution of the energy, which also increases the accuracy of the results.) For more details and examples see [S10], [S11], [S25], [S30].

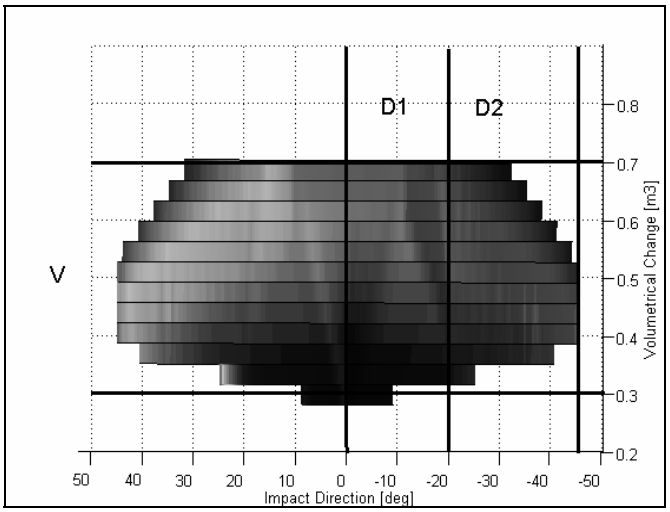


Fig. 3.16 Segmentation of the surface in Fig. 3.14

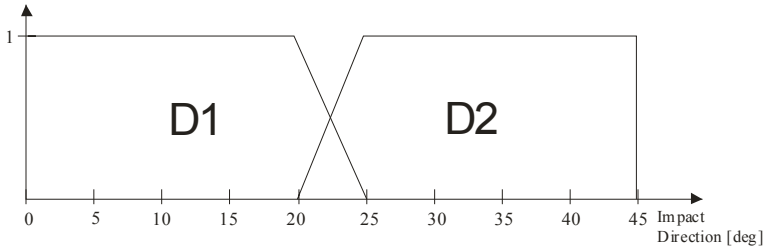


Fig. 3.17 Membership functions defined on the universe of impact direction

Fig. 3.18 Noise filtered photo of a crashed Audi (viewpoint 1); edge map of the car; corners detected in the image

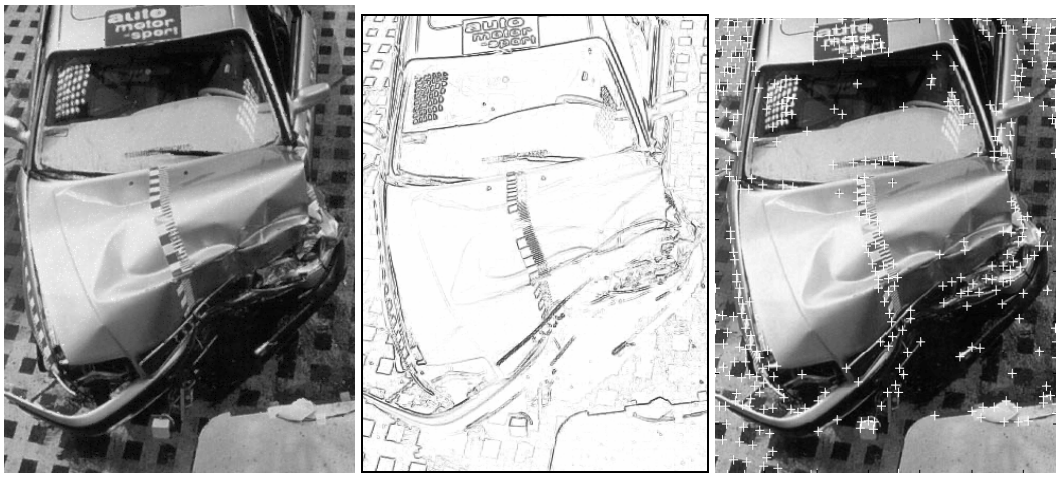


Fig. 3.19 Noise filtered photo of a crashed Audi (viewpoint 2); edge map of the car; corners detected in the image

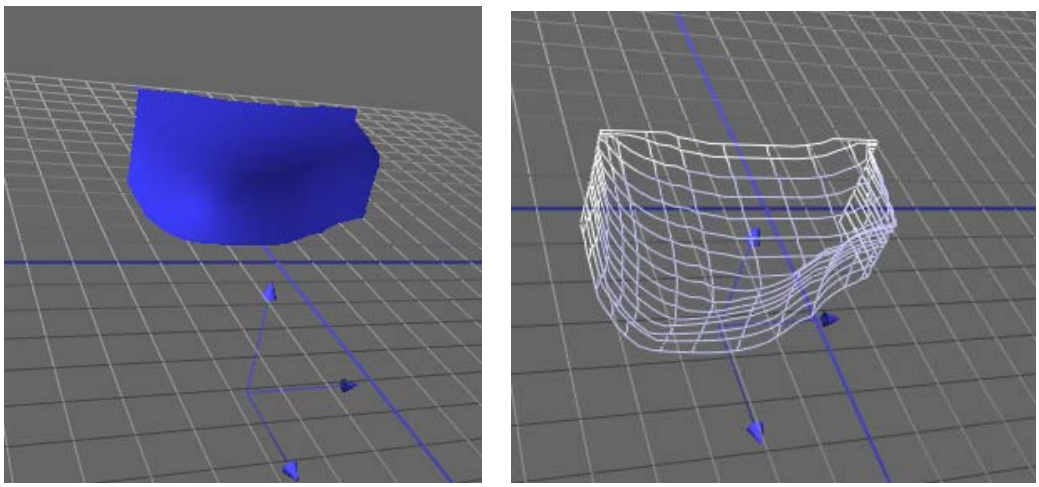


Fig. 3.20 3D model of the deformed part of the car body

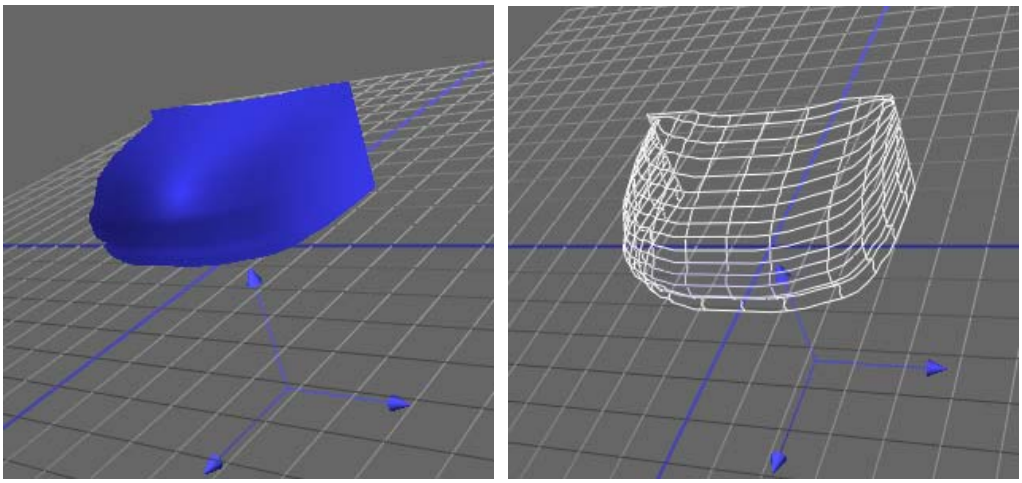


Fig. 3.21 3D model of the undeformed car-body

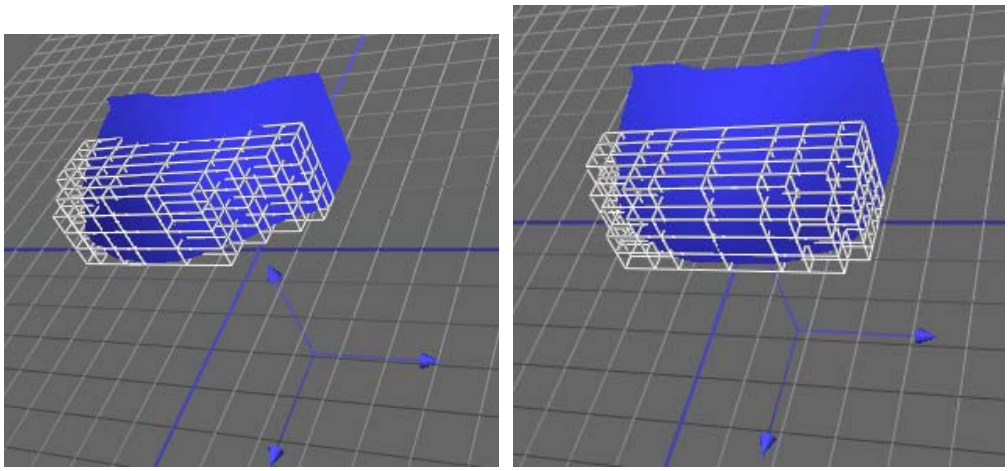


Fig. 3.22 3D energy cells of the deformed and undeformed car-bodies

Table 3.1 Estimation of the direction of impact and energy equivalent speed of the crashed Audi

	Direction of impact [deg]	**EES of the vehicle [km/h]**
Real Data	0	55
Proposed method	2	58
2D method	2	59,2

Part IV Generalization of Anytime Systems, Anytime Extension of Fuzzy and Neural Network Based Models

Nowadays practical solutions of engineering problems involve model-integrated computing. Model based approaches offer a very challenging way to integrate a priori knowledge into the procedure. Due to their flexibility, robustness, and easy interpretability, the application of soft computing, in particular fuzzy and neural network based models, may have an exceptional role at many fields, especially in cases where the problem to be solved is highly nonlinear or when only partial, uncertain and/or inaccurate data is available. Nevertheless, ever so advantageous their usage can be, it is still limited by their exponentially increasing computational complexity. Combining soft computing and anytime techniques is a possible way to overcome this difficulty, because the anytime mode of operation is able to adaptively cope with the available, usually imperfect or even missing information, the dynamically changing, possibly insufficient amount of resources and reaction time.
In this part the applicability of (Higher Order) Singular Value Decomposition based anytime Soft Computational models is analyzed in dynamically changing, complex, time-critical systems. Practical questions of implementing and operating anytime systems are also analyzed.

4.1 Introduction

Nowadays, solving engineering problems model-integrated computing has become very popular. This integration means that the available knowledge is represented in a proper form and acts as an active component of the procedure (computer program) to be executed during the operation of signal processing, diagnostics, measuring, control, etc. devices. Thus, model based approaches are very useful in integrating a priori knowledge into the procedures.
In case of linear problems, well established methods are available and they have been successfully combined with adaptive techniques to provide optimum performance. Nonlinear techniques, however, are far from this maturity. There is a wide variety of possible models to be applied based on both classical methods [1] and recent advances in handling information [2] but systematic methods to solve a larger family of nonlinear engineering problems became only recently available: the efforts on the fields of fuzzy and neural network (NN) modeling seem to result in a real breakthrough in this respect (see e.g. [3]-[11]). These techniques can be applied even in cases when no analytical knowledge is available about the system, the information is uncertain or inaccurate, or when the available mathematical form is too complex to be used.
One of the greatest advantages of fuzzy systems is that they are able to handle not only imprecise data, but also inexactly formulated concepts which cannot be easily expressed by classical tools only. Thus, fuzzy systems can be well used in cases, when only expert knowledge and/or (imprecise) sample data is available, and furthermore, the interpretation may depend on the context [12], [S54].
A major limitation of fuzzy (and NN) models is their exponentially increasing complexity. One can easily get into trouble when considering complex problems with a large number of parameters, especially, when only partial and uncertain measured data is available about the system to be modeled. The usability of fuzzy tools in time-critical systems is limited by the lack of any systematic method for determining the complexity of handling the system. Moreover, in order to better approximate the modeled system, one can be tempted to overestimate the needed

granularity (e.g. the number of antecedent fuzzy sets), which results in huge and redundant systems with too many rules and too high complexity.
An especially critical situation is, when due to failures or an alarm appearing in the modeled/processing system, the required reaction time is significantly shortened and one has to make decisions before the needed and sufficient information arrives or the processing can be completed. In such cases, it is an obvious requirement to provide enough computational power but the achievable processing speed is highly influenced by the precedence, timing, and data access conditions of the processing itself. It seems to be unavoidable even in the case of extremely careful design to get into situations where the shortage of necessary data and/or processing time becomes serious. Such situations may result in a critical breakdown of monitoring and/or diagnostic systems [S52].
The temporal shortage of time/resources may lead to evaluation of only a subset of the fuzzy rules. On the order hand, the significance of the rules can highly be dependent on the actual inputs; thus the estimation of the accuracy of the approximation, which is of primary importance from interpretation point of view, can become problematic. Without evaluating all of the rules or making further considerations, it is always possible, that the most significant ones are left out, i.e. this way does not offer any real solution to the problem.
In such cases, anytime techniques can be applied advantageously to carry on continuous operation and to avoid critical breakdowns. These systems are able to provide short response time and are able to maintain the information processing even in cases of missing input data, temporary shortage of time, or computational power [6], [13], [14]. Anytime processing tries to handle the case of too many abrupt changes and their consequences in the signal processing, monitoring, diagnostics, or larger scale embedded systems [13]. The idea is that if there is a temporal shortage of computational power and/or there is a loss of some data the actual operations should be continued to maintain the overall performance "at lower price", i.e., information processing based on algorithms and/or models of simpler complexity should provide outputs of acceptable quality to continue the operation of the complete system. The accuracy of the processing will be temporarily lower but possibly still enough to produce data for qualitative evaluations and supporting decisions. Thus, "anytime" processing provides short response time and is very flexible with respect to the available input information and computational power. Accordingly, the aim of these techniques is to ensure the continuous operation of the system in case of changing circumstances and to provide optimal overall performance for the whole system.
Anytime systems utilize mainly two types of processing: the first one is the iterative approach, which is much easier to use and needs less consideration however its applicability is limited by the availability of adequate iterative methods. The second technique uses a modular architecture which makes possible the usage of any algorithm/method in anytime environment, however with a burden of need for extra pre-considerations and decisions [S72].
Fuzzy and NN systems can be applied in anytime systems, as well. Embedding fuzzy and NN systems in anytime systems extends the advantages of the Soft Computing (SC) approach with the flexibility with respect to the available input information and computational power. There are mathematical tools, like Singular Value Decomposition (SVD), which offer a universal scope for handling the complexity problem by anytime operations, however only within the frame of modular architectures, which makes their operation less flexible [S72]. The extension of the processing of fuzzy and NN structures towards the iterative evaluation is highly welcome.
In this chapter we deal with the applicability of SC models in dynamically changing, complex, time-critical, anytime systems. The analyzed models are generated by using (Higher Order) Singular Value Decomposition ((HO)SVD). This technique provides a uniform frame for a family of modeling methods, and results in low (optimal or nearly optimal) computational complexity, easy realization, robustness, a possibility to maintain continuous operation, and to cope with the limits arising in the system or in its environment. Furthermore, the accuracy can also easily and flexibly be increased and we do not need any a priori or expert knowledge about the system to be modeled. It can be used either when a mathematical description (possibly too complex to be

handled) or when only (possibly partial, inaccurate, and uncertain) measurement data is available. A new transformation opening the possibility for iterative type evaluation of product-sum-gravity-singleton consequents (PSGS) fuzzy systems will also be presented.
The chapter is organized as follows: In Section 2 the generalized idea of anytime processing is introduced. Section 3 summarizes the basics of Singular Value Decomposition. Section 4 is devoted to the SVD based exact and non-exact complexity reduction of neural networks and fuzzy models while Section 5 presents the transformation of PSGS fuzzy systems to iterative models which can directly be operated in an an anytime way. Section 6 deals with the usage of fuzzy and NN models in anytime systems: both complexity reduction and the improvement of the approximation are discussed. Finally, in Section 7 the most fundamental elements of anytime control are outlined.

4.2 Anytime processing

Today, there are an increasing number of applications where the computing must be carried out on-line, with guaranteed response time and limited resources. Moreover, the available time and resources are not only limited but can also change during the operation of the system.
Good examples are the modern computer-based diagnostics and monitoring systems, which are able to supervise complex industrial processes and determine appropriate actions in case of failures or deviation from the optimal operational mode. In these systems, usually the model of the faultless system is used for the diagnosis. The evaluation of the model must be carried out on-line, thus the model needs not only to be correct, but also treatable by the limited resources during limited time. Moreover, if some abnormality occurs in the system's behavior, some kind of fault diagnostic task should also be completed, which means the reallocation of a part of the finite resources from the evaluation of the system model to this task. Also in case of an alarm signal, lower response time may be needed.
Several alternative solutions, like complexity reduction techniques, application of less accurate evaluations or decreasing the granulation of the model (e.g. by reduction of the sampling rate in the signal processing scheme), can be proposed if due to a temporary shortage of computer power the processing can not be performed in time (temporal shortage of computing power/time).
Another, more problematic consequence of system failures is the case of missing input data, when due to temporary overload of certain communication channels, the input data fail to arrive in time or are lost. If the output of the processing is still required the usual approach is to utilize some prediction mechanism. This means that based on previous data the information processing units try to generate estimations.
In Subsection 4.2.1 we outline two, the standard and a new, algorithm types that are suitable for anytime processing due to their adjustable evaluation time (and accuracy). Subsection 4.2.2 is devoted to the measurement of the dependence of error/speed characteristics of such algorithms. This characteristics is needed to schedule subtasks of complex anytime systems in a (nearly) optimal way, for which standard methods are discussed in 4.2.3. Although, the applicability of these techniques are limited by the complexity of the system and furthermore, the standard local compilation can be used only in case of SISO system modules. To overcome these problems, in 4.2.4 two new compilation methods, the hierarchical and the output-based incremental compilations are presented which can be used in MISO systems, as well. In subsection 4.2.5 some open questions are briefly discussed.

4.2.1 Alternatives of anytime processing

Iterative processing

In recourse, data, and time insufficient conditions, the so-called anytime algorithms and systems [14] can be used advantageously. They are able to provide guaranteed response time and are flexible with respect to the available input data, time, and computational power. This flexibility makes these systems able to work in changing circumstances without critical breakdowns in the performance. Naturally, while the information processing can be maintained, the complexity must be reduced, thus the results of the computing become less accurate [S68].
The algorithms/computing methods, which are suitable for anytime usage, have to fulfill the following requirements:

- Low complexity: for the optimized behavior the results must be produced with as little computing as possible, i.e. the redundant calculations must be omitted.
- Changeable, guaranteed response time/computational need and accuracy: the achievable accuracy of the results and the necessary amount of computing time/resources must be flexibly changeable and usually known in advance.
- Known error: the error, originating from the necessary model "fitting" (e.g. complexity reduction resulting in non-exact solutions) must also be known, to be able to find the optimal or at least still acceptable solution in the given circumstances, and to be able to compute the resultant error of the outputs.

Iterative algorithms are popular tools in anytime systems, because their complexity can easily and flexibly be changed. The idea is that based on the previous, less accurate approximation of the result, we improve the evaluation and produce a new, better approximation of the output. In some cases, this may also mean the improvement of the approximation (measurement, modeling, etc.) scheme itself. Concerning anytime systems, the main point is that these algorithms always give some, possibly not accurate result and more and more accurate results (better approximations) can be obtained, if the calculations are continued.
A further advantageous aspect of iterative algorithms is that we don't have to know the time/resource-need of a certain configuration in advance. The calculations can be started without any pre-considerations and the procedure can be continued until the results are needed. Then, by simply stopping the calculations, feasible results are obtained. (If the procedure does not produce "continuously" the output, we can apply a memory storing the last (in the given circumstances best) approximation.)

The new modular architecture ([S1], [S5])

Unfortunately, the usability of iterative algorithms is limited. There is a wide range of problems, which can be solved by iterative algorithms, however adequate iterative evaluation methods can not always be found. Even if some kind of iterative evaluation method is available, the accuracy of the results is often unknown: we only know, that the algorithm gives more and more accurate results, but we do not know, how much time is needed to achieve a given accuracy or what will be the rate of error if the calculations are stopped at a given point.
Because of this limitation, a general technique for the application of a wide range of other types of computing methods/algorithms in anytime systems has been suggested in [S72]. A wider class of problems can be addressed with this technique, however at the expense of lower flexibility and a need for extra planning and considerations.
The frame is based on a modular architecture (see Fig. 4.1).

Definition 4.1. An anytime modular architecture is composed of modules realizing the subtasks of a given problem. Each module of the system offers several implementations (characterized by different attribute-values) for a certain task. These units within a given module have uniform

interface (same set of input, output, and solve the same problem) but differ in their computational need and accuracy. An expert system is monitoring the actual circumstances (tasks to complete, achievable time/resources, needed accuracy, etc.) with the help of a set of shortage indicators, in order to choose the adequate configuration, i.e. the units to be used.

Thus, the whole system is optimized rather than the individual modules: in some cases it can be more advantageous to temporarily reduce the computational complexity and accuracy of some parts of the system and rearrange the resources to other, more important tasks.

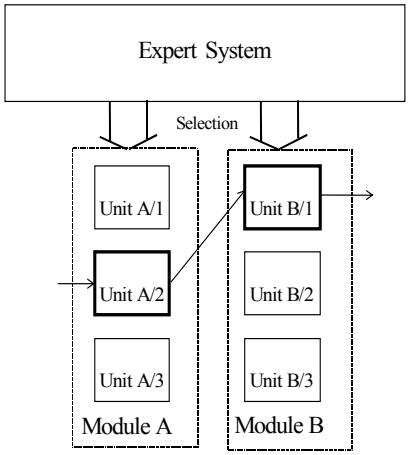


Fig. 4.1 Anytime system with modular architecture

Although, the units implementing a certain task may have different internal structure, from several points of view (e.g. transient behavior, see [S72]) it is advantageous if they are built of similar structure. In this case, the adaptation or change between the units means only the change of some parameter set.

4.2.2 Measurement of speed/accuracy characteristics (performance profiles)

Anytime algorithms are algorithms that can be executed for different running times with a tradeoff between time allocation and output quality. They are characterized by relations between execution times and the quality of the input or the output. These relations, or rather mappings, are called performance profiles [15], [16]; they are represented as one-, two-, and sometimes multidimensional tables.

Several types of *profiles* can be defined and constructed, the main distinction being which parameter is present as part of the input in the relation: input quality and/or output quality. If input quality is used, then the profile is *conditional*. The input-output relation may be treated statistically, in which case the output of the profile is a *distribution* of probability. Output quality is the parameter of this distribution and thus of the profile itself. Time is a parameter of every profile. The most important profile types are the Performance Profile (PP), the Conditional Performance Profile (CPP), the Performance Distribution Profile (PDP), the Conditional Performance Distribution Profile (CPDP), the Expected Performance Profile (EPP), and the Conditional Expected Performance Profile (CEPP) (Table 4.1). (Note that anytime literature uses the notation CPP mostly for Conditional Performance Distribution Profiles.)

Quality is the measure of the "goodness" of any given object in some aspect based on the possible values of the given object. Quality may be defined for simple numbers and for complex and abstract structures.

The *quality function* is mapping from object values to quality values. Together with this, the (time dependent) output function, and a measurement method, performance profiles can be constructed.

Table 4.1 Performance profiles

	Function	Name
Deterministical	$q(t)$	PP
	$q(q_{in}, t)$	CPP
Statistical	$p(t)[q_{out}]$	PDP
	$p(q_{in}, t)[q_{out}]$	CPDP
Expected	$q_E(t)$	EPP
	$q_E(q_{in}, t)$	ECPP

Since using algorithms with decreasing quality (or expected quality) by increasing time allocation is pointless, profiles must possess an *increasing monotonic* property along time allocation. Similarly, any *increasing input quality* is assumed to cause *non-decreasing output quality*.
Other profile types are also definable. For details on dynamic profiles and time-dependent planning under uncertainty, see [1], [3], [18], and [7].

4.2.3 Compilation of complex anytime algorithms

Complex anytime systems, like other complex systems, can be decomposed into elementary anytime or non-anytime modules to facilitate design. Systems operate as whole entities, therefore compilation is needed to handle and operate these systems. Compilation compiles numerous elementary algorithms into a so-called *composite* that acts as an elementary module and has one or more profiles and a special structure containing elementary allocation times for modules involved in compilation. Because a composite and an elementary module have the same type of description and, usually, the same behavior, a composite may be both the output of a compilation step and its input.
The *compilation process* compiles elementary modules into one composite. This process may involve one or more *compilation step*s, which make transitional composites and contain composites in their input.
Let expression $C\{S_1, S_2, ..., S_k, B_1, B_2, ..., B_n\}$ denote a single compilation step in which elements of sets S_i $(i = 1..k)$ and all elements B_i $(i = 1..n)$ are compiled into one composite.
For modules having at least one input driven by an output of another module, only conditional profiles (CPPs (ECPPs) and CPDPs) can be used.

Compilation of two single-input-single-output anytime modules
For systems with two single-input-single-output (SISO) anytime modules connected serially and using CPPs, output quality is formulated as follows:

$$q_2(q_1[q_{in}, T_1], T_2), \tag{4.1}$$

where q_{in} is the (omittable) quality of the system input, the second module (with index 2) is connected after the first one, q denotes conditional performance profiles, and T stands for time allocations.
The composite profile is constructed so that for every composite allocation and composite input quality, the elementary allocation pair with the highest composite output quality is selected from possible pairs.
Using CPPs:

$$q(q_{in}, T) = \max_{T_1} q_2(q_1[q_{in}, T_1], T - T_1), \tag{4.2}$$

where $T = T_1 + T_2$, obviously.

If modules are characterized by CPDPs, then better output quality means better expected output quality, because statistical profiles operate with probability distributions:

$$q(q_{in},T)[q_{out}] = \arg\max_{E} \max_{T_1} E\{\, p(q_{in},T_1,T-T_1)[q_{out}] \,\}$$
$$p(q_{in},T_1,T_2)[q_{out}] = \sum_{i=1}^{N_{q_1}} \left(p_1(q_{in},T_1)[q_{1out,i}] \cdot p_2(q_{1out,i},T-T_1)[q_{out}] \right), \quad (4.3)$$

where q_{in} and q_{out} denote the system input and output, respectively. N_{q_1} and N_{q_2} nominate the extent of elementary profiles among the quality dimension.
We focus here only on compilation with CPPs.

Global Compilation of single-input-single-output modules
The above method for two modules can be extended to multi-modular systems. Note, however, that the complexity of the algorithm grows exponentially with the number of modules, limiting its practical applicability.
Assume system Y with N number of elementary modules $M_1,...,M_N$ connected in *any structure* and characterized by CPPs. Let M_N be the notation of the module at system output. The output quality of the composite is expressed as follows:

$$q_C(q_{in},T_1,...,T_N) = q_N(...,q_1[...,T_1],...,q_{N-1}[...,T_{N-1}],...,T_N), \quad (4.4)$$

where $q_i(...,T_i)$ denotes the CPP of the i-th module. q_C is a compound function in which the elementary profiles and compound functions are the subfunctions.
The composite profile is formulated as

$$q(q_{in},T) = \max_{T_1,...,T_{N-1}} q_C(q_{in},T_1,...,T_{N-1},T-\sum_{i=1}^{N-1} T_i) \quad (4.5)$$

Unfortunately, because of its complexity figure, this method does not appear applicable for more than 3 modules.

Compilation of MISO modules by local compilation
Using SISO modules alone enables us to create chain structured systems only. More generally, tree systems can be made of multiple-input-single-output (MISO) modules (Fig.4.2). These tree structures are effectively compiled by the *local compilation* introduced by Zilberstein ([15]) and discussed in the next section.

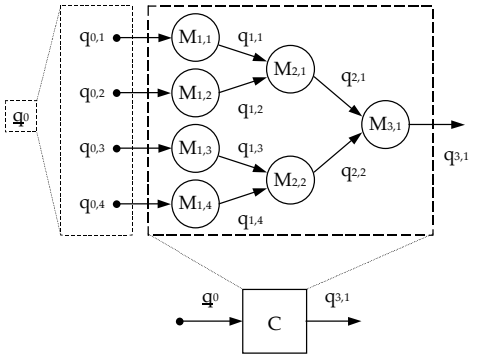


Fig.4.2 Tree-structured system

Local compilation compiles a module and modules connected to its immediate input into one composite. Local compilation produces a global optimum with pseudo-polynomial complexity under the following assumptions:

1. Modules have conditional profiles
2. Profiles are monotonic non-decreasing functions of input quality
3. System has a tree structure with bounded degree
4. CPDPs have the input linearity property

The proof is given in [15]. Tree and chain anytime systems are discussed briefly in [20].
The example in Fig. 4.2 may be compiled by the following *compilation process*:

$$C_{1,1} = C^{(L)}\{M_{1,1}, M_{1,2}, M_{2,1}\}$$

$$C_{1,2} = C^{(L)}\{M_{1,3}, M_{1,4}, M_{2,2}\}$$

$$C = C^{(L)}\{C_{1,1}, C_{1,2}, M_{3,1}\}$$

where operator $C^{(L)}$ denotes a local *compilation step* that is expressed for the compilation of modules $M_{1,1}$, $M_{1,2}$, and $M_{2,1}$, for example, in the following form:

$$q_{C,2,1}(q_{0,1}, q_{0,2}, T_{2,1}) = \max_{T_{1,1}, T_{1,2}} q_{2,1}[q_{1,1}(q_{0,1}, T_{1,1}), q_{1,2}(q_{0,2}, T_{1,2}), T_{2,1} - T_{1,1} - T_{1,2}] \qquad (4.6)$$

This formula differs from the general formula of global compilation (4.5) only in the arrangement of sub-functions and not in complexity, a tree structure with a degree of more than 3 has the same complexity as a chain with as many modules as the degree.
If the degree of the tree is quite high and/or the system has a structure other than a tree, the use of local compilation raises difficulties. Local compilation processes the graph *from the inputs,* so modules with multiple outputs cannot be compiled in such a way.

4.2.4 New compilation methods

Hierarchical compilation ([S24], [S100], [S97], [S103])
System graphs can also be processed *from output,* and subsystems with special interfaces can be compiled independently then added to the rest of the system. The idea of processing subsystems as independent systems leads to the novel hierarchical compilation and the new output-based incremental compilation.
Definition 4.2: Selecting a subsystem *S* with *single output* and *multiple input* from any given *directed acyclic graph-represented* (*DAG-represented)* anytime system *Y* and replacing the subsystem with its global, local, or hierarchical compiled composite is called *hierarchical compilation*.
Hierarchical compilation may yield the optimality of global compilation if certain assumptions are made:

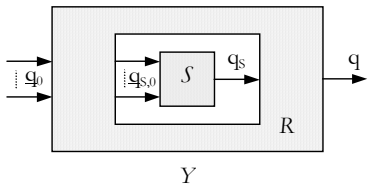


Fig. 4.3 Hierarchical compilation

1. *DAG represented system*
2. *Input monotonity (along both time and input quality dimensions)*
3. *Modules described by CPPs*

Theorem 4.1: Hierarchical compilation provides global optimum under the above assumptions. ([S24], [S100])

The proof of Theorem 4.1 needs the usage of two additional lemmas.

Lemma 4.1: For any monotonic increasing functions *f* and *g* and any vector v the formula below is true:

$$\forall x_1 < x_2 \mid f[x_1, \underline{v}] \le f[x_2, \underline{v}] \quad : \qquad \max_{\underline{u}} f[g(\underline{u}, \underline{v}), \underline{v}] = f[\max_{\underline{u}} g(\underline{u}, \underline{v}), \underline{v}] \tag{4.7}$$

Proof:
Let $\underline{u}_m$ and $\underline{u}_M$ denote the arguments belonging to the left and the right side of equation (4.7):

$$\begin{aligned} &\max_{\underline{u}} f[g(\underline{u}, \underline{v}), \underline{v}] = f[g(\underline{u}_m, \underline{v}), \underline{v}] \\ &f[\max_{\underline{u}} g(\underline{u}, \underline{v}), \underline{v}] = f[g(\underline{u}_M, \underline{v}), \underline{v}] \end{aligned} \tag{4.8}$$

By the maximum operation yielding $\underline{u}_M$:

$$g(\underline{u}_m, \underline{v}) \le g(\underline{u}_M, \underline{v}) \tag{4.9}$$

With input monotonity of f :

$$f[g(\underline{u}_m, \underline{v}), \underline{v}] \le f[g(\underline{u}_M, \underline{v}), \underline{v}] \tag{4.10}$$

By the maximum operation yielding $\underline{u}_m$:

$$f[g(\underline{u}_M, \underline{v}), \underline{v}] \le f[g(\underline{u}_m, \underline{v}), \underline{v}] \tag{4.11}$$

The last two equations prove the theorem.

Lemma 4.2: *Global compilation creates an increasing monotonic conditional profile from increasing monotonic conditional profiles.*

Proof: Let *S* be the subsystem and its global compiled profile:

$$q_S^{(G)}(\underline{q}_0, T) = \max_{T_1, \ldots, T_{N-1}} q_S\left(\underline{q}_0, T_1, \ldots, T_{N-1}, T - \sum_{i=1}^{N-1} T_i\right), \tag{4.12}$$

where $\underline{q}_0$ denotes the inputs of *S* and T expressed by T_i-s. q_S is a compound function which has two types of sub-functions:

1. Functions (profiles) of SISO and MISO modules and composites
2. Functions (profiles) SIMO and MIMO modules (composites can have only one output)

Assume that all sub-functions possess the assumptions given in Theorem 4.1. In this case, increasing of any input causes that the sub-function immediately after increases or reserves its output value (non-decreasing). This increasing output value causes the sub-function of the next "level" to give a non-decreasing output, etc. If the output quality is better with another allocation set, then by the maximum operation it also produces better quality. Consequently, if every

elementary module fulfills the assumptions of Theorem 4.1 then so does their composite, too. Finally, the assumptions also hold for the subsystem composite.

Proof of Theorem 4.1:

Let system Y be divided into two parts: MISO subsystem S and remainder R (Fig. 4.3). The number of modules in Y (N) is thus divided into two sets: $M_{S,1},...,M_{S,NS}$ are in S and $M_{R,1},...,M_{R,NR}$ are in R. $NR+NS=N$ is true. Time allocations for modules in S and R are $T_{S,1},...,T_{S,NS}$ and $T_{R,1},...,T_{R,NR}$, respectively. The global compilation of Y is:

$$q^{(G)}(\underline{q}_0,T)=\max_{T_1,...,T_{N-1}} q\left(\underline{q}_0,T_1,...,T_{N-1},T-\sum_{i=1}^{N-1}T_i\right). \tag{4.13}$$

By separating the allocations of S and R:

$$q^{(G)}(\underline{q}_0,T)=\max_{T_{S,1},...,T_{S,NS-1},T_{R,1},...,T_{R,NR}} q\left(\underline{q}_0,T_{S,1},...,T_{S,NS-1},T-\sum_{i=1}^{NS-1}T_{S,i}-\sum_{i=1}^{NR}T_{R,i,},T_{R,1},...,T_{R,NR}\right) \tag{4.14}$$

where T_S is expressed by other allocations. Hierarchical compilation of Y using S is:

$$q^{(H)}(\underline{q}_0,T)=\max_{T_{R,1},...,T_{R,NR}} q\left(\underline{q}_0,...,q_S^{(G)}\left[\underline{q}_{S,0},T-\sum_{i=1}^{NR}T_{R,i}\right],...,T_{R,1},...,T_{R,NR}\right), \tag{4.15}$$

where T_S is expressed by T and T_R-s. Expand (4.15) using (4.14):

$$q^{(H)}(\underline{q}_0,T)=\max_{T_{R,1},...,T_{R,NR}} q\left(\underline{q}_0,...,\max_{T_{S,1},...,T_{S,NS-1}} q_S\left[\underline{q}_{S,0},T_{S,1},...,T_{S,NS-1},T-\sum_{i=1}^{NR}T_{R,i}-\sum_{i=1}^{NS-1}T_{S,i}\right],\right.$$
$$\left.,...,T_{R,1},...,T_{R,NR}\right). \tag{4.16}$$

q and q_S are monotonic non-decreasing functions of their arguments, every $T_{S,i}$ is in the inner maximum function, and $\underline{q}_{S,0}$ depends on $T_{R,i}$-s only, thus, Lemma 4.1 can be used

$$q^{(H)}(\underline{q}_0,T)=\max_{T_{R,1},...,T_{R,NR}}\max_{T_{S,1},...,T_{S,NS-1}} q\left(\underline{q}_0,...,q_S\left[\underline{q}_{S,0},,T_{S,1},...,T_{S,NS-1},T-\sum_{i=1}^{NR}T_{R,i}-\sum_{i=1}^{NS-1}T_{S,i}\right],\right.$$
$$\left.,...,T_{R,1},...,T_{R,NR}\right), \tag{4.17}$$

Merging maximum functions and eliminating decomposition yields the following :

$$q^{(G)}(\underline{q}_0,T)=\max_{T_{S,1},...,T_{S,NS-1},T_{R,1},...,T_{R,NR}} q\left(\underline{q}_0,T_{S,1},...,T_{S,NS-1},T-\sum_{i=1}^{NS-1}T_{S,i}-\sum_{i=1}^{NR}T_{R,i},T_{R,1},...,T_{R,NR}\right) \tag{4.18}$$

,

Equation (4.18) is equivalent with (4.14) so the proof is ready.

Output-based incremental compilation and monitoring of contract algorithms

There are two types of anytime algorithms: *interruptible* and *contract*. Interruptible algorithms may be interrupted after any elapsed run time to provide valid output. Contract algorithms do not give useful output values before the *contract time expires*. Contract time is an allocation calculated before the algorithm is activated. Interruptible algorithms can be constructed from contract algorithms ([18]).

Anytime monitoring is the process of distributing allocations for elementary algorithms to optimize operation of the anytime system using information about the current state. The distribution itself is part of the *scheduling process* responsible for managing executable code under timing considerations as in real-time systems.

Anytime monitoring may be *passive* or *active*. Passive monitoring assigns allocation times *before* an anytime (sub)system is activated. Active monitoring assigns allocation times *during* the execution of modules and subsystems, i.e., interruptible anytime operation applies active monitoring.

Method 4.2 Output based incremental compilation ([S24], [S97], [S103]**):**

Let $M_1,...,M_N$ be a set of elementary contract modules characterized by their CPPs and connected in some structure. C denotes the system composite (created by compiling all M_i-s into one composite). Assume one system output and a total system allocation obtained from the current state, e.g., by utility-driven computation, nominated as T. The purpose of a system is to perform certain tasks by executing elementary operations in a well-defined order called an *execution order*. These operations in our anytime system are implementation functions of anytime modules. The task is accomplished when the output value appears in the system output.

When the scheduler is about to execute the subsequent (actual) elementary module, it must calculate an allocation for the module. This allocation depends only on the remainder of modules to be scheduled and the input quality of the actual module, i.e., the remainder may be treated as an individual anytime module or rather a composite. This remainder, called a *residual composite*, is used to obtain an allocation for it by the above total allocation calculation and allocation for the actual module is given by compilation. This mode of scheduling requires the creation of these residual composites.

Let $\{E_1,...,E_N\}$ denote the execution order, formally a list of modules sorted by the order of activation. The following expression is true:

$$\bigcup_{i=1}^{N} E_i \equiv \bigcup_{i=1}^{N} M_i \,, \tag{4.19}$$

If the system has only one output, then compilation steps implemented by hierarchical compilation can be used to build residual composites. The first composite will be the last module in the activation list; the second composite will be the result of the last two modules in the list, etc. This compilation process is called *output based incremental compilation (OBIC)*, since the actual residual composite and the forthcoming module are compiled together at each compilation step. The order of modules is the reverse of the activation list. Assuming a chain structure, the OBIC and the scheduling of active monitoring based on the OBIC are formulated as follows (formulas are too complicated for any other structure):

$$\begin{aligned} &T_i = T_{i,q,T}(q_{0,i}, T_{R,i}) \\ &T_{i,q,T}(q_{0,i}, T_{R,i}) \in C_{R,i} \\ &T_{R,i} = F_T[S, q_{R,i}(q_{0,i}, T_{R,i}), t_i, i, ...]' \\ &t_1 = 0 \end{aligned} \tag{4.20}$$

where

i: Step variable, $i = 1..N$.

T_i: Allocation for the module executed at the *i*-th step

$T_{i,q,T}(q_{0,i}, T_{R,i})$: Allocation table of the next module

$T_{R,i}$: Residual allocation

F_T: Function or algorithm giving the residual allocation

S: Set of state parameters

t_i: Elapsed absolute time from system activation at the *i*-th scheduling step

$C_{R,i}$: The *i*-th residual composite

The formulas of qualities are:

$$\begin{aligned} &q_{0,i} = q_{i-1}(q_{0,i-1}, T_{i-1}) \\ &q_1(q_{0,1}, T_1) \equiv \begin{cases} q_1(T_1) & : P_1 \equiv PP \\ q_1(q_0, T_1) & : P_1 \equiv CPP \end{cases}, \\ &q_i(q_{0,i}, T_i) \in C_{R,i} \end{aligned} \tag{4.21}$$

where P_i is the performance profile of module E_i and q_0 is the quality of system input.

$$\begin{aligned} &C_{R,i} = C^{(H)}\{E_i, C_{R,i+1}\} \\ &C_{R,N} \equiv E_N \end{aligned}, \tag{4.22}$$

where $C^{(H)}\{\}$ denotes hierarchical compilation.

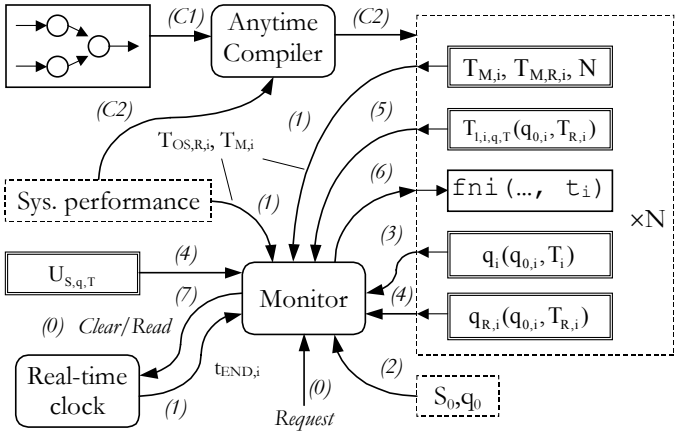


Fig. 4.4 Block diagram of active monitoring

Fig. 4.4. shows the block diagram of the monitoring system. The numbers in braces denote the sequence of the monitoring steps:

(0) Arrived request, real-time clock is set to zero

(1) Obtaining the time needed by the monitor for the scheduling of the residual composite. Calculating the approximate values of the current state (using the initial values). Computing the time allocation needed by the operating system in the interval of executing the residual composite (e.g. interrupts)

(2) Determining the input quality of the first module to execute and the initial state values

(3) Counting the input quality of the next elementary module by using the input quality and time allocation of the previously scheduled module (The value is simply read from the CPP of that module)
(4) Computing the optimal allocation ($T_{R,i}$) for the residual composite. (Utility driven computation may be used.) The input quality computed in step *(3)* is applied
(5) Obtaining the allocation for the forthcoming elementary module by using the input quality and the residual allocation ($T_{1,i,q,T}(q_{0,i}, T_{R,i})$ is used.)
(6) Executing the implementation function of the module with allocation obtained by the previous step
(7) Saving the current value of the real-time clock ($t_{\mathrm{END},i\text{-}1}$ for the next scheduling step)

4.2.5. Open questions

Anytime schemes based on feedback systems unavoidably suffer from transients. These well-known phenomena are due to the dynamic nature of the processing structures applied. With the spreading of time critical, reconfigurable, and embedded systems, transient handling, covering both active and passive methods, has become an important research area.
Active transient management methods (like state initialization and anti-transient signal injection) can over perform the passive methods when the run-time interaction is possible and the system has free computational power to run the transient management algorithms. These methods manipulate the system when the reconfiguration happens in order to decrease the transient effects. At the same time, active transient management methods have serious drawbacks: additional (sometimes remarkable) computational power may be required, and their application field is narrower than that of the passive methods [21].
Both parameter and structure adaptations generate transients. The nature of these transients depends not only on the transfer function of the structures to be implemented, but also on the actual implementation of the processing structure [22]. For this very reason the implementation of anytime algorithms must be performed using structures having good transient behavior. This structure dependency is strongly related to the „energy distribution" within the processing structure. It can be proved that orthogonal structures meet the general conditions of having relatively small transients [22].
Another aspect to be considered is that if the system is reconfigured between the old and the new configurations through intermediate steps, the transients may decrease. The transients depend on the selection of the number and the actual locations of these intermediate steps. Unfortunately, the simplest method, i.e., the linear interpolation does not ensure good results in most cases [22]. Fuzzy decision making may help to find the optimal strategy of the adaptation; however, controlling transients in reconfigurable systems is still an important area of investigations and research.
We have to mention here a further non-negligible effect of anytime systems. The temporary reduction of complexity causes the reduction of the accuracy, as well. In some cases, like SVD-based complexity reduction, the error can easily be estimated, however further computations must be made to obtain the so called resultant error (the error of the whole system): the errors of the different modules, as well as the path of data and error through the modules has to be considered.
For the calculation of the resultant error the error transfer functions of the modules must also be known. If module *B* uses the results of module *A*, then, if the accuracy of module *A* reduces, the accuracy of the output of *B* will reduce, as well. The $y=f_E(x)$ error transfer function means, that if the input of the module has an absolute error x, then the output of the module will have an additional absolute error y. The error along the data-path is cumulative. It is supposed, that the internal error of the modules, originating from inexact computations, noise, etc., can be modeled as an additive error component in the output of the module. Thus, in the example above, if module

A has an error E_A and module B has an error E_B then the resultant error on the output of B is $f_{e,B}(E_A)+E_B$, where $f_{e,B}$ is the error expansion function of module B.

In dynamic systems, further considerations must be made. In these cases, the error can spread not only in space, but also in time in the system, namely, the temporary reduction of accuracy may effect the operation of the system even after the restoration of the original accuracy. If the system contains additive memory elements then the error theoretically will never disappear from the system. (However, it can also be proved that if the absolute value of the error expansion function $f_E()$ is always less then one then the error will sooner or later disappear from the system, but if its absolute value is greater or equal than one then the effect of a temporal accuracy-reduction will influence all later results ([23].)

4.3 Singular Value Decomposition

Singular Value Decomposition (SVD) has successfully been used to reduce the complexity of a large family of systems based on both classical and soft techniques [24]. An important advantage of the SVD reduction technique is that it offers a formal measure to filter out the redundancy (exact reduction) and also the weakly contributing parts (non-exact reduction). This implies that the degree of reduction can be chosen according to the maximum acceptable error corresponding to the current circumstances. In case of multi-dimensional problems, the SVD technique can be defined in a multidimensional matrix form, i.e. Higher Order SVD (HOSVD) can be applied.

SVD is serious candidate to overcome the complexity problems arising in modeling of complex systems where we either have an analytical description of the system (possibly too complicated to be handled), or the system is represented only by input-output sample pairs. In these cases, we can build a model approximating the system using local (linear) models. Such techniques include Takagi-Sugeno (TS) fuzzy model approximation [25] or polytopic model approximation (PMA). These methods have theoretically a universal approximation property; however, it can not really be exploited because they have an exponentially increasing complexity growing with the number of parameters. This means that if the number of the local units is bounded then the built model will only be an approximation of the original system. Thus, we have to find a balance between the computational complexity and the accuracy. On the other hand, after ensuring a needed or given accuracy, which may mean the application of a huge number of local models, the computational complexity can be reduced by applying some kind of exact or non-exact complexity reduction method like SVDR.

Definition 4.3. (SVDR): The SVD based complexity reduction algorithm is based on the decomposition of any real valued $\underline{\underline{F}}$ matrix:

$$\underline{\underline{F}}_{(n_1\times n_2)} = \underline{\underline{A}}_{1,(n_1\times n_1)}\,\underline{\underline{B}}_{(n_1\times n_2)}\,\underline{\underline{A}}^{T}_{2,(n_2\times n_2)} \tag{4.23}$$

where $\underline{\underline{A}}_k$, k=1,2 are orthogonal matrices ($\underline{\underline{A}}_k\,\underline{\underline{A}}_k^T = \underline{\underline{E}}$), and $\underline{\underline{B}}$ is a diagonal matrix containing the λ_i singular values of $\underline{\underline{F}}$ in decreasing order. The maximum number of the nonzero singular values is $n_{SVD} = \min(n_1, n_2)$. The singular values indicate the significance of the corresponding columns of $\underline{\underline{A}}_k$. Let the matrices be partitioned in the following way:

$$\underline{\underline{A}}_k = \left| \underline{\underline{A}}^{r}_{k,(n_k\times n_r)} \quad \underline{\underline{A}}^{d}_{k,(n_k\times(n_k-n_r))} \right| \text{ and } \underline{\underline{B}} = \begin{vmatrix} \underline{\underline{B}}^{r}_{(n_r\times n_r)} & 0 \\ 0 & \underline{\underline{B}}^{d}_{((n_1-n_r)\times(n_2-n_r))} \end{vmatrix},$$

where r denotes "reduced" and $n_r \le n_{SVD}$.

If $\underline{\underline{B}}^d$ contains only zero singular values then $\underline{\underline{B}}^d$ and $\underline{\underline{A}}_k^d$ can be dropped: $\underline{\underline{F}} = \underline{\underline{A}}_1^r \underline{\underline{B}}^r \underline{\underline{A}}_2^{rT}$. If $\underline{\underline{B}}^d$ contains nonzero singular values, as well, then the $\underline{\underline{F}}' = \underline{\underline{A}}_1^r \underline{\underline{B}}^r \underline{\underline{A}}_2^{rT}$ matrix is only an approximation of $\underline{\underline{F}}$ and the maximum difference between the values of $\underline{\underline{F}}$ and $\underline{\underline{F}}'$ equals ([26])

$$E_{RSVD} = \left|\underline{\underline{F}} - \underline{\underline{F}}'\right| \le \left(\sum_{i=n_r+1}^{n_{SVD}} \lambda_i\right) \underline{\underline{1}}_{(n_1 \times n_2)} \quad . \tag{4.24}$$

Theorem 4.2. (4.24) can be improved and better upper bound can be given for the SVD-based matrix reduction. The error of the matrix reduction can be estimated by the maximum element of the error matrix ([S22], [S70], [S76], [S89]):

$$E_{RSVD} = \max(\underline{\underline{E}}), \text{ where } \underline{\underline{E}} = \left|\underline{\underline{F}} - \underline{\underline{F}}'\right| = \sum_{p=n_r+1}^{n_{SVD}} \lambda_p \underline{a}_{1,k} \underline{a}_{2,k}^T$$

Proof:

Utilizing that the first n_r columns of matrices $\underline{\underline{A}}_k$ és $\underline{\underline{A}}_k^r$ are similar, the elements of matrix $\underline{\underline{F}}'$ can be calculated as

$$f'_{i,j} = \sum_{p=1}^{n_r} a_{1,i,p} b_{p,p} a_{2,j,p} = \sum_{p=1}^{n_r} a_{1,i,p} \lambda_p a_{2,j,p} \; .$$

$$\underline{\underline{F}}' = \sum_{p=1}^{n_r} b_p \underline{a}_{1,k} \underline{a}_{2,k}^T \; . \tag{3.41}$$

Utilizing that the last n_k-n_r columns of matrices $\underline{\underline{A}}_k$ és $\underline{\underline{A}}_k^d$ are similar, the maximum difference between the elements of matrices $\underline{\underline{F}}$ and $\underline{\underline{F}}'$ is

$$e_{i,j} = \left|f_{i,j} - f'_{i,j}\right| = \sum_{p=n_r+1}^{n_{SVD}} a_{1,i,p} b_{p,p} a_{2,j,p} = \sum_{p=n_r+1}^{n_{SVD}} \lambda_p a_{1,i,p} a_{2,j,p} \; ,$$

$$\underline{\underline{E}} = \left|\underline{\underline{F}} - \underline{\underline{F}}'\right| = \sum_{p=n_r+1}^{n_{SVD}} \lambda_p \underline{a}_{1,k} \underline{a}_{2,k}^T \; , \tag{3.42}$$

Thus, the error of the matrix reduction can be estimated by the maximum element of error matrix $\underline{\underline{E}}$.

For higher order cases Higher Order SVD (HOSVD) can be applied in a similar way (see e.g. [S78]).
Here we have to remark that if SVD is applied to a two dimensional matrix then it can be proved that the resulting matrix of lower rank will be the best approximation of the original matrix in least-squares sense (minimum $\|L_2\|$ norm of the error, i.e. the reduction is "optimal"). In case of higher dimension matrices where HOSVD is applied, the minimum property does not hold anymore. We can only state that the "significant" singular values will have the "lower" indices. However, in most cases if there is a considerable difference among the singular values HOSVD results in an approximation which is "very near" to the optimal one.
Remark: The above procedure can be also applied on a sub-matrix of $\underline{\underline{E}}$. This offers a possibility to give a better error bound for a subspace of the system, which can be advantageous in cases when the output of the system may vary in a large domain and we need lower bounds for the relative error of the system in the different domains.

4.4 Exact and Non-Exact Complexity Reduction of Fuzzy Models and Neural Network Based on SVD

SVD based complexity reduction can be applied to various types of soft computational systems such as Product-Sum-Gravity-Singleton Consequent (PSGS) [27], Product-Sum-Gravity-Non-Singleton Consequent (PSGN), Takagi-Sugeno fuzzy models [25], and generalized neural networks (GNN) [S83]. Combined with the Singular Value Decomposition technique, they are excellent tools for "anytime" operations. By using SVD, not only the "sequence" of the rules can be defined but also the extent in which they contribute to the mapping. To cope with the limits arising in the system or in its environment, determined by the computational need of the remaining truncated model, one can appropriately abandon the less significant part of the rule base and give the approximation error. Their further advantages are that they are suitable for modeling a large class of non-linear problems and may have relatively low (optimal) computational complexity. It can be proved that after exact reduction the remaining computational complexity is minimal and the computational need and error of the further, non-exact reductions can easily be obtained. The unavoidable extra calculations, caused by the SVD algorithm itself, can be pre-executed off-line.
Here, as examples the complexity reduction of the product-sum-gravity fuzzy systems with singleton consequents (PSGS), that of the Takagi-Sugeno fuzzy models, and the reduction of generalized neural networks (GNN) are presented. Extensions of the SVD based reduction to PSGN fuzzy models and for systems having extremely large rule-bases, where the size of the rule-base is greater then the available operational memory, can be found in [S57], [S62], and [28].

4.4.1 Reduction of PSGS fuzzy rule-bases with SVD

Let us first enumerate some definitions which will be used in the followings. Let $\mu_i(x)\ i=1,2,...n$ be functions, defined on some compact domain, and $\{\mu_i(x)\ i=1,2,...n\}$ a function set. The following properties of the function set can be defined:

Definition 4. 4. *Sum normalization* (SN): the $\{\mu_i(x)\ i=1,2,...n\}$ function set is SN, if

$$\sum_{i=1}^{n} \mu_i(x) = 1,$$

for every x of the domain. A matrix is SN, if the sums of the rows are 1.

Definition 4. 5. *Nonnegativeness* (NN): the $\{\mu_i(x)\ i=1,2,...n\}$ function set is NN, if

$$\mu_i(x) \geq 0, \quad i=1,2,...,n$$

for every x of the domain. A matrix is NN, if all of its elements are nonnegative.

Definition 4. 6. *Normality* (NO): the $\{\mu_i(x)\ i=1,2,...n\}$ function set is NO, if it is SN and NN, and each of the $\mu_i(x)$ functions takes the value 1 at some point within the domain. A matrix is NO, if it is SN and NN, and has an element of value 1 in every column.
If the $\mu_i(x)$ functions are membership functions, then the SN and NN properties ensure that the membership values fall into the [0,1] interval. Besides, the SN property means, that if a given x has a high membership value in one of the fuzzy sets, then it has low membership values in the other fuzzy sets which may be a typical property in case of the antecedent fuzzy sets of a fuzzy inference system. The NO property implies, that the membership functions are normalized.

Definition 4.7. *Ruspini-partition*: the fuzzy sets, defined by the $\mu_i(x)\ i=1,2,...,n$ membership functions are in Ruspini-partition, if $\forall x: \sum_{i=1}^{n} \mu_i(x) = 1$; the membership functions satisfy the SN and NN conditions.

Definition 4.8. *Product-sum-gravity (PSG) fuzzy inference with singleton consequences* (PSGS): The antecedent fuzzy sets are in Ruspini-partition, the consequent fuzzy sets are singletons. The rules are

$R_{i_1,\ldots,i_N}$: If x_1 is A_{1,i_1}, x_2 is A_{2,i_2},…, and x_N is A_{N,i_N} then $y = y_{i_1,\ldots,i_N}$.

The output of the system equals

$$y^* = \sum_{i_1,\ldots,i_N} y_{i_1,\ldots,i_N} \prod_{j=1}^{N} \mu_{X_{j,i_j}}(x_j^*)$$

where $x_1,\ldots,x_N$ are the input variables, A_{k,i_k} is the antecedent fuzzy set of the *k*-th input variable in the $R_{i_1,\ldots,i_N}$ rule and $y_{i_1,\ldots,i_N}$ is the consequence of the rule, in this case a singleton value.

Definition 4.9. *Product-sum-gravity fuzzy inference with non-singleton consequences* (PSGN): The antecedent fuzzy sets are in Ruspini-partition, the consequences are fuzzy sets. The rules are:

$R_{i_1,\ldots,i_N}$: If x_1 is A_{1,i_1}, x_2 is A_{2,i_2},…, and x_N is A_{N,i_N} then y is $Y_{i_1,\ldots,i_N}$,

where $x_1,\ldots,x_N$ are the input variables, A_{k,i_k} is the antecedent fuzzy set of the *k*-th input variable in the $R_{i_1,\ldots,i_N}$ rule and $Y_{i_1,\ldots,i_N}$ is the consequence fuzzy set of the rule. The consequent fuzzy sets can be characterized by two parameters: their center of gravity ($y_{i_1,\ldots,i_N}$) and their area ($s_{i_1,\ldots,i_N}$). The fuzzification method is singleton, during the inference, product *T*-norm and sum *S*-norm are used, thus the result of the inference is

$$y^* = \frac{\sum_{i_1,\ldots,i_N} b_{i_1,\ldots,i_N} \prod_{j=1}^{N} \mu_{X_{j,i_j}}(x_j^*)}{\sum_{i_1,\ldots,i_N} s_{i_1,\ldots,i_N} \prod_{j=1}^{N} \mu_{X_{j,i_j}}(x_j^*)}, \text{ where } b_{i_1,\ldots,i_N} = y_{i_1,\ldots,i_N} s_{i_1,\ldots,i_N}$$

Definition 4.10. *Near SPGS fuzzy inference*: The only difference compared to the PSGS fuzzy systems is, that the input fuzzy sets are not in Ruspini-partition. Because the input fuzzy sets are not in Ruspini-partition, the result of the inference will take the form of

$$y^* = \frac{\sum_{i_1,\ldots,i_N} y_{i_1,\ldots,i_N} \prod_{j=1}^{N} \mu_{X_{j,i_j}}(x_j^*)}{\prod_{j=1}^{N} \mu_{X_{j,i_j}}(x_j^*)}.$$

Consider a fuzzy rule base with two inputs, where the antecedent fuzzy sets are in Ruspini-partition and the consequence fuzzy sets are singletons. Thus, the rules are

$R_{i,j}$: If x_1 is $A_{1,i}$ and x_2 is $A_{2,j}$ then $y = y_{i,j}$, where $i = 1\ldots n_1$ and $j = 1\ldots n_2$.

The fuzzyfication method is singleton and during the inference, product *T*-norm and sum *S*-norm are used. The result of the fuzzy inference in case of the input values (x_1^*, x_2^*)

$$y^* = \sum_{i_1,i_2} y_{i_1,i_2} \mu_{A_{1,i1}}(x_1^*) \mu_{A_{2,i2}}(x_2^*). \qquad (4.25)$$

Let $\underline{\underline{F}}$ be a matrix, containing the $y_{i,j}$ elements, then apply the SVDR procedure discussed in Section 3 to obtain $\underline{\underline{F}} \approx \underline{\underline{F}}' = \underline{\underline{A}}_1^r \underline{\underline{B}}^r \underline{\underline{A}}_2^{rT}$, where $\underline{\underline{A}}_1$ and $\underline{\underline{A}}_2$ are SN (Sum-Normalized: the sum of

each row equals to one) and NN (Non-Negative), and "*r*" denotes "reduced". The new rule-base takes the form of

$$R'_{i,j}: \text{If } x_1 \text{ is } A'_{1,i} \text{ and } x_2 \text{ is } A'_{2,j} \text{ then } y = y'_{i,j},$$

where $i = 1 \ldots n_1^r$, $j = 1 \ldots n_2^r$, $y'_{i,j}$ are the elements of $\underline{\underline{B}}$, and the new membership functions can be obtained as

$$\mu_{A'_{k,i}}(x_k) = \sum_j \mu_{A_{k,j}}(x_k) A_{k,j,i}, \tag{4.26}$$

$A_{k,j,i}$ stands for the *(j,i)*-th element of $\underline{\underline{A}}_k$.

The reduced rule-base contains only $n_1^r * n_2^r$ rules instead of $n_1 * n_2$ and the error can be estimated from the discarded singular values.

Definition 4.11. (HOSVD): The SVDR method can be extended to *n*-dimension cases by applying HOSVD, as follows ($(\underline{\underline{A}}_1, \cdots, \underline{\underline{A}}_n, \underline{\underline{F}}^r) = HOSVDR(\underline{\underline{F}})$): In this case the reduction can be made in *n* steps, in every step one dimension of matrix $\underline{\underline{F}}$, containing the $y_{i_1,\ldots,i_n}$ consequences is reduced. The first step sets $\underline{\underline{F}}_1 = \underline{\underline{F}}$. In the followings, $\underline{\underline{F}}_i$ is generated by step *i*-1. The *i*-th step of the algorithm (*i*>1) is

1. Spreading out the *n*-dimensional matrix $\underline{\underline{F}}_i$ (size: $n_1^r \times \ldots \times n_{i-1}^r \times n_i \times \ldots \times n_n$) into a two-dimensional matrix $\underline{\underline{S}}_i$ (size: $n_i \times (n_1^r * \ldots * n_{i-1}^r * n_{i+1} * \ldots * n_n)$).
2. Reduction of $\underline{\underline{S}}_i$: $\underline{\underline{S}}_i \approx \underline{\underline{A}}_i \underline{\underline{B}} \underline{\underline{A}}'^T_i = \underline{\underline{A}}_i \underline{\underline{S}}^*_i$, where the size of $\underline{\underline{A}}_i$ is $n_i \times n_i^r$ and the size of $\underline{\underline{S}}^*_i$ is $n_i^r \times (n_1^r * \ldots * n_{i-1}^r * n_{i+1} * \ldots * n_n)$.
3. Re-stacking $\underline{\underline{S}}^*_i$ into the *n*-dimensional matrix $\underline{\underline{F}}_{i+1}$ (size $n_1^r \times \ldots \times n_i^r \times n_{i+1} \times \ldots \times n_n$), and continuing with step 1. for $\underline{\underline{F}}_{i+1}$.

The consequences of the reduced rule-base are the elements of $\underline{\underline{F}}_n$ and the new membership functions are $\mu_{A'_{k,i}}(x_k) = \sum_j \mu_{A_{k,j}}(x_k) A_{k,j,i}$.

The reduced rule base contains only $n_1^r * \ldots * n_n^r$ rules instead of $n_1 * \ldots * n_n$.

The following theorem illustrates the advantageous property of SVD reduction:

Theorem 4.3. The maximum error (e_{FSVD}) of the SVD based reduction of PSGS fuzzy systems with non-linear antecedent fuzzy set in Ruspini partition can not exceed the sum of the discarded singular values at any point ([S22], [S76]).

Proof of the 2-dimensional case:

The statement we prove reads:

$$e_{FSVD} \le \sum_{i=n_r+1}^{n_{SVD}} \lambda_i \tag{4.27}$$

The output of the non-reduced system can be written as (see also 4.25)

$$y^* = \sum_{i_1,i_2} y_{i_1,i_2}\mu_{1,i_1}(x_1^*)\mu_{2,i_2}(x_2^*) = \left|\mu_{1,1}(x_1^*) \quad \cdots \quad \mu_{1,n_1}(x_1^*)\right| \left[y_{i_1,i_2}\right] \begin{vmatrix} \mu_{2,1}(x_2^*) \\ \vdots \\ \mu_{2,n_2}(x_2^*) \end{vmatrix} = \underline{\mu_1}(x_1^*)\underline{\underline{F}}\,\underline{\mu_2}(x_2^*) \quad (4.28)$$

where matrix $\underline{\underline{F}}$ contains consequents y_{i_1,i_2} and vectors $\underline{\mu_k}(x_k^*)$ correspond to the membership values of input x_k^* in the antecedent fuzzy sets of the *k*-th (*k*=1,2) input variable. Let apply SVDR to obtain the reduced rule-base

$$R'_{i_1,i_2}: \text{If } x_1 \text{ is } X'_{1,i_1} \text{ and } x_2 \text{ is } X'_{2,i_2} \text{ then } y = y'_{i_1,i_2},$$

where $i_k = 1...n_k^r$, az y'_{i_1,i_2} are the elements of $\underline{\underline{F}}'$ (see also (4.24)) and the new membership functions can be obtained as

$$\mu'_{k,j} = \sum \mu_{k,j}(x_k) a_{k,j,i}\,, \tag{4.29}$$

where $a_{k,j,i}$ correspond to the elements of matrix $\underline{\underline{A}}_k$.
The output of the new, reduced rulebase is (from (4.28) and (4.29))

$$\bar{y}^* = \sum_{i_1=1}^{n_1^r}\sum_{i_2=1}^{n_2^r} \mu_{1,i_1}^r(x_1^*)\mu_{2,i_2}^r(x_2^*)b_{i_1,i_2}^r = \left[\mu_{1,1}^r(x_1^*) \quad \cdots \quad \mu_{1,n_1^r}^r(x_1^*)\right]\left[b_{i_1,i_2}^r\right] \begin{bmatrix} \mu_{2,1}^r(x_2^*) \\ \vdots \\ \mu_{2,n_2^r}^r(x_2^*) \end{bmatrix} = \underline{\mu_1}(x_1^*)\underline{\underline{A}}_1\underline{\underline{B}}\underline{\underline{A}}_2\,\underline{\mu_2}(x_2^*) \tag{4.30}$$

The error of the reduction

$$\left|y^* - \bar{y}^*\right| = \left|\underline{\mu_1}(x_1^*)\underline{\underline{F}}\,\underline{\mu_2}(x_2^*) - \underline{\mu_1}(x_1^*)\underline{\underline{A}}_1\underline{\underline{B}}\underline{\underline{A}}_2\,\underline{\mu_2}(x_2^*)\right| = \left|\underline{\mu_1}(x_1^*)(\underline{\underline{F}} - \underline{\underline{A}}_1\underline{\underline{B}}\underline{\underline{A}}_2)\underline{\mu_2}(x_2^*)\right| \tag{4.31}$$

The elements of $\left|\underline{\underline{F}} - \underline{\underline{A}}_1\underline{\underline{B}}\underline{\underline{A}}_2\right|$ correspond to the errors in the grid points and they are less or equal then the sum of the discarded singular values (E_{RSVD})

$$\left|\underline{\underline{F}} - \underline{\underline{A}}_1\underline{\underline{B}}\underline{\underline{A}}_2\right| = \left|\underline{\underline{F}} - \underline{\underline{F}}'\right| \le E_{RSVD}\underline{\underline{1}}_{(n_1\times n_2)}\,. \tag{4.32}$$

Using (4.24) and that the original antecedent fuzzy sets are in Ruspini-partition follows that

$$\left|y^* - \bar{y}^*\right| \le \left|\underline{\mu_1}(x_1^*)\begin{bmatrix} E_{RSVD} & \cdots & E_{RSVD} \\ \vdots & \ddots & \vdots \\ E_{RSVD} & \cdots & E_{RSVD} \end{bmatrix}\underline{\mu_2}(x_2^*)\right| = \left|\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\mu_{1,i_1}(x_1^*)\mu_{2,i_2}(x_2^*)E_{RSVD}\right| = E_{RSVD}\,, \tag{4.33}$$

Proof of the *n*-dimensional case:

Here we have to prove

$$e_{FSVD} \le \left(\sum_{k=1}^{n} \sum_{j=1}^{d_k} \lambda_{k,j} \right) \tag{4.34}$$

where d_k corresponds to the number of discarded singular values in the k-th step, $\lambda_{k,i}$ stands for the j-th discarded singular value in step k.
The rulebase of a PSGS fuzzy system with N inputs is

$$R_{i_1,\ldots,i_N}: \text{If } x_1 \text{ is } X_{1,i_1} \text{ and } x_2 \text{ is } X_{2,i_2} \text{ and } \ldots \text{ and } x_N \text{ is } X_{N,i_N} \text{ then } y = y_{i_1,\ldots,i_N},$$

where $i_j = 1 \ldots n_j$, n_j stands for the number of antecedent fuzzy sets in the universe of the j-th input, X_{j,i_j} corresponds to the i_j.-th antecedent fuzzy set of input j, and $y_{i_1,\ldots,i_N}$ is a real valued output of rule $R_{i_1,\ldots,i_N}$.
The result of the inference with the original rule-base can be written as

$$y^* = \sum_{i_1=1}^{n_1} \ldots \sum_{i_n=1}^{n_n} \mu_{1,i_1}(x_1^*) \ldots \mu_{n,i_n}(x_n^*) y_{i_1,i_2,\ldots,i_n} = \sum_{i_2=1}^{n_2} \ldots \sum_{i_n=1}^{n_n} \mu_{2,i_2}(x_2^*) \ldots \mu_{n,i_n}(x_n^*) b_{i_2,\ldots,i_n}, \tag{4.35}$$

where

$$b_{i_2,\ldots,i_n} = \sum_{i_1=1}^{n_1} \mu_{1,i_1}(x_1^*) y_{i_1,i_2,\ldots,i_n} = \begin{bmatrix} \mu_{1,1}(x_1^*) & \cdots & \mu_{1,n_1}(x_1^*) \end{bmatrix} \underline{\underline{S}}_1 .$$

The size of matrix $\underline{\underline{S}}_1$ is $n_1 \times (n_2 \ldots n_n)$ and it contains values $y_{i_1,i_2,\ldots,i_n}$.
During the reduction, matrix $\underline{\underline{S}}_1$ and with it values $b_{i_2,\ldots,i_n}$ are estimated: $\underline{\underline{S}}_1 \approx \underline{\underline{A}}_1 \underline{\underline{B}} \underline{\underline{A}}_1'^T = \underline{\underline{A}}_1 \underline{\underline{S}}_1^*$. The error of this estimation is less or equal then the sum of the discarded singular values ($E_{RSVD,1}$)
The error of the estimation of values $b_{i_2,\ldots,i_n}$ equals to

$$\begin{aligned} \left| b_{i_2,\ldots,i_n} - \bar{b}_{i_2,\ldots,i_n} \right| &= \left| \begin{bmatrix} \mu_{1,1}(x_1^*) & \cdots & \mu_{1,n_1}(x_1^*) \end{bmatrix} \left(\underline{\underline{S}}_1 - \underline{\underline{A}}_1 \underline{\underline{B}} \underline{\underline{A}}_1'^T \right) \right| \\ &\le \left| \begin{bmatrix} \mu_{1,1}(x_1^*) & \cdots & \mu_{1,n_1}(x_1^*) \end{bmatrix} \begin{bmatrix} E_{RSVD,1} & \cdots & E_{RSVD,1} \\ \vdots & \ddots & \vdots \\ E_{RSVD,1} & \cdots & E_{RSVD,1} \end{bmatrix} \right| \end{aligned} \tag{4.36}$$

$$\left| b_{i_2,\ldots,i_n} - \bar{b}_{i_2,\ldots,i_n} \right| \le \left| \sum_{i_1=1}^{n_1} \mu_{1,i_1}(x_1^*) E_{RSVD,1} \right| = E_{RSVD,1},$$

Because the original antecedent fuzzy sets are in Ruspini-partition.
Using (4.35) and (4.36), the error of the first step of the reduction:

$$\left| y^* - \bar{y}^* \right| = \left| \sum_{i_2=1}^{n_2} \ldots \sum_{i_n=1}^{n_n} \mu_{2,i_2}(x_2^*) \ldots \mu_{n,i_n}(x_n^*) (b_{i_2,\ldots,i_n} - \bar{b}_{i_2,\ldots,i_n}) \right| \le \sum_{i_2=1}^{n_2} \ldots \sum_{i_n=1}^{n_n} \mu_{2,i_2}(x_2^*) \ldots \mu_{n,i_n}(x_n^*) E_{RSVD,1} = E_{RSVD,1} \tag{4.37}$$

Because the original antecedent fuzzy sets are in Ruspini-partition. Thus, the error of the first step will not exceed the sum of the discarded singular values.
For the following steps, the same procedure and the same error bound is valid. The only difference is that in (4.37), not only the original but also the reduced antecedent fuzzy sets appear. Thus, for

the validity of the error bound, the reduced antecedent fuzzy sets must be in Ruspini-partition, as well. This can be ensured, if matrices $\underline{\underline{A}}_k$ are sum-normalized (SN).

So, the overall error of the reduction at any point, is less or equal the the sum of the discarded singular values, even if the antecedent fuzzy sets are non-linear

$$e_{FSVD} \leq \left(\sum_{k=1}^{n} \sum_{j=1}^{d_k} \lambda_{k,j} \right).$$

4.4.2 Reduction of near PSGS fuzzy rule-bases with SVD

While the SVD-based complexity reduction can be applied to PSGS fuzzy systems, in case of rule-bases, constructed from expert knowledge, the input fuzzy sets are not always in Ruspini-partition. This section presents how the SVD-based reduction can be extended to "near PSGS" fuzzy systems, where the input fuzzy sets are not in Ruspini-partition.

Theorem 4.4. Non-exact HOSVD complexity reduction can be applied to "Near PSGS" fuzzy systems [S98].

Proof:

Consider a "near PSGS" fuzzy system with two inputs, where the numbers of antecedent fuzzy sets are n_1 and n_2, respectively. The antecedent fuzzy sets are not in Ruspini-partition and the consequents are singletons. Thus, the rule-base contains $n_1 * n_2$ rules:

$$R_{i_1,i_2}: \text{If } x_1 \text{ is } X_{1,i_1} \text{ and } x_2 \text{ is } X_{2,j} \text{ then } y = y_{i_1,i_2}.$$

The fuzzification method is singleton, during the inference product *T*-norm, sum *S*-norm, and center-of-gravity defuzzification are used, so the result of the inference in case of the input values (x_1^*, x_2^*) is

$$y^* = \frac{\sum_{i_1,i_2} y_{i_1,i_2} \mu_{1,i_1}(x_1^*) \mu_{2,i_2}(x_2^*)}{\sum_{i_1,i_2} \mu_{1,i_1}(x_1^*) \mu_{2,i_2}(x_2^*)} = \frac{\sum_{i_1,i_2} y_{i_1,i_2} \mu_{1,i_1}(x_1^*) \mu_{2,i_2}(x_2^*)}{\sum_{i_1,i_2} \mu_{1,i_1}(x_1^*) \mu_{2,i_2}(x_2^*) * 1} = \frac{N_0}{D_0}, \tag{4.38}$$

where $\mu_{k,i_k}(x_k)$ is the membership function of the i_k-th antecedent set of the *k*-th input. Thus, the SVD-based complexity-reduction algorithm given for PSGS fuzzy systems can not be applied in this case. However, (4.38) has the same form, as a PSGN fuzzy system (see Definition 4.9), where the centers of gravity of the consequent fuzzy sets are y_{i_1,i_2}, and their areas (s_{i_1,i_2}) are 1. Thus, the reduction can be solved similarly, as in case of PSGN systems (Definition 4.11. HOSVD) and the result of the reduction of a near PSGS fuzzy system will be a PSGN fuzzy system.

Let us define the 3-dimensional matrix $\underline{\underline{F}}$ (with size $n_1 \times n_2 \times 2$) as follows

$$f_{i_1,i_2,1} = y_{i_1,i_2} \text{ and } f_{i_1,i_2,2} = 1. \tag{4.39}$$

First, matrix $\underline{\underline{F}}$ is spread out to the 2-dimensional matrix $\underline{\underline{S}}_1$ (size: $n_1 \times 2n_2$), and $\underline{\underline{S}}_1$ is reduced with the the SVD-based matrix-reduction algorithm:

$$\underline{\underline{S}}_1 \approx \underline{\underline{A}}_1 \underline{\underline{B}} \underline{\underline{A}}_1'^T = \underline{\underline{A}}_{1(n_1 \times n_1^r)} \underline{\underline{S}}^*_{1(n_1^r \times 2n_2)}, \tag{4.40}$$

where matrix $\underline{\underline{A}}_1$ is SN and NN.

In the second step, matrix $\underline{\underline{S}}_1^*$ is restacked to the 3-dimensional matrix $\underline{\underline{F}}_2$ (size: $n_1^r \times n_2 \times 2$). Then $\underline{\underline{F}}_2$ is spread out to the 2-dimensional matrix $\underline{\underline{S}}_2$ (size: $n_2 \times 2n_1^r$), and $\underline{\underline{S}}_2$ is reduced:

$$\underline{\underline{S}}_2 \approx \underline{\underline{A}}_2 \underline{\underline{B}} \underline{\underline{A}}'^T_2 = \underline{\underline{A}}_{2(n_2 \times n_2^r)} \underline{\underline{S}}^*_{2(n_2^r \times 2n_1^r)}, \tag{4.41}$$

where matrix $\underline{\underline{A}}_1$ is SN and NN.

By the restacking of $\underline{\underline{S}}_2$ we get the 3-dimensional matrix $\underline{\underline{F}}^*$ (size $n_1^r \times n_2^r \times 2$).

The elements of the original $\underline{\underline{F}}$ can be estimated as:

$$f_{i_1,i_2,i_3} \approx f'_{i_1,i_2,i_3} = \sum_{\bar{i}_1=1}^{n_1^r} \sum_{\bar{i}_2=1}^{n_2^r} a_{1,i_1,\bar{i}_1} a_{2,i_2,\bar{i}_2} f^*_{\bar{i}_1,\bar{i}_2,i_3}, \tag{4.42}$$

where the $f^*_{\bar{i}_1,\bar{i}_2,i_3}$ values are the elements of $\underline{\underline{F}}^*$, and the $a_{k,i_k,\bar{i}_k}$ values are the elements of the matrices $\underline{\underline{A}}_k$.

The reduced, PSGN rule-base:

$$R'_{i_1,i_2}: \text{If } x_1 \text{ is } X'_{1,i_1} \text{ and } x_2 \text{ is } X'_{2,i_2} \text{ then } y = Y'_{i_1,i_2},$$

where the new antecedent fuzzy sets are

$$\mu'_{l,i_k}(x_k) = \sum_l a_{k,i_k,l} \mu_{l,i_k}(x_k). \tag{4.43}$$

Because of the SN and NN properties of the $\underline{\underline{A}}_k$ matrices, the values of the new membership functions are in the [0,1] interval and the new antecedent fuzzy sets are in Ruspini-partition. Thus, the reduced system is a PSGN fuzzy system. The consequent fuzzy sets have center of areas of y'_{i_1,i_2} and areas of s'_{i_1,i_2}, which parameters can be determined from the elements of matrix $\underline{\underline{F}}^*$

$$y'_{i_1,i_2} = \frac{f^*_{i_1,i_2,1}}{f^*_{i_1,i_2,2}} \quad \text{and} \quad s'_{i_1,i_2} = f^*_{i_1,i_2,2}. \tag{4.44}$$

The new rule-base contains $n_1^r \times n_2^r$ rules instead of the original $n_1 \times n_2$.

The above-described procedure can be extended to *N*-dimensional cases, as well. In that case matrix $\underline{\underline{F}}$ is *N*+1 dimensional, and the reduction can be made in *N* steps. In every step, $\underline{\underline{F}}$ will be spread along one of its dimensions, and the given dimension will be reduced. The result will be a reduced *N*+1 dimensional matrix, $\underline{\underline{F}}^*$, which contains the parameters of the new consequent fuzzy sets, and the $\underline{\underline{A}}_k$ matrices (*k*=1..*N*), from which the new membership functions can be generated.

Because the reduced rule-base is of different type than the original one, for the determination of the rate of reduction further considerations are needed. The computational need of the original and the reduced rule-bases from Definitions 4.9 and 4.10 :

	Original	Reduced
Calculation of membership functions	$\sum_{i=1}^{N} n_i$	$\sum_{i=1}^{N} n_i$
Multiplications	$N\prod_{i=1}^{N} n_i$	$\sum_{i=1}^{N} n_i n_i^r + (N+1)\prod_{i=1}^{N} n_i^r$

Additions	$2\prod_{i=1}^{N} n_i$	$\sum_{i=1}^{N}(n_i n_i^r - 1) + 2\prod_{i=1}^{N} n_i^r$
Division	1	1

N stands for the number of inputs, n_i and n_i^r are the number of antecedent fuzzy sets of the i-th input variable in the original and reduced rule-bases, respectively.
It can be seen, that in case of a complex system, where N and n_i are high, the computational complexity of the original and the reduced systems will be approximately proportional to $\prod_{i=1}^{N} n_i$ and $\prod_{i=1}^{N} n_i^r$. In case of smaller systems, the whole equations must be considered.

Theorem 4.5. The error bound of the nonexact HOSVD reduction for „near" PSGS fuzzy systems can be given as ([S98])

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{1-E_0}\right| \max\left|y^* - 1\right| \le \left|\frac{\left(\sum_{k=1}^{n}\sum_{j=1}^{d_k}\lambda_{k,j}\right)\prod_{k=1}^{N}\max\{1,U_k\}}{1-\left(\sum_{k=1}^{n}\sum_{j=1}^{d_k}\lambda_{k,j}\right)\prod_{k=1}^{N}\max\{1,U_k\}}\right| \max\left|y^* - 1\right|,$$

Where U_k (k=1,…,N) stand for the upper bounds of the sum of the membership functions.

Proof:

While the SVD-based complexity-reduction of near PSGS fuzzy systems can be solved similarly to the reduction of PSGN fuzzy systems, for the error-bound of the non-exact reduction a new proof is needed, because in case of PSGN fuzzy systems the proof of the error-bound is based on the fact, that the input fuzzy sets are in Ruspini-partition [29].
The output of the reduced system, in the N-dimensional case equals

$$\bar{y}^* = \frac{\sum_{i_1,\ldots,i_N} y'_{i_1,\ldots,i_N}\, s'_{i_1,\ldots,i_N}\, \mu'_{1,i_1}(x_1^*)\ldots\mu'_{2,i_N}(x_N^*)}{\sum_{i_1,\ldots,i_N} s'_{i_1,\ldots,i_N}\, \mu'_{1,i_1}(x_1^*)\ldots\mu'_{2,i_N}(x_N^*)} = \frac{N'_0}{D'_0} \qquad (4.45)$$

The errors of the numerator and denominator can be estimated separately. In the followings, the estimation of the error of the numerator will be shown; the error of the denominator can be estimated similarly.
Let L_k and U_k (k=1,…,N) be the lower and upper bounds of the sum of the membership functions:

$$L_k \le \sum_{i=1}^{n_k} \mu_{k,i}(x) \le U_k \quad \text{for every } x \text{ of the domain.} \qquad (4.46)$$

The numerators of the original system can be written as:

$$N_0 = \sum_{i_2=1}^{n_2} \ldots \sum_{i_N=1}^{n_N} \mu_{2,i_2}(x_2^*)\ldots\mu_{N,i_N}(x_N^*) b_{i_2,\ldots,i_N}, \qquad (4.47)$$

where

$$b_{i_2,\ldots,i_N} = \sum_{i_1=1}^{n_1} \mu_{1,i_1}(x_1^*) y_{i_1,i_2,\ldots,i_N} = \left[\mu_{1,1}(x_1^*) \quad \cdots \quad \mu_{1,n_1}(x_1^*)\right] \underline{\underline{H}}_{1,1} \text{ and } \underline{\underline{S}}_1 = \left|\underline{\underline{H}}_{1,1} \underline{\underline{H}}_{1,2}\right|.$$

The size of matrix $\underline{\underline{S}}_1$ is $n_1 \times (n_2 \ldots n_N 2)$, and it contains the $y_{i_1,\ldots,i_N}$ and $s_{i_1,\ldots,i_N} = 1$ elements, namely, $\underline{\underline{H}}_{1,1}$ contains the $y_{i_1,\ldots,i_N}$ values and $\underline{\underline{H}}_{1,2}$ the $s_{i_1,\ldots,i_N} = 1$ elements.

In the first step of the reduction, matrix $\underline{\underline{S}}_1$ and thus the $b_{i_2,\ldots,i_N}$ elements are estimated: $\underline{\underline{S}}_1 \approx \underline{\underline{A}}_1 \underline{\underline{B}} \underline{\underline{A}}_1'^T = \underline{\underline{A}}_1 \underline{\underline{S}}_1^*$. The error of the estimation ($E_{RSVD,1}$) is not greater, then the sum of the discarded singular values in the first step, $\overline{\lambda_1}$.

The error of the estimation of the $b_{i_2,\ldots,i_N}$ values:

$$\begin{aligned}
\left|b_{i_2,\ldots,i_N} - \bar{b}_{i_2,\ldots,i_N}\right| &= \left|\left[\mu_{1,1}(x_1^*) \quad \cdots \quad \mu_{1,n_1}(x_1^*)\right]\left(\underline{\underline{S}}_1 - \underline{\underline{A}}_1 \underline{\underline{B}} \underline{\underline{A}}_1'^T\right)\right| \\
&\le \left|\left[\mu_{1,1}(x_1^*) \quad \cdots \quad \mu_{1,n_1}(x_1^*)\right] \begin{bmatrix} E_{RSVD,1} & \cdots & E_{RSVD,1} \\ \vdots & \ddots & \vdots \\ E_{RSVD,1} & \cdots & E_{RSVD,1} \end{bmatrix}\right| \\
\left|b_{i_2,\ldots,i_N} - \bar{b}_{i_2,\ldots,i_N}\right| &\le \left|\sum_{i_1=1}^{n_1} \mu_{1,i_1}(x_1^*) E_{RSVD,1}\right| = U_k E_{RSVD,1}.
\end{aligned} \tag{4.48}$$

The error of the numerator in the first step from (4.47) and (4.48):

$$\begin{aligned}
\left|N_0 - N_{0,1}'\right| &= \left|\sum_{i_2=1}^{n_2} \ldots \sum_{i_N=1}^{n_N} \mu_{2,i_2}(x_2^*) \ldots \mu_{N,i_N}(x_N^*)\left(b_{i_2,\ldots,i_N} - \bar{b}_{i_2,\ldots,i_N}\right)\right| \\
&\le \sum_{i_2=1}^{n_2} \ldots \sum_{i_N=1}^{n_N} \mu_{2,i_2}(x_2^*) \ldots \mu_{N,i_N}(x_N^*) U_1 E_{RSVD,1} = U_1 U_2 \ldots U_N E_{RSVD,1} \le U_1 \ldots U_N \overline{\lambda_1}
\end{aligned} \tag{4.49}$$

In the following steps, the error can be estimated similarly, the only difference is, that in (4.49) the antecedent fuzzy sets of the reduced system are also present. Because the antecedent fuzzy sets of the reduced system are in Ruspini-partition, the above described error bound is valid, if $1 \le U_k$ for every *k*. In general, the overall error of the numerators can be estimated as:

$$\left|N_0 - N'_0\right| \le \left(\sum_{k=1}^{N} \sum_{j=1}^{d_k} \lambda_{k,j}\right) \prod_{l=1}^{N} \max\{1, U_l\} = E_0, \tag{4.50}$$

where d_k is the number of discarded singular values in the *k*th step, and $\lambda_{k,j}$ is the *j*th singular value, discarded in the *k*th step.

The error of the denominator can be estimated similarly, the above-described error bound is valid for that, as well:

$$\left|D_0 - D'_0\right| \le \left(\sum_{k=1}^{N} \sum_{j=1}^{d_k} \lambda_{k,j}\right) \prod_{l=1}^{N} \max\{1, U_l\} = E_0, \tag{4.51}$$

Let us define

$$\Delta N_0 = N'_0 - N_0 \text{ and } \Delta D_0 = D'_0 - D_0. \tag{4.52}$$

The error of the output:

$$\left|y^* - \bar{y}^*\right| = \left|\frac{N_0}{D_0} - \frac{N'_0}{D'_0}\right| = \left|\frac{D_0(N_0 + \Delta N_0) - N_0(D_0 + \Delta D_0)}{D_0 D'_0}\right| = \left|\frac{\Delta N_0}{D'_0} - \frac{N_0}{D_0}\frac{\Delta D_0}{D'_0}\right| \quad (4.53)$$

Because $|\Delta N_0| \le E_0$ and $|\Delta D_0| \le E_0$

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{D'_0}\right|\left|\frac{N_0}{D_0}\right| - 1\right| \le \left|\frac{E_0}{D'_0}\right| \max\left|y^* - 1\right| \quad (4.54)$$

If $D'_0 \ge 0$, namely $E_0 \le \prod_{k=1}^{N} L_k$, then from (4.50), (4.51) and (4.54) the error-bound is

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{1 - E_0}\right| \max\left|y^* - 1\right| \le \left|\frac{\left(\sum_{k=1}^{n}\sum_{j=1}^{d_k} \lambda_{k,j}\right)\prod_{k=1}^{N} \max\{1, U_k\}}{1 - \left(\sum_{k=1}^{n}\sum_{j=1}^{d_k} \lambda_{k,j}\right)\prod_{k=1}^{N} \max\{1, U_k\}}\right| \max\left|y^* - 1\right| \quad (4.55)$$

Remark: Based on the value of y^* a better error bound can be given starting of (4.53).
If $|y^*| \le 1$, then

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{D'_0}(1 - y^*)\right| \le \left|\frac{E_0}{D'_0}\right|(1 - y^*) \le E_1(1 - \min|y^*|), \quad (4.56)$$

where $E_1 = \left|\frac{\left(\sum_{k=1}^{n}\sum_{j=1}^{d_k} \lambda_{k,j}\right)\prod_{k=1}^{N} \max\{1, U_k\}}{\prod_{k=1}^{N} L_k - \left(\sum_{k=1}^{n}\sum_{j=1}^{d_k} \lambda_{k,j}\right)\prod_{k=1}^{N} \max\{1, U_k\}}\right|$.

If $y^* \ge 1$, then

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{D'_0}\right|(y^* - 1) \le \left|\frac{E_0}{D'_0}\right|(y^* - 1) \le E_1 y^* \ . \quad (4.57)$$

If $y^* \le -1$, then

$$\left|y^* - \bar{y}^*\right| \le \left|\frac{E_0}{D'_0}\right|(1 - y^*) \le \left|\frac{E_0}{D'_0}\right|\left|y^*\right| \le E_1\left|y^*\right|. \quad (4.58)$$

4.4.3 Reduction of Takagi-Sugeno fuzzy models with SVD ([S3], [S20], [S23])

Takagi-Sugeno (TS) fuzzy modeling is a technique to describe a nonlinear dynamic system using local linearized models [25]. The idea is that the system dynamics is captured by a set of fuzzy implications, which characterize local regions in the state space. The overall fuzzy model, i.e. the description of the whole system is achieved by the convex combination (fuzzy blending) of the

linear models. The combination is usually defined by an array of local models and the *n*-dimensional matrix product of basis functions, which expresses the local dominance of the local models. Using the TS fuzzy model approximation, the controller design and Lyapunov stability analysis reduces to solving the Linear Matrix Inequalities (LMIs) problem. Appropriately chosen operating points, i.e., the number of local linear models and the size of the corresponding regions used in the supervision system can guarantee the stability of the dynamic system [31].
Unfortunately, this latter can be a serious limitation on the applicability of such control schemes because the computational complexity of the system increases exponentially with the number of models. This leads to the same problem as we have discussed in case of PSGS fuzzy models. The solution can be the application of exact (to find the minimum number of the necessary models) and non-exact (to cope with the temporal circumstances) HOSVD based complexity reduction.
TS fuzzy modeling technique can be used both if we have an analytical description of the system, i.e. the system is given e.g. by differential equations or if only input-output samples are given, thus we make a black-box modeling. In the first case, we are to sample the system over a rectangular hyper-grid, which leads to a similar problem as black-box modeling, accept that in this case the samples, i.e. the approximation points can directly serve as linear models, while in the latter case we have to evaluate a Lagrange interpolation to adapt the local models to force the overall model to copy the behavior of the system in the sampling points.
TS fuzzy models are theoretically universal approximators. Despite this advantage, their use is practically limited, because the computational complexity grows exponentially with the number of parameters and the universal approximation property doesn't hold if the number of antecedent sets are limited [33]. Consequently, methods helping to find the minimum number of necessary building units to a given accuracy are highly desirable.
In Fig. 4.5 the block-diagram of a TS fuzzy observer based control scheme is shown [30].

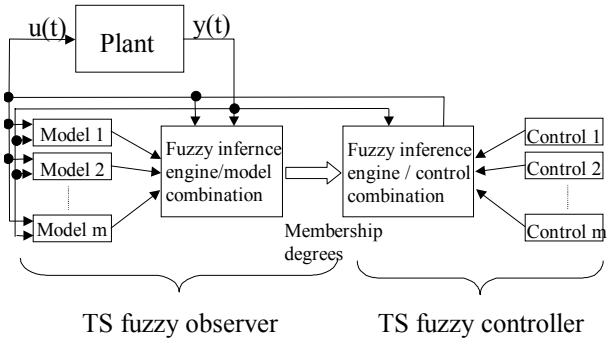


Fig. 4.5 TS fuzzy observer based control scheme

For the fuzzy observer design it is assumed that the fuzzy system model is locally observable. Using the idea of Parallel Distributed Compensation (PDC) [32] a linear time invariant observer can be associated with each rule of the TS fuzzy model:

$$\text{If } \omega \text{ is fuzzy set } A_t \text{ then model } M_t, \tag{4.59}$$

$t = 1..m$ and m is the number of the models.
Model M_t is defined as (see (3.1) in [32]):

$$\left.\begin{aligned} \dot{\underline{\hat{x}}}(t) &= \underline{\underline{A}}_t \underline{\hat{x}}(t) + \underline{\underline{B}}_t \underline{u}(t) + \underline{\underline{L}}_t (\underline{y}(t) - \underline{\hat{y}}(t)) \\ \underline{\hat{y}}(t) &= \underline{\underline{C}}_t \underline{\hat{x}}(t) \end{aligned}\right\} \Rightarrow M_t \tag{4.60}$$

Let us use arbitrary shaped fuzzy sets $A_t : \mu_{A_t}(\omega)$, t=1…T. Actually, fuzzy sets A_t are the weighting functions for the combination of the models. For general view and simpler notation let us define weighting functions $f_t(\omega)$ instead of fuzzy sets $A_t : \mu_{A_t}(\omega)$. So the combination of the models according to the fuzzy rules can be expressed as

$$\hat{M}(\varpi) = \sum_{t=1}^{T} f_t(\omega) M_t \,. \tag{4.61}$$

The t -th generalized model is calculated as:

$$M_t \Rightarrow \begin{vmatrix} \underline{z}_{1,t} \\ \underline{z}_{2,t} \\ \vdots \\ \underline{z}_{L,t} \end{vmatrix} = \begin{vmatrix} \underline{\underline{B}}_{t,1,1} & \underline{\underline{B}}_{t,1,2} & \cdots & \underline{\underline{B}}_{t,1,U} \\ \underline{\underline{B}}_{t,2,1} & \underline{\underline{B}}_{t,2,2} & & \underline{\underline{B}}_{t,1,U} \\ \vdots & & \ddots & \\ \underline{\underline{B}}_{t,L,1} & \underline{\underline{B}}_{t,L,2} & \cdots & \underline{\underline{B}}_{t,L,U} \end{vmatrix} \begin{vmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_U \end{vmatrix}$$

From this, the following general form can be got for the weighed combination of the models

$$\underline{\mathbf{y}}_l(t) = \sum_{t=1}^{T} f_t(\omega) \underline{z}_{l,t}(t) = \sum_{t=1}^{T} f_t(\omega) \sum_{u=1}^{U} \underline{\underline{B}}_{t,l,u} \underline{x}_u(t) \,. \tag{4.62}$$

The structure of the TS fuzzy model based approximation is illustrated in Fig. 4.6.

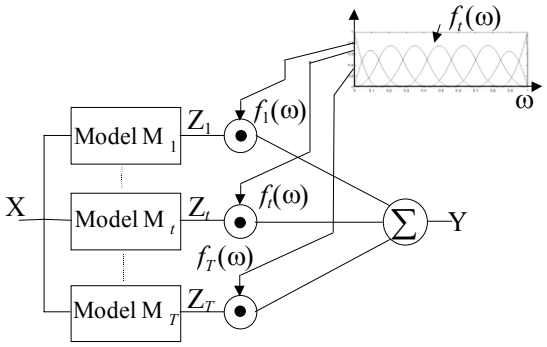


Fig. 4.6 Weighted combination of the models. The number of the models M_t and functions $f_t(\omega)$ equals to T

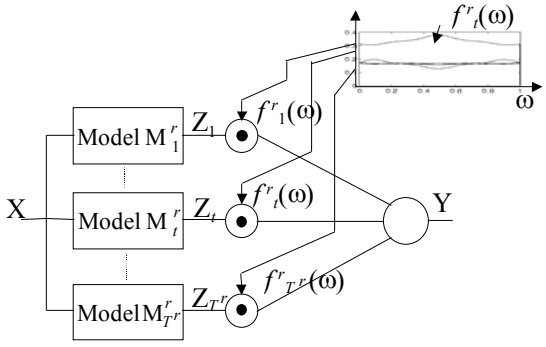


Fig. 4.7 Compressed model structure

Applying SVD based complexity reduction to (8) yields the following form

$$\underline{y}_l(t) = \underline{\underline{A}}_l \sum_{t=1}^{T^r} f_t(\omega) \underline{z}^r_{t,l}(t) = \underline{\underline{A}}_l \sum_{t=1}^{T^r} f^r_t(\omega) \sum_{u=1}^{U} \underline{\underline{B}}^r_{t,l,u} \underline{\underline{C}}^{\mathrm{T}}_u \underline{x}_u(t) \,, \tag{4.63}$$

where "r" denotes "reduced", the sizes of $\underline{\underline{A}}_l$, C_u, and $\underline{\underline{B}}^r_{t,l,u}$ are $O_l \times O^r_l$, $I_u \times I^r_u$ and $O^r_l \times I^r_u$, respectively, further $\forall l : O^r_l \le O_l$, $\forall u : I^r_u \le I_u$ and the number of models is reduced as $T^r \le T$. The reduced form is shown in Fig. 4.7. (For more details see [S20].)

If not only zero singular values are discarded then the effectiveness of the reduction is improved, however reduction error is obtained. The error resulted by SVDR is bounded by the sum of the discarded singular values.

4.4.4 Reduction of generalized neural networks with SVD

Definition 4.12. **Generalized neural networks (GNN).** The classical multi-layer neural network can be generalized if the non-linear transfer functions are moved from the nodes into the links. It results in neurons that apply only a sum operation to the input values, and links that are characterized by possibly non-linear weighting functions instead of simple constant weights (see Fig. 4.8). A further advantage of this generalization is that it makes possible to apply even different weighting functions at the connections.

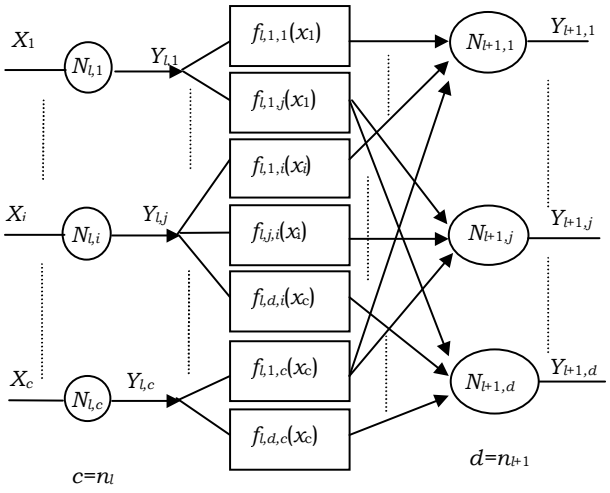


Fig. 4.8 Generalized Neural Network

Let us focus on two neighboring layers l and l+1 of a forward model. Let the neurons be denoted as $N_{l,i}$, $i = 1..n_l$ in layer l, where n_l is the number of the neurons. Further, let input values of $N_{l,i}$ be $x_{l,i,k}$, $k = 1..n_{l-1}$ and its output $y_{l,i}$. The connection between layers l and l+1 can be defined by the $f_{l,j,i}(y_{l,i})$ weighting functions ($j = 1..n_{l+1}$). Thus

$$x_{l+1,j,i} = f_{l,j,i}(y_{l,i}) \tag{4.64}$$

and the output of neuron $N_{l+1,j}$ is

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) \tag{4.65}$$

.

The weighting functions can also be changed during the training: the unknown weighting functions are approximated with linearly combined known functions, where only the linear combination must be trained (Fig. 4.9).

Definition 4.13. **PSGS generalized neural networks (SGNN).** For the approximation of the weighting functions of the GNN, PSGS fuzzy systems are used, with one input and one output

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) = \sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t} \tag{4.66}$$

.

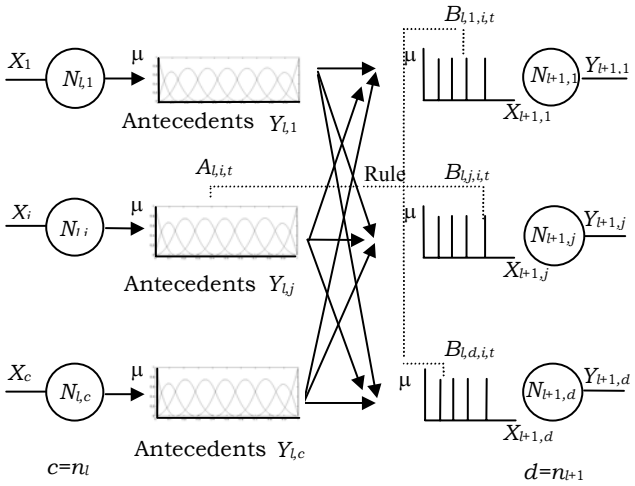


Fig. 4.9 SGNN approximation of the GNN

To reduce the size of a generalized neural network the SVD based complexity reduction can be used. (4.66) can always be transformed into the following form

$$y'_{l+1,j} = \sum_{z=1}^{n_{l+1}^r} a_{l,j,z} \sum_{i=1}^{n_l} \sum_{t=1}^{m_{l,i}^r} \mu_{l,i,t}^r (y_{l,i}) b'_{l,z,i,t} \tag{4.67}$$

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

The reduced form is represented as a neural network with an extra inner layer between layers *l* and *l*+1 (see Fig. 4.10). Between the original layer *l* and the new layer the weighting functions are approximated from the reduced PSGS fuzzy systems, and layer *l+1* simply computes the weighted sum ($a_{l,j,z}$) of the output of the new layer.

The reduction means the reduction of the $\underline{\underline{B}} = \lfloor b_{l,j,i,t} \rfloor$ three-dimensional matrix in two steps by applying the HOSVD reduction algorithms ((Definition 4.11.): In the first step, the first dimension is reduced, and the $a_{l,j,z}$ values are determined while in the second the third dimension is reduced, and the new membership functions are determined. The detailed description of the algorithm can be found in [34].

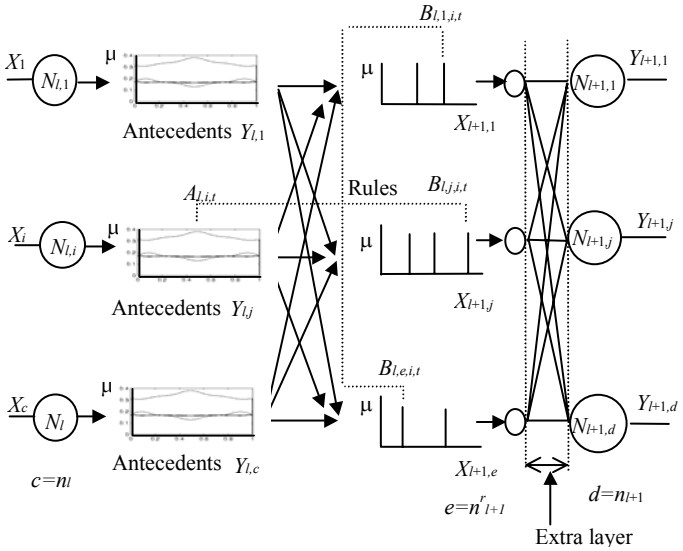


Fig. 4.10 Reduced SGNN

The maximal error of the resulted neural network can be computed from the discarded singular values, considering that the singular values discarded in the first step "count" n_l times ([35]).

Definition 4.14. **PSGN generalized neural networks (NGNN).** For the approximation of the weighting functions of the GNN, PSGN fuzzy systems are used with rules

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

where the consequent fuzzy sets are non-singleton fuzzy sets with centers of gravity of d_t and areas of s_t (PSGN fuzzy inference). The result of the inference is

$$y^* = \frac{\sum_{t=1}^{m} \mu_t(x^*) d_t s_t}{\sum_{t=1}^{m} \mu_t(x^*) s_t} \tag{4.68}$$

In order to approximate the contribution of each neuron to the neurons of the next layer by the PSGN technique, let (4.68) be substituted into (4.65). Let a more general form be defined where all antecedent universes may have different number of antecedent sets:

$$y_{l+1,j} = \sum_{i=1}^{n_l} f_{l,j,i}(y_{l,i}) = \sum_{i=1}^{n_l} \frac{\sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) d_{l,j,i,t} s_{l,j,i,t}}{\sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) s_{l,j,i,t}} = \sum_{i=1}^{n_l} \frac{\sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) b_{l,j,i,t}}{\sum_{t=1}^{m_{l,i}} \mu_{l,i,t}(y_{l,i}) s_{l,j,i,t}} = \sum_{i=1}^{n_l} \frac{N_{l,j,i}}{D_{l,j,i}} \quad . \tag{4.69}$$

Here $m_{l,i}$ is the number of antecedent sets in layer l, $b_{l,j,i,t} = d_{l,j,i,t} s_{l,j,i,t}$, $N_{l,j,i}$ denote the nominators and $D_{l,j,i}$ the denominators. Fig. 4.11 depicts the NGNN representing (4.69).

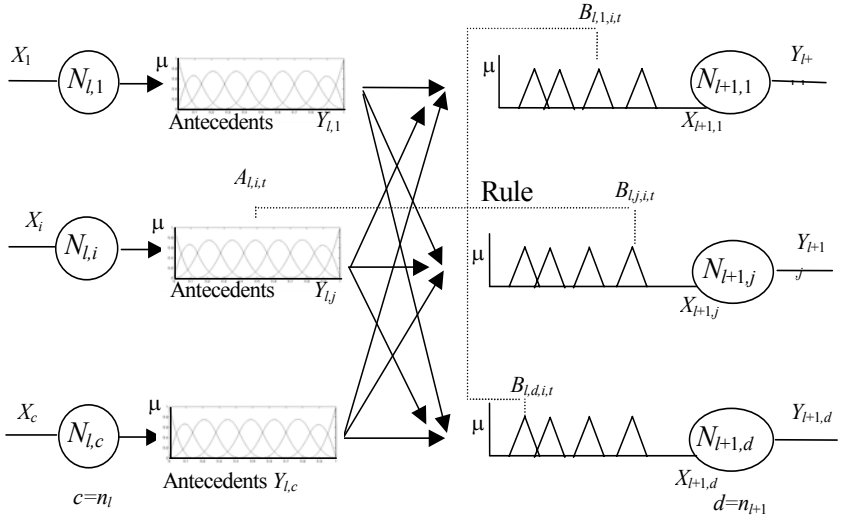


Fig. 4.11 NGNN approximation of the GNN

Theorem 4.6. (4.69) can always be transformed into the form of

$$y'_{l+1,j} = \sum_{i=1}^{n_{l+1}} \frac{\sum_{t=1}^{m^r_{l,i}} \mu^r_{l,i,t}(y_{l,i}) b'_{l,z,i,t}}{\sum_{t=1}^{m^r_{l,i}} \mu^r_{l,i,t}(y_{l,i}) s'_{l,z,i,t}} = \sum_{i=1}^{n_{l+1}} \frac{N'_{l,j,i}}{D'_{l,j,i}} \tag{4.70}$$

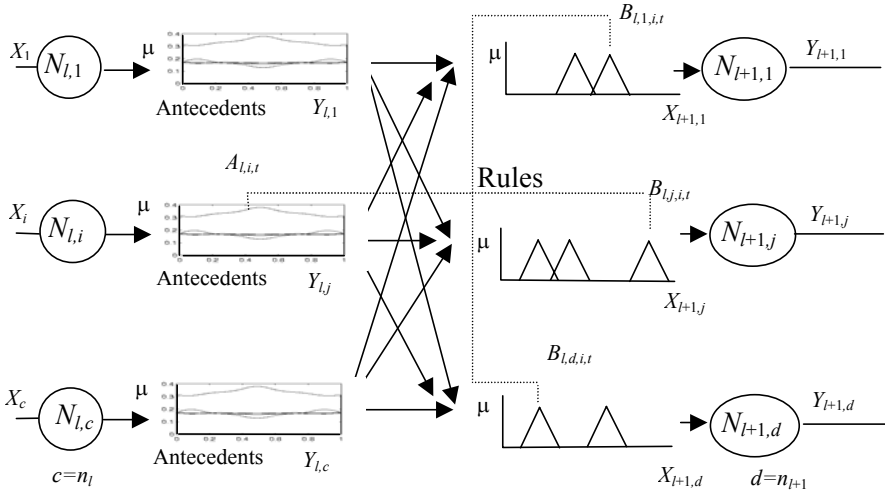


Fig. 4.12. Reduced NGNN

where "*r*" denotes "reduced", and $\forall i : m_{l,i}^r \le m_{l,i}$, and the maximum error of the reduction can be computed from the discarded singular values [S73], [S83]. The reduced form can be represented as a neural network in Fig. 4.12.

Proof:

Let $\underline{\mu}_i$ and $\underline{\underline{M}}_{l,i}$ be defined as:

$$\underline{\mu}_i(y_{l,i}) = \left\lfloor \mu_{l,i,1}(y_{l,1}) \quad \cdots \quad \mu_{l,i,m_{l,i}}(y_{l,i}) \right\rfloor$$

$$\underline{\underline{O}}_{l,i} = \begin{vmatrix} b_{l,1,i,1} & \cdots & b_{l,n_{l+1},i1} \\ \vdots & \ddots & \vdots \\ b_{l,1,i,m_{l,i}} & \cdots & b_{l,n_{l+1},i,m_{l,i}} \end{vmatrix}, \ \underline{\underline{P}}_{l,i} = \begin{vmatrix} s_{l,1,i,1} & \cdots & s_{l,n_{l+1},i1} \\ \vdots & \ddots & \vdots \\ s_{l,1,i,m_{l,i}} & \cdots & s_{l,n_{l+1},i,m_{l,i}} \end{vmatrix}, \ \underline{\underline{M}}_{l,i} = \left| \underline{\underline{O}}_{l,i} \underline{\underline{P}}_{l,i} \right| .$$

With this notations

$$\left\lfloor N_{l,1,i} \quad \cdots \quad N_{l,n_{l+1},i} \right\rfloor = \underline{\mu}_i(y_{l,i}) \underline{\underline{O}}_{l,i}, \text{ and } \left\lfloor D_{l,1,i} \quad \cdots \quad D_{l,n_{l+1},i} \right\rfloor = \underline{\mu}_i(y_{l,i}) \underline{\underline{P}}_{l,i} . \tag{4.71}$$

Applying the above described singular value-based reduction to the $\underline{\underline{M}}_{l,i}$ matrices one will get

$$\underline{\underline{M}}_{l,i} \approx \underline{\underline{T}}_{l,i} \underline{\underline{U}}_{l,i} \underline{\underline{V}}_{l,i} = \underline{\underline{T}}_{l,i} \underline{\underline{M}}'_{l,i} = \underline{\underline{T}}_{l,i} \left| \underline{\underline{O}}'_{l,i} \underline{\underline{P}}'_{l,i} \right| \tag{4.72}$$

The error of this reduction ($\underline{\underline{M}}_{l,i} - \underline{\underline{T}}_{l,i} \underline{\underline{M}}'_{l,i}$) will be the sum of the discarded singular values, E_i. The output of the reduced neural network is

$$y'_{l+1,j} = \sum_{i=1}^{n_{l+1}} \frac{\sum_{t=1}^{m_{l,i}^r} \mu_{l,i,t}^r(y_{l,i}) b'_{l,z,i,t}}{\sum_{t=1}^{m_{l,i}^r} \mu_{l,i,t}^r(y_{l,i}) s'_{l,z,i,t}} = \sum_{i=1}^{n_{l+1}} \frac{N'_{l,j,i}}{D'_{l,j,i}}, \tag{4.73}$$

where

$$\left[N'_{l,1,i} \quad \cdots \quad N'_{l,n_{l+1},i}\right] = \underline{\mu}^{r}{}_{i}(y_{l,i})\underline{\underline{O}}'_{l,i}, \text{ and } \left[D'_{l,1,i} \quad \cdots \quad D'_{l,n_{l+1},i}\right] = \underline{\mu}^{r}{}_{i}(y_{l,i})\underline{\underline{P}}'_{l,i}.$$

The antecedent fuzzy sets of the reduced network can be obtained as

$$\underline{\mu}^{r}{}_{i}(y_{l,i}) = \left[\mu^{r}{}_{l,i,1}(y_{li1}) \quad \cdots \quad \mu^{r}{}_{l,i,m_{l,i}}(y_{l,i})\right] = \underline{\mu}_{i}(y_{l,i})\underline{\underline{T}}_{l,i} \tag{4.74}$$

The reduced consequent fuzzy sets of the reduced network will have centers of gravity of $d'_{l,j,i,t} = b'_{l,j,i,t} / s'_{l,j,i,t}$ and areas of $s'_{l,j,i,t}$, where $b'_{l,j,i,t}$ and $s'_{l,j,i,t}$ are the elements of $\underline{\underline{O}}'_{l,i}$ and $\underline{\underline{P}}'_{l,i}$, accordingly.
The errors of the numerators can be determined as:

$$\left|N_{l,j,i} - N'_{l,j,i}\right| = \left|\underline{\mu}_{i}(y_{l,i})\underline{\underline{O}}_{l,i} - \underline{\mu}^{r}{}_{i}(y_{l,i})\underline{\underline{O}}'_{l,i}\right| = \left|\underline{\mu}_{i}(y_{l,i})(\underline{\underline{O}}_{l,i} - \underline{\underline{T}}_{l,i}\underline{\underline{O}}'_{l,i})\right| \le \left|\sum_{i=1}^{n_l}\mu_{l,i,t}(y_{l,i})E_i\right| = E_i \tag{4.75}$$

because the original antecedent fuzzy sets are in Ruspini-partition.
The error of the denominator can be estimated in the same way, and it is also equal or less, then E_i.
Let

$$\Delta N_{l,j,i} = N'_{l,j,i} - N_{l,j,i} \text{ and } \Delta D_{l,j,i} = D'_{l,j,i} - D_{l,j,i}.$$

The overall error of $y_{l+1,j}$ is

$$\begin{aligned}\left|y'_{l+1,j} - y_{l+1,j}\right| &= \left|\sum_{i=1}^{n_l}\frac{N'_{l,j,i}}{D'_{l,j,i}} - \sum_{i=1}^{n_l}\frac{N_{l,j,i}}{D_{l,j,i}}\right| \le \sum_{i=1}^{n_l}\left|\frac{N'_{l,j,i}}{D'_{l,j,i}} - \frac{N_{l,j,i}}{D_{l,j,i}}\right| = \\ &= \sum_{i=1}^{n_l}\left|\frac{D_{l,j,i}(N_{l,j,i} + \Delta N_{l,j,i}) - N_{l,j,i}(D_{l,j,i} + \Delta D_{l,j,i})}{D_{l,j,i}D'_{l,j,i}}\right| = \sum_{i=1}^{n_l}\left|\frac{\Delta N_{l,j,i}}{D'_{l,j,i}} - \frac{N_{l,j,i}}{D_{l,j,i}}\frac{\Delta D_{l,j,i}}{D'_{l,j,i}}\right|\end{aligned} \tag{4.76}$$

Because $\left|\Delta N_{l,j,i}\right| \le E_i$ and $\left|\Delta D_{l,j,i}\right| \le E_i$:

$$\left|y'_{l+1,j} - y_{l+1,j}\right| \le \sum_{i=1}^{n_l}\left(\left|\frac{E_i}{D'_{l,j,i}}\right| + \left|\frac{N_{l,j,i}}{D_{l,j,i}}\right|\left|\frac{E_i}{D'_{l,j,i}}\right|\right) = \sum_{i=1}^{n_l}\left(\left|\frac{E_i}{D'_{l,j,i}}\right|\left(1 + \left|\frac{N_{l,j,i}}{D_{l,j,i}}\right|\right)\right) \tag{4.77}$$

Since the original antecedent fuzzy sets were in Ruspini-partition, the maximum of $\frac{N_{l,j,i}}{D_{l,j,i}}$ will be in one of the grid-points (points, where one of the original $\mu_{l,i,t}(y_{l,i}) = 1$ for every i):

$$\left|\frac{N_{l,j,i}}{D_{l,j,i}}\right| \le \max_{t}(d_{l,i,j,t}). \tag{4.78}$$

If $D'_{l,j,i} \ge 0$, i.e. $E_i \le \min_{t}(s_{l,j,i,t})$, then the error is

$$\left|y'_{l+1,j}-y_{l+1,j}\right| \le \sum_{i=1}^{n_l}\left(\left|\frac{E_i}{\min_t(s_{l,j,i,t})-E_i}\right|\left(1+\max_t(d_{l,j,i,t})\right)\right), \tag{4.79}$$

where E_i is the sum of discarded singular values, $s_{l,j,i,t}$ are the areas and $d_{l,j,i,t}$ are the centers of gravity of the original consequent fuzzy sets.

The effectiveness of the complexity reduction may be improved, if the fuzzy logic form can be lost in the resulted network.

Theorem 4.7. (4.69) can always be written into the form of

$$y'_{l+1,j}=\sum_{i=1}^{n_l}\frac{\sum_{z=1}^{n^r_{l,i}} a_{l,i,z}\sum_{t=1}^{m^r_{l,i}}\mu^r{}_{l,i,t}(y_{l,i})b''_{l,z,i,t}}{\sum_{z=1}^{\bar{n}^r_{l,i}}\bar{a}_{l,i,z}\sum_{t=1}^{\bar{m}^r_{l,i}}\bar{\mu}^r{}_{l,i,t}(y_{l,i})s''_{l,z,i,t}}=\sum_{i=1}^{n_l}\frac{N''_{l,j,i}}{D''_{l,j,i}}, \tag{4.80}$$

where „r" denotes „reduced", „ $\bar{}$ " denotes values in the denominator corresponding to values in the numerator, further $n^r_{l+1},\bar{n}^r_{l+1}\le n_{l+1}$ and $\forall i: m^r_{l,i},\bar{m}^r_{l,i}\le m_{l,i}$, and the maximum error of the reduction can be computed from the discarded singular values [S73], [S83].

Proof:

The numerator and denominator parts can be reduced separately. In the following, first the reduction of the numerator is shown and the error bound is given. The reduction of the denominator can be carried out in the same way, and the error-bound can be determined similarly. The reduction can be made in two steps. First, the $a_{l,j,z}$ values are computed, while in the second the new $\mu^r{}_{l,i,t}(y_{l,i})$ functions are determined.

Step 1. *Determination of the* $a_{l,j,z}$ *values*

Let $\underline{\mu}_i$ and $\underline{\underline{S}}_l$ be defined as:

$$\underline{\mu}_i=\left[\mu_{l,i,1}(y_{l,i}) \quad \cdots \quad \mu_{l,i,m_{l,i}}(y_{l,i})\right]^T$$

$$\underline{\underline{S}}_l=\left|\underline{\underline{H}}_{l,1} \quad \cdots \quad \underline{\underline{H}}_{l,n_l}\right| \qquad \underline{\underline{H}}_{l,i}=\begin{vmatrix} b_{l,1,i,1} & \cdots & b_{l,1,i,m_{l,i}} \\ \vdots & \ddots & \vdots \\ b_{l,n_{l+1},i,1} & \cdots & b_{l,n_{l+1},i,m_{l,i}}\end{vmatrix}$$

where the $\underline{\mu}_i$ vectors (lengths: $m_{l,i}$) contain the values of the membership functions in a given $(y_{l,1},...,y_{l,n_l})$ point, and $\underline{\underline{S}}_l$ contains all the $b_{l,j,i,t}$ values. The size of $\underline{\underline{S}}_l$ is $n_{l+1}\times\sum_{i=1}^{n_l}m_{l,i}$.

With this notation:

$$\left[N_{l,1,i} \quad \cdots \quad N_{l,n_{l+1},i}\right]=\underline{\underline{H}}_{l,i}\underline{\mu}_i(y_{l,i}). \tag{4.81}$$

Applying the above described singular value-based reduction to the $\underline{\underline{S}}_l$ matrix one gets:

$$\underline{\underline{S}}_l\approx\underline{\underline{A}}_l\underline{\underline{D}}_l\underline{\underline{V}}_l=\underline{\underline{A}}_l\underline{\underline{S}}'_l=\underline{\underline{A}}_l\left|\underline{\underline{H}}'_{l,1} \quad \cdots \quad \underline{\underline{H}}'_{l,n_l}\right|, \tag{4.82}$$

The error of the reduction is the sum of the discarded singular values, $E_{N,1}$:

$$\left| b_{l,j,i,t} - \sum_{z=1}^{n^r_{l+1}} a_{l,j,z} b'_{l,z,i,t} \right| \le E_{N,1}, \tag{4.83}$$

where $b'_{l,j,i,t}$ are the elements of $\underline{\underline{S}}'_l$. The numerators after the first step are:

$$\left[N'_{l,1,i} \quad \cdots \quad N'_{l,n_{l+1},i} \right] = \underline{\underline{A}}_l \underline{\underline{H}}'_{l,i} \underline{\mu}_i(y_{l,i}). \tag{4.84}$$

The difference between the current and the original numerators, using that the original antecedent fuzzy sets are in Ruspini-partition (i.e. $sum(\underline{\mu}_{1,i}(y_{l,i})) = 1$ for every $y_{l,i}$):

$$\left| \underline{\underline{A}}_l \underline{\underline{H}}'_{l,i} \underline{\mu}_{1,i}(y_{l,i}) - \underline{\underline{H}}_{l,i} \underline{\mu}_{1,i}(y_{l,i}) \right| \le \left| \left(\underline{\underline{A}}_l \underline{\underline{H}}'_{l,i} - \underline{\underline{H}}_{l,i} \right) \underline{\mu}_{1,i}(y_{l,i}) \right| \le E_{N,1} \tag{4.85}$$

Step 2. *Determination of the new membership functions*
Applying the singular value-based reduction to the $\underline{\underline{H}}'^T_{l,i}$ matrices one gets:

$$\underline{\underline{H}}'^T_{l,i} \approx \underline{\underline{T}}_{l,i} \underline{\underline{D}}_{l,i} \underline{\underline{V}}_{l,i} = \underline{\underline{T}}_{l,i} \underline{\underline{H}}''^T_{l,i}. \tag{4.86}$$

The error of this reduction ($\underline{\underline{H}}'^T_{l,i} - \underline{\underline{T}}_{l,i} \underline{\underline{H}}''^T_{l,i}$) will be the sum of the discarded singular values, $E_{N,2,i}$ (the error also depends on *i*). The numerators after the reduction is:

$$\left[N''_{l,1,i} \quad \cdots \quad N''_{l,n_{l+1},i} \right]^T = \underline{\mu}_i^T(y_{l,i}) \underline{\underline{T}}_{l,i} \underline{\underline{H}}''^T_{l,i} \underline{\underline{A}}_l^T. \tag{4.87}$$

The new $\mu^r{}_{l,i,t}(y_{l,i})$ functions can be obtained as

$$\underline{\mu}^r{}_i(y_{l,i}) = \left[\mu^r{}_{l,i,1}(y_{li1}) \quad \cdots \quad \mu^r{}_{l,i,m_{l,i}}(y_{l,i}) \right] = \underline{\mu}_i^T(y_{l,i}) \underline{\underline{T}}_{l,i}, \tag{4.88}$$

where the $b''_{l,j,i,t}$ values are the elements of $\underline{\underline{H}}''_{l,i}$.
The error of the numerator in the second step:

$$\left| \left[N'_{l,1,i} \quad \cdots \quad N'_{l,n_{l+1},i} \right]^T - \left[N''_{l,1,i} \quad \cdots \quad N''_{l,n_{l+1},i} \right]^T \right| = \left| \underline{\mu}_i^T(y_{l,i}) \underline{\underline{T}}_{l,i} \underline{\underline{H}}''^T_{l,i} \underline{\underline{A}}_l^T - \underline{\mu}_i^T(y_{l,i}) \underline{\underline{H}}'^T_{l,i} \underline{\underline{A}}_l^T \right| =$$

$$= \left| \underline{\mu}_i^T(y_{l,i}) \left(\underline{\underline{T}}_{l,i} \underline{\underline{H}}''^T_{l,i} - \underline{\underline{H}}'^T_{l,i} \right) \underline{\underline{A}}_l^T \right| \le E_{N,2,i} \tag{4.89}$$

For the last step, the sum of the rows of $\underline{\underline{A}}_l$ must be less or equal than one. This can be ensured by a further transformation in the first step:

$$\underline{\underline{S}}_l \approx \underline{\underline{A}}'_l \underline{\underline{D}}'_l \underline{\underline{V}}'_l = \left(\underline{\underline{A}}'_l \underline{\underline{K}}_l \right) \underline{\underline{K}}_l^{-1} \underline{\underline{D}}'_l \underline{\underline{V}}'_l = \underline{\underline{A}}_l \underline{\underline{S}}'_l,$$

$$\underline{\underline{K}}_l = \left\langle 1 / \max_j \{ \sum_z a_{l,j,z} \} \right\rangle, \text{ and } \underline{\underline{S}}'_l = \underline{\underline{K}}_l^{-1} \underline{\underline{D}}'_l \underline{\underline{V}}'_l \tag{4.90}$$

Thus, the error of the numerator will be altogether the sum of the errors originating from the first and second steps (from (4.85) and (4.89)):

$$E_{N,i} = \left| N_{l,j,i} - N''_{l,j,i} \right| \le E_{N,1} + E_{N,2,i} . \tag{4.91}$$

Similarly can be obtained $E_{D,i}$, the error of the denominator, as well.

Let

$$\Delta N'_{l,j,i} = N''_{l,j,i} - N_{l,j,i} \text{ and } \Delta D'_{l,j,i} = D''_{l,j,i} - D_{l,j,i} .$$

The overall error of $y_{l+1,j}$ is (using (4.76))

$$\left| y'_{l+1,j} - y_{l+1,j} \right| = \left| \sum_{i=1}^{n_l} \frac{N''_{l,j,i}}{D''_{l,j,i}} - \sum_{i=1}^{n_l} \frac{N_{l,j,i}}{D_{l,j,i}} \right| \le \sum_{i=1}^{n_l} \left| \frac{\Delta N'_{l,j,i}}{D'_{l,j,i}} - \frac{N_{l,j,i}}{D_{l,j,i}} \frac{\Delta D'_{l,j,i}}{D''_{l,j,i}} \right| \tag{4.92}$$

Because $\left| \Delta N'_{l,j,i} \right| \le E_{N,i}$ and $\left| \Delta D'_{l,j,i} \right| \le E_{D,i}$:

$$\left| y'_{l+1,j} - y_{l+1,j} \right| \le \sum_{i=1}^{n_l} \left(\left| \frac{E_{N,i}}{D'_{l,j,i}} \right| + \left| \frac{N_{l,j,i}}{D_{l,j,i}} \right| \left| \frac{E_{D,i}}{D'_{l,j,i}} \right| \right) \tag{4.93}$$

If $D''_{l,j,i} \ge 0$, i.e., $E_{D,i} \le \min_t (s_{l,j,i,t})$ then the error is, using (4.78):

$$\left| y'_{l+1,j} - y_{l+1,j} \right| \le \sum_{i=1}^{n_l} \left(\left| \frac{1}{\min_t (s_{l,j,i,t}) - E_{D,i}} \right| \left(E_{N,i} + E_{D,i} \max_t (d_{l,j,i,t}) \right) \right), \tag{4.94}$$

where $s_{l,j,i,t}$ are the areas and $d_{l,j,i,t}$ are the centers of gravity of the original consequent fuzzy sets, and $E_{N,i}$ and $E_{D,i}$ are the sums of the discarded singular values during the reduction of the numerators and denominators, accordingly.

4.5 Transformation of PSGS fuzzy systems to iterative models

In case of fuzzy systems, the needed time/resources can be flexibly changed by the evaluation of a subset of the rules. However, since the significance of the rules highly depends on the actual inputs, it has been hard to tell until recently, which rules could be omitted if we wanted to ensure a given accuracy. Nor could we ensure that the result of the processing would be the available most accurate one. For this, we should be able to give the order of the significance of the rules, for arbitrary input and then evaluate them one by one until the available time is over. Unfortunately, the rules which are formulated based on either expert knowledge or some other source (analytical model, sampling, etc.) are not of this type.

In the following, an SVD based transformation method will be described, by which PSGS fuzzy systems can be transformed into a new form. The new form can be evaluated iterative-type, rule by rule with known error bound in every step (see also [S27]) and holds the optimum (minimum error) criteria. This makes the use of fuzzy systems possible in anytime applications, without constructing a modular architecture.

4.5.1. SVD based transformation of PSGS fuzzy systems to iterative models

Theorem 4.8. PSGS fuzzy system can always be transformed to a form, which can be evaluated iterative type, ensuring the assumption of monotonously decreasing error of the estimation ([S4], [S27], [S88]).

Proof:

Consider a PSGS fuzzy rule base with N inputs, containing m_1, m_2, ... , m_N antecedent fuzzy sets, respectively. The antecedent fuzzy sets are in Ruspini-partition, the consequent fuzzy sets are singletons. Thus, the rule base contains $m_1*m_2*...*m_N$ rules as

$$R_{i_1,i_2,...,i_N}: \text{If } x_1 \text{ is } X_{1,i_1} \text{ and } x_2 \text{ is } X_{2,i_2} \text{ and } ... \text{ and } x_N \text{ is } X_{N,i_N} \text{ then } y = y_{i_1,i_2,...,i_N}, 1 \le i_j \le m_j .$$

The fuzzification method is singleton, and during the inference product T-norm and sum S-norms are used. The result of the fuzzy inference in case of the $(x_1^*, x_2^*, ..., x_N^*)$ input values

$$y^* = \sum_{i_1,i_2,...,i_N} y_{i_1,i_2,...,i_N} \mu_{1,i_1}(x_1^*)\mu_{2,i_2}(x_2^*)...\mu_{N,i_N}(x_N^*)$$

Let choose H $(1 \le H \le N-1)$ so that $n_1=m_1*...*m_H$ and $n_2=m_{H+1}*...*m_N$ be as close to each other, as possible – the efficiency of the transformation is usually better, if the dimensions are close to each other. By this

$$y^* = \left|\mu_{1,1}(x_1^*)*...*\mu_{H,1}(x_H^*), \quad \cdots \quad ,\mu_{1,n_1}(x_1^*)*...*\mu_{H,n_H}(x_H^*)\right| \begin{bmatrix} y_{1,...,1} & \cdots & y_{1,...,1,n_{H+1},...,n_N} \\ \vdots & \ddots & \vdots \\ y_{n_1,...,n_H,1,...,1} & \cdots & y_{n_1,...,n_N} \end{bmatrix} *$$

$$* \begin{vmatrix} \mu_{H+1,1}(x_{H+1}^*)*...*\mu_{N,1}(x_N^*) \\ \vdots \\ \mu_{H+1,n_{H+1}}(x_{H+1}^*)*...*\mu_{N,n_N}(x_N^*) \end{vmatrix} = \underline{\mu}_1(x_1^*,...,x_H^*)\underline{\underline{Y}}\underline{\mu}_2(x_{H+1}^*,...,x_N^*) \tag{4.95}$$

where the length of the vectors $\underline{\mu}_1(x_1^*,...,x_H^*)$ and $\underline{\mu}_2(x_{H+1}^*,...,x_N^*)$ are n_1 and n_2, respectively. The $(i_1+(i_2-1)*n_1+...+(i_H-1)*n_1*...*n_{H-1})$ -th element of $\underline{\mu}_1(x_1^*,...,x_H^*)$ is $\mu_{1,i_1}(x_1^*)*...*\mu_{H,i_H}(x_H^*)$ and the $(i_{H+1}+(i_{H+2}-1)*n_{H+1}+...+(i_N-1)*n_{H+1}*...*n_{N-1})$ -th element of $\underline{\mu}_2(x_{H+1}^*,...,x_N^*)$ is $\mu_{H+1,i_{H+1}}(x_{H+1}^*)*...*\mu_{N,i_N}(x_N^*)$.

The size of matrix $\underline{\underline{Y}}$ is $n_1 \times n_2 = (m_1*...*m_H)\times(m_{H+1}*...*m_N)$ and the element $y_{i_1,i_2,...,i_N}$ is in the position

$(i_1+(i_2-1)*n_1+...+(i_H-1)*n_1*...*n_H, i_{H+1}+(i_{H+2}-1)*n_{H+1}+...+(i_N-1)*n_{H+1}*...*n_{N-1})$ A ccording to (1), matrix $\underline{\underline{Y}}$ can be decomposed as

$$\underline{\underline{Y}}_{(n_1\times n_2)} = \underline{\underline{A}}_{1,(n_1\times n_1)}\underline{\underline{B}}_{(n_1\times n_2)}\underline{\underline{A}}^T_{2,(n_2\times n_2)}$$

where $\underline{\underline{A}}_k$ (k=1,2) are orthogonal matrices, and the diagonal matrix $\underline{\underline{B}}$ contains the λ_i singular values of $\underline{\underline{Y}}$ in decreasing order. The maximum number of the nonzero singular values equals $n_{SVD} = \min(n_1, n_2)$. Let new vectors be defined as

$$\underline{\mu}'_1(x_1^*,...,x_H^*) = \underline{\mu}_1(x_1^*,...,x_H^*) * \underline{\underline{A}}_1 \text{ and } \quad \underline{\mu}'_2(x_{H+1}^*,...,x_N^*) = \underline{\underline{A}}_2^T * \underline{\mu}_2(x_{H+1}^*,...,x_N^*) .$$

Thus:

$$y^* = \underline{\mu}'_1(x_1^*,...,x_H^*)\underline{\underline{B}}\underline{\mu}'_2(x_{H+1}^*,...,x_N^*) = \sum_{k=1}^{n_{SVD}} \lambda_k * \underline{\mu}'_1(x_1^*,...,x_H^*)[k] * \underline{\mu}'_2(x_{H+1}^*,...,x_N^*)[k], \tag{4.96}$$

where [k] denotes the k-th element of the given vector.

4.5.2. Error estimation

The transformed form of a PSGS fuzzy system ((4.96)) can be evaluated gradually, by computing and summing the terms one after the other. The advantage of this form is that the error can be estimated at any point of the evaluation without considering the actual inputs contrary to the original form where the error usually highly depends on the input values.

Theorem 4.9. The error of the estimation after evaluating $n_{SVD} - m$ terms can not exceed the sum of the discarded singular values of the transformed form ([S4], [S27], [S88]).

Proof:

After summing up $n_{SVD} - m$ terms, the output is

$$y_m^* = \sum_{k=1}^{n_{SVD}-m} \lambda_k * \underline{\mu}'_1(x_1^*,...,x_H^*)[k] * \underline{\mu}'_2(x_{H+1}^*,...,x_N^*)[k] = \underline{\mu}_1(x_1^*,...,x_H^*)\underline{\underline{A}}^r_{1,m}\underline{\underline{B}}^r_m\underline{\underline{A}}^{rT}_{2,m}\underline{\mu}_2(x_{H+1}^*,...,x_N^*) \tag{4.97}$$

where

$$\underline{\underline{A}}_j = \left|\underline{\underline{A}}^r_{j,m(n_j\times m)} \quad \underline{\underline{A}}^d_{j,m(n_j\times(n_j-m))}\right| \text{ and } \underline{\underline{B}} = \begin{vmatrix} \underline{\underline{B}}^r_{m,(m\times m)} & 0 \\ 0 & \underline{\underline{B}}^d_{((n_1-m)\times(n_2-m))} \end{vmatrix}.$$

Let $E_{SVD,m}$ be the upper bound for the error of the SVD based matrix reduction after discarding m singular values

$$\left|\underline{\underline{Y}} - \underline{\underline{Y}}'\right| = \left|\underline{\underline{Y}} - \underline{\underline{A}}^r_{1,m}\underline{\underline{B}}^r_m\underline{\underline{A}}^{rT}_{2,m}\right| \le E_{SVD,m} \quad . \tag{4.98}$$

Thus, the error of the output after summarizing $n_{SVD} - m$ terms

$$\left|y^* - y_m^*\right| = \left|\underline{\mu}_1(x_1^*,...,x_H^*)(\underline{\underline{Y}} - \underline{\underline{A}}^r_{1,m}\underline{\underline{B}}^r_m\underline{\underline{A}}^{rT}_{2,m})\underline{\mu}_2(x_{H+1}^*,...,x_N^*)\right| \le$$

$$\le \left|\underline{\mu}_1(x_1^*,...,x_H^*)\begin{bmatrix} E_{SVD,m} & \cdots & E_{SVD,m} \\ \vdots & \ddots & \vdots \\ E_{SVD,m} & \cdots & E_{SVD,m} \end{bmatrix}\underline{\mu}_2(x_{H+1}^*,...,x_N^*)\right| \le$$

$$\le \left|E_{SVD,m}\sum_{i_1,i_2,...,i_N}\mu_{1,i_1}(x_1^*)\mu_{2,i_2}(x_2^*)...\mu_{N,i_N}(x_N^*)\right| \le E_{SVD,m}, \tag{4.99}$$

(Here we have used the fact that the original antecedent fuzzy sets are in Ruspini partition.)
(4.99) implies two important things. First, the terms corresponding to zero singular values can be left out without degrading the accuracy, i.e. an exact complexity reduction can be made.
Secondly, if because of some temporal time/resource shortage further non-zero terms are left out, the error will still be easily estimable, according to (4.99). It is always equal to or less than the sum of the discarded singular values at the given point, thus the accuracy of the computations is monotonously increasing by adding more and more terms.
Moreover, since the SVD algorithm sets the singular values in the order of magnitude (starting with the highest one in matrix $\underline{\underline{B}}$), the most significant terms are added first during the evaluation, ensuring that the error decreases as fast as possible. Thus, if the available time is not enough to evaluate all the terms corresponding to nonzero singular values and some of them must be left out, it will cause the smallest possible error.

It is worth mentioning, that the effectiveness of the evaluation method depends on the properties of the original fuzzy system. If the singular values are nearly equal, which means that the terms contribute to the result in nearly the same degree, non-exact complexity reduction may cause considerable, possibly not acceptable error with a magnitude more or less proportional to the number of neglected terms. Even if that is the case, the results may help in making qualitative decisions.
On the contrary, if the singular values are highly different (in the practice, this is the more common case) then the first terms have much higher contribution to the result then the last ones. The error will decrease faster and a good approximation can be generated by using only those terms which have significant weights and the negligible components can be omitted.

4.6 Anytime Modeling: Complexity Reduction and Improving the Approximation

With the help of the SVD-based reduction not only the redundancy of the rule-bases of fuzzy systems (or neural nets) can be removed, but further reduction can also be obtained, considering the allowable error. This latter can be done adaptively according to the temporal conditions, thereby offering a way to use soft computational, fuzzy and generalized neural network based models in anytime systems.

Theorem 4.10. PSG fuzzy and GNN models can be operated in anytime mode ([S1], [S5], [S4], [S12], [S3], [S18], [S21], [S22], [S72], [S26], [S93], [S94], [S95], [S75], [S77], [S82], [S86], [S87], [S91], [S95], [S102]).

Proof: Methods 4.3-4.6, discussed below, ensure Theorem 4.10.

The method also offers a way for improving the model if later on we get into possession of new information (approximation points) or more resources. An algorithm can be suggested, which finds the common minimal implementation space of the compressed original and the new approximation points, thus the complexity will not explode as we include new information into the model.
These two techniques, non-exact complexity reduction and the improvement of the approximation accuracy, ensure that we can always cope with the temporarily available (finite) time/resources and find the balance between accuracy and complexity.

4.6.1 Reducing the complexity of the model

The steps of using anytime models for coping with the temporarily insufficient resources and/or computational time, are the followings:

Method 4.3. ([S1], [S5], [S18], [S4], [S27], [S72], [S77])

Step 1. First a practically "accurate" fuzzy or NN system is to be constructed. For the determination of the rule-base, expert knowledge can be used. Further improvement can be obtained by utilizing training data and some learning algorithm.

Step 2. In the second step, by applying the above described (HO)SVD based complexity reduction algorithm, a reduced but "accurate" model can be generated. (In this step only the redundancy is removed.)

Step 3. The SVD based model can be used in anytime systems either by applying the iterative transformation algorithm described in Section 4.5 (PSGS fuzzy systems) or in the more general frame of the modular architecture presented in Section 4.2.

In the first case, the transformation can be performed off-line, before the anytime operation starts (i.e. it does not cause any additional computational load on the system) and the model evaluation can be executed without knowing about the available amount of time. The newest output will always correspond to the in the given circumstances obtainable best results.
In the second case, based on the SVD transformed model of the system, further non-exact variations of the rule-bases of the model must be constructed. These models will differ in their accuracy and complexity. For anytime use, an alternative rule-base is characterized by its complexity and its error that can be estimated by the sum of the discarded singular values.
The different rule-bases form the different units realizing a given module (Fig. 4.1).

Step 4. During the operation, an intelligent expert system, monitoring the actual state of the supervised system, can adaptively determine and change for the units (rule base models) to be applied according to the available computing time and resources at the moment. These considerations need additional computational time/resources (further reducing the resources). On the other hand, because the inference algorithm within the models of a certain module is the same, only the rule-bases - a kind of parameter set - must be changed resulting in advantageous dynamic behavior.

It is worth mentioning, that the SVD based reduction finds the optimum, i.e., minimum number of parameters which is needed to describe the system.
One can find more details about the intelligent anytime monitor and the algorithmic optimization of the evaluations of the model-chain in Section 4.2.3 and in [S24], [S97], [36].

4.6.2 Improving the approximation of the model ([S72], [S92], [S93], [S21], [S75], [S86])

The main goal of anytime systems is to keep on the continuous, near optimal operation through finding a balance between the quality of the processing and the available resources. The complexity of the model can be tuned both by evaluating only a degraded model (decreasing the granulation), and both by improving the existing model (increasing the granulation) in the knowledge of new information. In our case, this latter means the improvement of the density of the approximation points. Here a very important aim is not to let to explode the complexity of the compressed model when the approximation is extended with new points. Thus, if approximation A is extended to B with a new set of approximation points and basis, then the question is how to transform A^r to B^r directly based on the new information without decompressing A^r, where A^r and B^r are the SVD based reduced forms of A and B. In the followings an algorithm is summarized for the complexity compressed increase of such approximations.
To enlighten more the problem, let us show a simple example. Assume that we deal with the approximation of function $F(x_1,x_2)$ (see Fig. 4.13). For simplicity, assume that the applied approximation A is a bi-linear approximation based on the sampling of $F(x_1,x_2)$ over a rectangular grid (Fig. 4.14), so, the basis are formed of triangular fuzzy sets (or first order B-spline functions). After applying the SVD based reduction technique, the minimal dimensionality of the basis functions is defined. In Fig. 4.15, as the minimum basis, two basis functions are shown on each dimension instead of the original three as depicted in Fig. 4.14. (Note that after reduction, the grid-net of the approximation points disappears. The approximation points can be localized by normalization.)
Let us suppose that at a certain stage, further points are sampled (Fig. 4.16) in order to increase the density of the approximation points in dimension X_1, hence, to improve approximation A to achieve approximation B. The new points can easily be added to approximation A shown in Fig. 4.14 to yield approximation B with an extended basis, as is shown in Fig. 4.17. Usually, however, once reduced approximation A^r is found then the new points should directly be added to A^r (where there is no localized approximation point) to generate a reduced approximation B^r (see Fig. 4.18). Here again, as an illustration, two basis are obtained in each dimension, hence the calculation complexity of A^r and B^r are the same, but the approximation is improved.

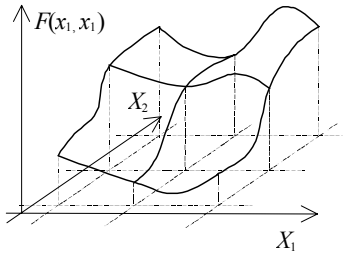


Fig. 4.13 Sampling $F(x_1, x_2)$ over a rectangular grid

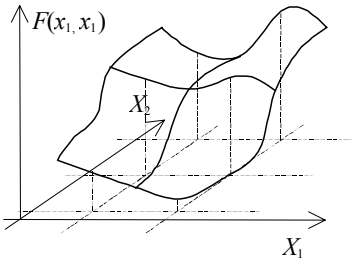


Fig. 4.16 Sampling further approximation points

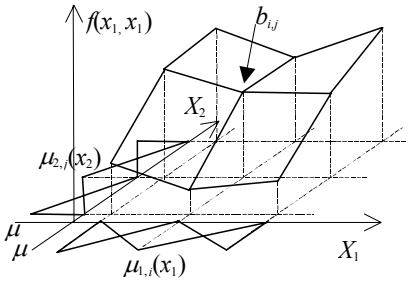


Fig. 4.14 Bi-linear approximation A of function $F(x_1, x_2)$

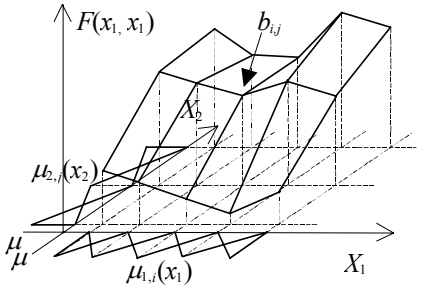


Fig. 4.17 Approximation B

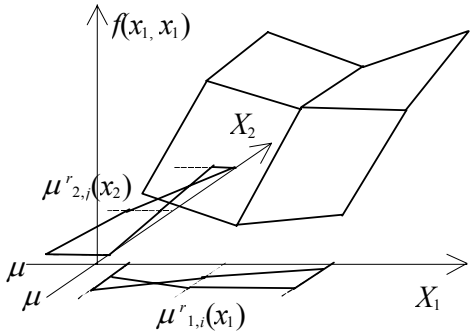


Fig.4.15 Approximation A^r, which is the reduced form of approximation A

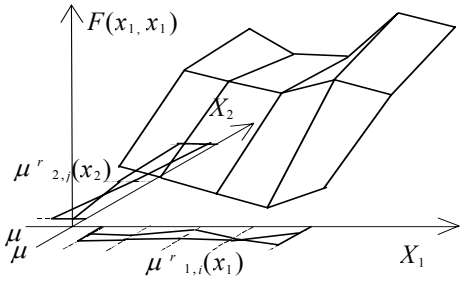


Fig. 4.18 Reduced approximation B^r

In more general, the crucial point here is to inject new information, given in the original form, into the compressed one. If the dimensionality of B^r is larger than A^r then the new points and basis lead to the expansion of the basis' dimensionality of the reduced form A^r. On the other hand, if the new points and basis have no new information on the dimensionality of the basis then they are swallowed in the reduced form without the expansion of the dimensionality, however the approximation is improved. Thus, the approximation can get better with new points without increasing the calculation complexity. This implies a practical question, namely: how to apply those extra points taken from a large sampled set to be embedded, which have no new information on the dimensionality of the basis, but carry new information on the approximation? Again, the main difficulty is that the extra points and bases are given in the form of A and they have to be embedded in A^r without decompressing it.

Method 4.4. For the fitting of two approximations into a common basis system, we will use the transformation of the rational general form of PSGS and Takagi-Sugeno-Kang fuzzy systems. The

rational general form ([37]) means that these systems can be represented by a rational fraction function

$$y = \frac{\sum_{j_1=1}^{e_1} \cdots \sum_{j_n=1}^{e_n} \prod_{i=1}^{n} \mu_{i,j_i}(x_i) f_{j_1,\cdots,j_n}(x_1,\cdots,x_n)}{\sum_{j_1=1}^{e_1} \cdots \sum_{j_n=1}^{e_n} \prod_{i=1}^{n} \mu_{i,j_i}(x_i) w_{j_1,\cdots,j_n}} \tag{4.100}$$

where

$$f_{i_1,\cdots,i_n}(x_1,\cdots,x_n) = \sum_{t=1}^{m} b_{i_1,\cdots,i_n,t} \phi(x_1,\cdots,x_n).$$

It can be proved (see e.g. [S57], [24]) that (19) can always be transformed into the form of

$$y = \frac{\sum_{j_1=1}^{e_1^r} \cdots \sum_{j_n=1}^{e_n^r} \prod_{i=1}^{n} \mu_{i,j_i}^r(x_i) f_{j_1,\cdots,j_n}^r(x_1,\cdots,x_n)}{\sum_{j_1=1}^{e_1^r} \cdots \sum_{j_n=1}^{e_n^r} \prod_{i=1}^{n} \mu_{i,j_i}^r(x_i) w_{j_1,\cdots,j_n}^r} \tag{4.101}$$

where

$f_{i_1,\cdots,i_n}^r(x_1,\cdots,x_n) = \sum_{t=1}^{m} b_{i_1,\cdots,i_n,t}^r \phi(x_1,\cdots,x_n)$ and $\forall i : e_i^r \le e_i$, which is the essential point in the sense of complexity reduction.

Let us suppose that two n-variable approximations are defined on the same domain with the same basis functions $\underline{\mu}_i$. One is called "original" and is defined by matrix $\underline{\underline{O}}$ of size $e_1 \times \cdots \times e_n \times p$ where p is m or $m+1$ (see (4.100) and (4.101)). The other one is called "additional" and is given by matrix $\underline{\underline{A}}$ of the same size. Let us assume that both approximations are reduced by the HOSVD complexity reduction technique detailed in Section 4.1 as: $(\underline{\underline{N}}_1,\cdots,\underline{\underline{N}}_n,\underline{\underline{O}}^r) = HOSVDR(\underline{\underline{O}})$ and $(\underline{\underline{G}}_1,\cdots,\underline{\underline{G}}_n,\underline{\underline{A}}^r) = HOSVDR(\underline{\underline{A}})$, where the sizes of matrices $\underline{\underline{N}}_i$, $\underline{\underline{O}}^r$, $\underline{\underline{G}}_i$ and $\underline{\underline{A}}^r$ are $e_i \times r_i^o$, $r_1^o \times \cdots \times r_1^o \times p$, $e_i \times r_i^a$ and $r_1^a \times \cdots \times r_1^a \times p$, respectively, and $\forall i : r_i^o \le e_i$ and $\forall i : r_i^a \le e_i$. This implies that the size of $\underline{\underline{O}}^r$ and $\underline{\underline{A}}^r$ may be different, thus the number and the shape of the reduced basis of the two functions can also be different.

The method detailed in the followings finds the minimal common basis for the reduced forms. (The reduction can be exact or non-exact, the number of the minimal basis in the non-exact case can be defined according to a given error threshold like in case of HOSVD.)

Method 4.5. Finding the minimal common basis $(\underline{\underline{U}}_i,\underline{\underline{\Phi}}^o,\underline{\underline{\Phi}}^a)$ for $(\underline{\underline{N}}_i, \underline{\underline{O}}^r)$ and $(\underline{\underline{G}}_i, \underline{\underline{A}}^r)$

The following steps have to be executed in each $i = 1..n$ dimension ($\forall i : (\underline{\underline{\mathbf{U}}}_i,\underline{\underline{\mathbf{\Phi}}}^o,\underline{\underline{\mathbf{\Phi}}}^a) = unify(i,\underline{\underline{\mathbf{N}}}_i,\underline{\underline{\mathbf{O}}}^r,\underline{\underline{\mathbf{G}}}_i,\underline{\underline{\mathbf{A}}}^r)$):

Step 1. The first step of the method is to determine the minimal unified basis $(\underline{\underline{U}}_i)$ in the i-th dimension:

Let us apply $(\underline{\underline{U}}_i,\underline{\underline{Z}}_i) = reduct(i,\lfloor \underline{\underline{N}}_i \quad \underline{\underline{G}}_i \rfloor)$ where function $reduct(d,\underline{\underline{B}})$ reduces the size of an n-dimensional ($e_1 \times \cdots \times e_n$) matrix $\underline{\underline{B}}$ in the d-th dimension. The results of the function are matrices $\underline{\underline{N}}$ and $\underline{\underline{B}}^r$. The size of $\underline{\underline{N}}$ is $e_d \times e_d^r$, $e_d^r \le e_d$; the size of $\underline{\underline{B}}^r$ is $c_1 \times \cdots \times c_n$, where

$c_d = e_d^r$ and $\forall i, i \neq d : c_i = e_i$. (The algorithm of the function is similar to the HOSVD reduction algorithm, the steps are: spread out, reduction, re-stack.)
As a result of the first step we get $\underline{\underline{U}}_i, \underline{\underline{Z}}_i$ where the size of $\underline{\underline{U}}_i$ is $e_i \times r_i^u$ ("u" denotes unified) and the size of $\underline{\underline{Z}}_i$ is $r_i^u \times (r_i^o + r_i^a)$.

Step 2. The second step of the method is the transformation of the elements of matrices $\underline{\underline{O}}^r$ and $\underline{\underline{A}}^r$ to the common basis:
Let $\underline{\underline{Z}}_i$ be partitioned as: $\underline{\underline{Z}}_i = \left[\underline{\underline{S}}_i \quad \underline{\underline{T}}_i\right]$, where the sizes of $\underline{\underline{S}}_i$ and $\underline{\underline{T}}_i$ are $r_i^u \times r_i^o$ and $r_i^u \times r_i^a$ respectively. $\underline{\underline{\Phi}}^o$ and $\underline{\underline{\Phi}}^a$ are the results of transformations $\underline{\underline{\Phi}}^o = product(i, \underline{\underline{S}}_i, \underline{\underline{O}}^r)$ and $\underline{\underline{\Phi}}^a = product(i, \underline{\underline{T}}_i, \underline{\underline{A}}^r)$ where function $(\underline{\underline{A}}) = product(d, \underline{\underline{N}}, \underline{\underline{L}})$ multiplies the multi-dimensional matrix $\underline{\underline{L}}$ of $e_1 \times \cdots \times e_n$ by matrix $\underline{\underline{N}}$ in the d-th dimension. If the size of $\underline{\underline{N}}$ is $g \times h$ then $\underline{\underline{L}}$ must hold that $e_d = h$. The size of the resulted matrix $\underline{\underline{A}}$ is $a_1 \times \cdots \times a_n$, where $\forall i, i \neq d : a_i = e_i$, and $a_d = g$.

Let us return to the original aim, which is injecting the points of additional approximation A into O^r that is the reduced form of the original approximation O. According to the problem the union of A and O^r must be done without the decompression of O^r. For this purpose the following method is proposed:
Let us assume that an n-variable original approximation O is defined by basis functions $\underline{\mu}_i^o$, $i = 1..n$ and matrix $\underline{\underline{O}}$ of size $e_1^o \times \cdots \times e_n^o \times p$ in the form of (4.100) (see also Fig. 4.14). Let us suppose that the density of the approximation grid lines is increased in the k-th dimension (Figs. 4.16 and 4.17). Let the extended approximation E be defined by matrix $\underline{\underline{E}}$ whose size agrees with the size of $\underline{\underline{O}}$ except in the extended k-th dimension where it equals $e_k^e = e_k^o + e_k^a$ (e_k^a indicates the number of additional basis functions) (Fig. 4.17). The basis of the extended approximation is the same as the original one in all dimensions except in the k-th one, which is simply the joint set of the basis functions of approximations O and A

$$\underline{\mu}_k^e = \underline{\underline{P}} \begin{bmatrix} \underline{\mu}_k^o \\ \underline{\mu}_k^a \end{bmatrix}, \tag{4.102}$$

$\underline{\mu}_k^a$ is the vector of the additional basis functions. $\underline{\underline{P}}$ stands for a perturbation matrix if some special ordering is needed for the basis functions in $\underline{\mu}_k^e$. The type of the basis functions, however, usually depends on their number due to various requirements of the approximation, like non-negative-ness, sum normalization, and normality. Thus, in case of increasing the number of the approximation points, the number of the basis functions is increasing as well and their shapes are also changing. In this case, instead of simply joining vectors $\underline{\mu}_k^o$ and $\underline{\mu}_k^a$, a new set of basis $\underline{\mu}_k^e$ is defined according to the type of the approximation like in Fig. 4.16. Consequently, having approximation O and the additional points the extended approximation E can easily be obtained as $\underline{\underline{E}} = fit(k, \underline{\underline{O}}, \underline{\underline{A}})$ where function $\underline{\underline{A}} = fit(d, \underline{\underline{L}}_1, \cdots, \underline{\underline{L}}_z)$ is for fitting the same sized, except in the d-th dimension, matrices in the d-th dimension: Matrices $\underline{\underline{L}}_k = [l_{k,i_1,\cdots,j_n}]$ have the size of $e_{k,1} \times \cdots \times e_{k,n}$, $k = 1..z$ to the subject that $\forall k, i, i \neq d : e_{k.i} = e_i$. The resulted matrix $\underline{\underline{A}}$ has the size as $e_1 \times \cdots \times e_n$, where $e_d = \sum_{k=1}^{z} e_{k,d}$ and the elements of $\underline{\underline{A}} = [a_{i_1,\cdots,i_n}]$ are $a_{i_1,\cdots,i_n} = l_{k,j_1,\cdots,j_n}$

where $\forall t, t \neq d : i_t = j_t \ , \ i_d = j_d + \sum_{s=1}^{k-1} e_{s,d} \ , \ k = 1..z$.

More precisely, according to the perturbation matrix in (21) $\underline{\underline{E}} = product(k, \underline{\underline{P}}, fit(k, \underline{\underline{O}}, \underline{\underline{A}}))$. Once again, the main goal is to embed the additional approximation to the reduced O, namely, to $(\underline{\underline{N}}_1, \cdots, \underline{\underline{N}}_n, \underline{\underline{O}}^r) = HOSVDR(\underline{\underline{O}})$ on such a way that the result will be the same as the reduction of the extended approximation $(\underline{\underline{U}}_1, \cdots, \underline{\underline{U}}_n, \underline{\underline{E}}^r) = HOSVDR(\underline{\underline{E}})$. As a matter of fact, the result of SVDR is not unique. The "same" means here that the reduced size of $\underline{\underline{E}}$ must be the same on both ways.

Method 4.6. Embedding the new approximation *A* into the reduced form of *O*

The steps of the method are as follows:

Step 1. First, the redundancy of approximation A is filtered out by applying $(\underline{\underline{G}}_1, \cdots, \underline{\underline{G}}_n, \underline{\underline{A}}^r) = HOSVDR(\underline{\underline{A}})$.

Step 2. As next, the merged basis of O^r and A^r is defined. The common minimal basis is determined in all dimensions except the k-th one:

Let $\underline{\underline{W}}_{[1]} = \underline{\underline{O}}^r$ and $\underline{\underline{Q}}_{[1]} = \underline{\underline{A}}^r$. Then, for t= 1 … n-1 evaluate $(\underline{\underline{U}}_j, \underline{\underline{W}}_{[t+1]}, \underline{\underline{Q}}_{[t+1]})$ as

$$(\underline{\underline{U}}_j, \underline{\underline{W}}_{[t+1]}, \underline{\underline{Q}}_{[t+1]}) = unify(j, \underline{\underline{N}}_j, \underline{\underline{W}}_{[t]}, \underline{\underline{G}}_j, \underline{\underline{Q}}_{[t]}) \text{ where } j = \begin{cases} t & t < k \\ t+1 & t \geq k \end{cases}.$$

Finally, let $\underline{\underline{\Phi}}^o = \underline{\underline{Q}}_{[n]}$ and $\underline{\underline{\Phi}}^a = \underline{\underline{Q}}_{[n]}$.

For the k-th dimension let $\underline{\underline{M}} = \underline{\underline{P}} \begin{bmatrix} \underline{\underline{N}}_k & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{G}}_k \end{bmatrix}$, where $\underline{\underline{0}}$ contains only zero elements and $\underline{\underline{P}}$ can ensure any special ordering, as used in (4.102). $\underline{\underline{N}}_k$ and $\underline{\underline{G}}_k$ are full rank matrices which means that no further (exact) reduction of $\underline{\underline{M}}$ can be obtained. According to the basis, matrices $\underline{\underline{\Phi}}^o$ and $\underline{\underline{\Phi}}^a$ are unified as $\underline{\underline{F}} = fit(k, \underline{\underline{\Phi}}^o, \underline{\underline{\Phi}}^o)$.

Step 3. Finally, the redundancy is removed, i.e., the linear dependence between matrices $\underline{\underline{\Phi}}^o$ and $\underline{\underline{\Phi}}^a$ is filtered out of $\underline{\underline{F}}$ by $(\underline{\underline{K}}, \underline{\underline{E}}^r) = reduct(k, \underline{\underline{F}})$. Thus, $\underline{\underline{U}}_k = \underline{\underline{M}}\,\underline{\underline{K}}$. (Here we would like to note again that $\underline{\underline{K}}$ is full rank matrix, i.e., no further (exact) reduction of $\underline{\underline{U}}_k$ can be obtained.)

Matrix $\underline{\underline{U}}_i$, having the size of $e_i \times r_i^u$, is to transform the basis as $\underline{\mu}_i^u = \underline{\underline{U}}_i^{\mathrm{T}} \underline{\mu}_i^e$. The size of matrix $\underline{\underline{E}}^r$ is $r_1^u \times \cdots \times r_n^u \times p$. For more details about the improvement of the approximation, see [S21].

4.7 Anytime Control Using SVD Based Models ([S12], [S68], [S72], [S91], [S92], [S93], [S94], [S96], [S101])

Recently, the popularity of fuzzy control has grown rapidly. There are numerous successful applications of fuzzy control which affect on the analysis and design of fuzzy control systems. The previously discussed ideas and tools can fruitfully be combined if a certain type of fuzzy model based control, namely TS fuzzy modeling and for the controller design Parallel Distributed Compensation (PDC) [38] is used. If the model approximation is given in the form of TS fuzzy

model, the controller design and Lyapunov stability analysis of the nonlinear system reduce to solving the LMI problem. This means that first of all we need a Takagi-Sugeno model of the nonlinear system to be controlled. The construction of this model is of key importance. This can be carried out either by identification based on input-output data pairs or we can derive the model from given analytical system equations.
The PDC offers a direct technique to design a fuzzy controller from the TS fuzzy model. This procedure means that a local controller is determined to each local model. This implies that the more complex the system model is, the more complex controller will be obtained. According to the complexity problems outlined in the previous sections we can conclude that when the approximation error of the model tends to zero, the complexity of the controller explodes to infinity. This pushes us to focus on possible complexity reduction and anytime use.
SVD based complexity reduction can be applied on two levels in the TS fuzzy controller. First, we can reduce the complexity of the local models (local level reduction). Secondly, it is possible to reduce the complexity of the overall controller by neglecting those local controllers, which have negligible or less significant role in the control (model level reduction). Both can be applied in an anytime way, where we take into account the "distance" between the current position and the operating point, as well. The model granularity or the level of the iterative evaluation can cope with this distance: the further we are, the more rough control actions can be tolerated. Although, the approximated models may not guarantee the stability of the original nonlinear system, this can also be ensured by introducing robust control (see e.g. [39]).

4.8 Anytime Development Environment and the ATDL Anytime Description Language ([S24], [S100], [S97], [S103])

In this chapter, we introduce a simple anytime development tool and an anytime description meta-language to ensure an effective environment for the development. In Section 4.8.1 the basic structure and main requirements are outlined. The anytime library is detailed in Section 4.8.2, while Section 4.8.3 presents the instructions and usage of the introduced ATDL meta-language.

4.8.1. Anytime Development Tool

The design of an anytime system has the main parts as

- (System specification/system decomposition) [D]
- (Creation of anytime/non-anytime algorithms) [D]
- Determination of the profiles [D] [C]
- Compilation of the anytime algorithms [C]
- Monitoring/scheduling [R]

The points surrounded by round brackets are out of the scope of this work. It can be easily noticed that the points marked by [D] belong to the design process. Mark [C] is for activities done by the Anytime Compiler which is a part of the Anytime Development Tool and is responsible for the creation of the run-time information database of the system used by the Monitor during the operation (see *(C1)* and *(C2)* in Fig. 4.4.). The Monitor operates in run-time [R], obviously.
Analyzing the operation of the anytime systems, it can be realized that on one hand, there is a gap between the Compiler and the Monitor in that sense that the compilation occurs before the activation of the system in contrast to the monitoring which works after the activation, during the process of the system operation. On the other hand, compilation and the monitoring are strongly related to each other, a particular monitoring scheme requires certain compilation modes and methods.
The operation of the elementary modules is implemented by functions written in some programming language and the execution of these algorithms is started by function-call-like code fragments. The gap itself is caused by the simple fact that it is not trivial (or is even impossible) to

"dig out" the execution order from a source code, especially off-line. (E.g. if these function calls are located inside *if* or *for* structures.)
The way of the description of the anytime algorithms is not as easy in practice as it comes from the theoretical results and from the abstract handling of the profiles at first glance because the profiles are tables with certain sampling properties and dimensions. It is unlikely to have profiles with fitting dimensions and appropriate sampling intervals. Different profiles are valid for different definitions of the output qualities and they have to be adjusted properly for different target platforms.
In the following, a simple meta-language is presented which can be used advantageously in anytime systems. Needless to say, the complete solution of the above problems is far from what is proposed here. The requirements set against the meta-language follow from the features of the anytime operation, i.e., that from system point of view, compilation (fitting) can be best down in compilation time while monitoring should be executed during run time.
The problem is that for the correct compilation we need a lot of information about the modules that are not trivial to "dig out" from the source files (execution order, etc.). The result of the anytime development has to be an executable file written in a given programming language but on source code level. This can not simply be fulfilled in case of high level languages. One solution is that the anytime development tool prepares a so called *target source code* from the anytime algorithms, from their description, and from the system source code. The target source code is compiled to an executable file by a special compiler. This is supported by a meta-language which helps the anytime development tool to connect the anytime algorithms and their description tables stored in files.
For the above purposes, a simple AnyTime Description meta-Language (ATDL) is proposed which makes the full description of anytime systems possible, in cooperation with the C++ high level programming language. From the ATDL description, a C++ code is generated, which operates as an extension to the anytime operating system. This extension can be included in the operating system on source code level and results in the target source code. Thus, to each anytime system a separate anytime operating system is created. The block diagram of this architecture is shown in Fig. 4.19.
The anytime system design block covers the source code of the system, where the system features and properties are given in ATDL language. It collects all the results of the design process: it contains source code portions, the architectural description of the anytime system, and many option settings.
The anytime library contains the source codes, profiles of the anytime algorithms, and their ATDL descriptions.

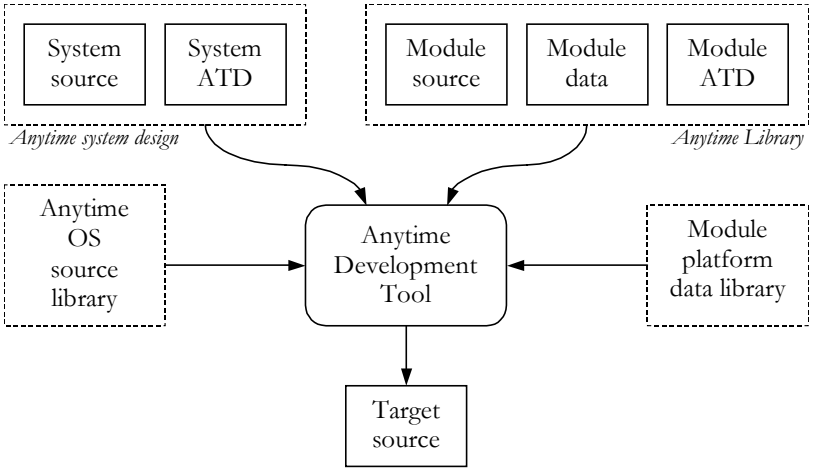


Fig. 4.19 Block diagram of the architecture of anytime development

The anytime platform data library includes the timing data. This database is built by measurement performed by the Anytime Development Tool with the aid of the so-called *timer functions*. This database is not necessarily created in compilation time and at every design iteration (step) since working with the same platform means the platform database is needed to be created only once.
The anytime operation and the monitor are described in the anytime OS source library. This part of the development system contains template-like source code portions for the several types of anytime systems. It supports various monitoring schemes and an individual operation system can be created from it for the given anytime system design. This results in a set of compilable and/or linkable *target sources* and data files which are processed by an extant language compiler (C++ compiler, for example) to get an executable software as the final result of the anytime system design process.
The anytime development tool ensures the development environment. The target source is the result of the development (compilable C++ source and data files).

4.8.2. Anytime Library

An anytime algorithm can be completely described by the (1) Source code(s) (written in certain language(s)), (2) Profiles, (3) Description, and (4) Other information.
The *source code* can be written in several languages and compiled to intermediate files like object files or can be included into the source of the system code.
Profiles bring the essential information about the time dependent performance of the algorithms. Different platforms have different timing properties therefore the profiles have to be transformed. A reference computer has been suggested by Zilberstein which can be compared to a given platform in order to connect profile times with the real time. However, it is apparent that various possible execution times (allocations) of an algorithm are resulted from conditional commands in its code therefore choosing timing properties of a specific platform for parametrisation of the profiles is not the most efficient way. An algorithm can have only finite number of possible allocations and if the profiles would have the index of these allocation times as their time parameter then they would be platform independent. This allocation time index is called as *step time*. The quality in the profiles can be defined in several ways so a particular profile has to store its *quality function* (a mapping from output values to quality values or distribution) to keep all the information consistent.
The platform-independent anytime algorithms can be collected in an Anytime Library which can provide basic or complex elements for the system design. Fig. 4.20 summarizes the data needed to completely characterize an anytime algorithm.

Algorithm Characterization	
Source code(s): + language info., compiler/compilation options	
Profiles	profile type
	profile data
	quality function(s)
	source code of the profile maker fn.

Fig. 4.20 Anytime algorithm characterization

The Module description is responsible for supporting the linkage between the pure source, the profile, and timing database (see later), and the target source. It can also define calling conventions of the implementation function. The *profile maker function* creates the profiles of the algorithm by executing it for appropriate inputs, determining the output qualities or distributions in the proper way, and collecting the performance data. The profile data is an array containing the samples of the mapping implemented by the profile.

4.8.3. Description of anytime systems: the new ATDL meta-language

The Module ATD and System ATD sub-blocks in Fig. 4.19. can be given in a meta-language. A simple example of a proposed a language is shown in this section. The AnyTime Description Language (ATDL) can be used to define modules, tasks, and systems. From the ATDL description a C++ code is generated which operates as an extension to the anytime operating system. This extension can be included in the operating system on source code level and results in the target source code. Thus, to each anytime system a separate anytime operating system is created.
Assume a simple anytime system shown in Fig. 4.21.

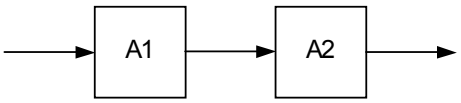


Fig. 4.21 Simple anytime chain

In the followings, the simplicity and plasticity of the proposed ATDL language is illustrated via characteristic parts of the problem description: the descriptions of the system, a module, and a (sub)task are given in Figs. 4.22-4.25. The **system**, **task**, and **module** commands define the anytime system, task-units, and anytime modules, respectively. Tasks can be included in the system with the help of the **include** command. With the **monitor** command the type of monitoring (**active**, **passive**) can be set and the **header** points to the header file of the system. The utility function file can be pointed by the **utility** command. The connections among the modules can be defined by the **connection** command. **External_use** means that the given inputs and outputs are the external inputs and outputs of the task-unit. The **profile**, **header**, and **timing** elements give the corresponding file names. The **implementation** is followed by the interface of the implementation function of the algorithm. The profile functions can be given after the **profile_maker** or **maker** commands.
The ATDL code parts can easily be merged with C++ codes resulting in an anytime operating system.
The ATDL description of the block A1 can be given in the form shown in Fig.4.22. The profile is stored in a `.prf` file, the header (included by the `#include` directive) of the module functions is "`A1.h`", the timing data are located in file "`A1.tim`".
Fig. 4.23. shows an example of a header. Note that the algorithm's identifier (class) is "`MA1`".
The request of the task consisting of the execution of the two modules can also be described by the ATDL. This task description determines the structure of the system (see Fig. 4.24).
The two module definitions (A1 and A2) have to be included to instantiate the algorithm `a1` of type `MA1`, and the `a2` which is an `MA2` typed anytime algorithm. The single connection between the modules is defined by the `connections` keyword. The output and the input of the system can be referred by the identifiers `T1_input` and `T1_output`.
The system definition has the form shown in Fig. 4.25. In this example, the system can run only one task, `T1`.

```
// A1.atd
module MA1(A1_INPUTSTRUCT input - double output)
{
      profile "A1.prf";
      header "A1.h";
      timing "A1.tim";
      implementation A1(input, output);
      profile_maker A1_profile_maker;
      timing_meter A1_timing_meter;
};
```

Fig. 4.22 ATDL Example for module (algorithm) definition

```
// A1.h
#if !defined(_A1_H_INCLUDED_)
#define _A1_H_INCLUDED_
#include "atdefs.h"

      struct A1_INPUTSTRUCT
      {
            int a, b;
      };

      void A1(A1_INPUTSTRUCT &, double &, unsigned);
      CProfile* A1_profile_maker();
      CTiming* A1_timing_meter();

#endif
```

Fig. 4.23 ATDL Example for an algorithm header

```
// T1.atd
task T1
{
      utility "T1.utl";
      include "A1.atd";
      include "A2.atd";

      MA1 a1;
      MA2 a2;

      connections a1.output - a2.input;

      external a1.input T1_input;
      external a2.output T1_output;
};
```

Fig. 4.24 ATDL example for Task definition

```
system Sys
{
      include "T1.atd";
      header "Sys.inl";
      monitor active;
};
```

Fig. 4.25 ATDL example for system definition

4.9 Conclusion

In modern embedded signal processing, monitoring, diagnostics, measurement, and control systems, the available time and resources are often not only limited, but can also change during the operation of the system. In these cases, the so called anytime models and algorithms can be used advantageously. While different soft computing methods are widely used in system modeling, their usability is limited, because the lack of any universal method for the determination of the needed complexity often results in huge and redundant neural networks/ fuzzy rule-bases. In Part IV, we investigated a possible way to carry out anytime processing in certain fuzzy and neural network models, with the help of the (Higher Order) Singular Value Decomposition based transformation and complexity reduction.

Practical questions of implementing and operating anytime systems are also analyzed. Although, such systems may provide an optimal tradeoff between time/resource needs and computational complexity and the quality (accuracy) of results, since the monitor operates under prescribed response time requirements and the number and complexity of the executable tasks may be very high, new considerations must constantly be made to achieve optimal or acceptable performance. In software terms, this requires the application of special compilation methods dealing also with timing considerations and constraints of the underlying operating system and even with the run-time characteristics of the monitor. This must be supported by anytime development tools and special anytime description languages. In this part, a hierarchical compilation method is also introduced together with theoretical considerations about a possible anytime development tool and the basics of the anytime description meta-language ATDL are presented, as well.

Part V Observer Based Iterative System Inversion

Nowadays model based techniques play very important role in solving measurement and control problems. Recently for representing nonlinear systems fuzzy and neural network (NN) models became very popular. For evaluating measurement data and for controller design also the inverse models are of considerable interest. In Part V., different observer based techniques to perform fuzzy and neural network model inversion are introduced. The methods are based on solving a nonlinear equation derived from the multiple-input single-output (MISO) forward fuzzy model simple by interchanging the role of the output and one of the inputs. The utilization of the inverse model can be either a direct compensation of some measurement nonlinearities or a controller mechanism for nonlinear plants. For discrete-time inputs the technique provides good performance if the iterative inversion is fast enough compared to system variations, i.e., the iteration is convergent within the sampling period applied. The proposed method can be considered also as a simple nonlinear state observer which reconstructs the selected input of the forward (fuzzy or NN) model from its output using an appropriate strategy and a copy of the fuzzy model itself [S49]. Improved performance can be obtained by introducing genetic algorithms in the prediction-correction mechanism [S56]. Although, the overall performance of the suggested technique is highly influenced by the nature of the non-linearity and the actual prediction-correction mechanism applied, it can also be shown that using this observer concept completely inverted models can be derived. The inversion can be extended towards anytime modes of operation, as well, providing short response time and flexibility during temporal loss of computational power and/or time [S69].

5.1 Introduction

Model based schemes play an important role among the measurement and control strategies applied to dynamic plants. The basically linear approaches to fault diagnosis [1], optimal state estimation [2] and controller design [3] are well understood and successfully combined with adaptive techniques (see. e.g. [4]) to provide optimum performance. Nonlinear techniques, however, are far from this maturity or still are not well understood. Although, as we have mentioned earlier, there is a wide variety of possible models to be applied based on both classical methods [5] and recent advances in handling [6] information, up till recently practically no systematic method was available which could be offered to solve a larger family of nonlinear control problems. The efforts on the field of fuzzy and neural network (NN) based modeling and control however, seem to result in a real breakthrough also in this respect. With the advent of adaptive fuzzy controllers very many control problems could be efficiently solved and the model based approach to fuzzy controller design became a reality [7]. Using model based techniques in measurement and control also the inverse models play a definite role [4].
Recently different techniques have been published (see e.g. [7]-[11]) for inverting certain fuzzy models, however, exact inverse models can be derived only with direct limitations on the fuzzy models applied. In this part, an alternative approach is investigated which is based on the quite general concept of state observation widely used in measurement and filtering applications. The key element of this concept is to force a model of a physical system to “copy” the behavior of the system to be observed (see Fig. 5.1). This scheme is the so called observer structure which is a common structural representation for the majority of iterative data and/or signal processing

algorithms. From Fig. 5.1, it is obvious that here the inversion of dynamic system models is considered.
Traditionally, the observer is a device to measure the states of dynamic systems having state variable representation. According to our proposition, however, these states can be regarded as unknown inputs, and therefore their "copy" within the observer as the result of model inversion. In this scheme, the fuzzy models appear as static nonlinear Multiple Input Single Output (MISO) mappings from the state variables to the system output, i.e., represent the output equation. A copy of this output equation is also present within the observer, and in this case observer dynamics is simply due to the iterative nature of the algorithm.

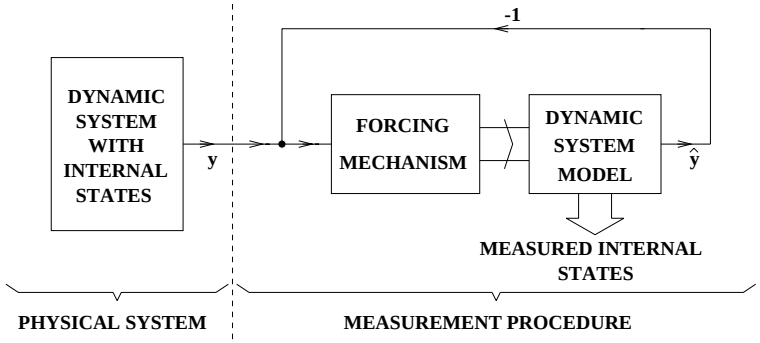


Fig. 5.1 The observer concept

At this point it is important to note that the inversion is not unique if more than one input is considered, i.e., from one observer input value more than one output is to be calculated. In this chapter only unique inversions are investigated, therefore the calculation of one controllable input is regarded based on the desired output and uncontrollable input values. Although, we also point out the generalization of the inversion for more then one input.
The chapter is organized as follows. The possible role of inverse fuzzy models and the main features of the explicit inversion methods are described in Section 5.2 Section 5.3 presents the observer based iterative inversion technique. Section 5.4 is devoted to the genetic algorithm and its application within the inversion procedure. In Section 5.5 the inversion scheme is extended to be able to operate in anytime modes.

5.2 Inverse fuzzy models in measurement and control

Nowadays solving measurement and control problems involves model-integrated computing. This integration means that the available knowledge finds a proper form of representation and becomes an active component of the computer program to be executed during the operation of the measuring and control devices. Since fuzzy models represent a very challenging alternative to transform typically linguistic a priori knowledge into computing facilities therefore it worth reconsidering all the already available model based techniques whether a fuzzy model can contribute to better performance or not.
The role of the inverse models in measurements is obvious: observations are mappings from the measured quantity. This mapping is performed by a measuring channel the inverse model of which is inherent in the data/signal processing phase of the measurement. In control applications inverse plant models are to be applied as controllers in feedforward (open-loop) systems, as well as in various alternative control schemes. Additionally there are very successful control structures incorporating both forward and inverse plant models (see e.g. [4]).

In measurement and control applications the forward and inverse models based on fuzzy techniques are typically MISO systems (see Fig. 5.2) representing static nonlinear mappings. Typical Single Input Single Output (SISO) dynamic system models are composed of two delay lines and a MISO fuzzy model as in Fig. 5.3. The first delay line is a memory for a limited number of last input samples ($u(n)$, $u(n\text{-}1)$, …) while the second one contains the last segment of the outputs ($\hat{y}(n), \hat{y}(n-1),...$). In the scheme of Fig. 3, $\hat{y}(n+1)$ is an estimated system output for time instant $n+1$ calculated from input and estimated output samples available at time instant n. It is important to note that there is an inherent delay in such and similar systems since the model evaluations take time. The (partial) inverse of such a dynamic system is also a SISO scheme (see Fig. 5.4) which is in correspondence with Fig. 5.3 except the role of the input and output is transposed. The input $r(n)$ is a sample of the reference signal and a predicted value of the input will be the model output. It is obvious from these figures that if the forward fuzzy model is available only the inverse (nonlinear static) mapping must be derived.

There are different alternatives to perform such a derivation. One alternative is to invert the fuzzy model using the classical regression technique based on input-output data. To solve this regression problem, iterative algorithms can also be considered. The result of such a procedure is an approximation of the inverse, and the accuracy of this approximation depends on the efficiency of the model fitting applied. In this case, however, the inverse is not necessarily a fuzzy model in its original sense.

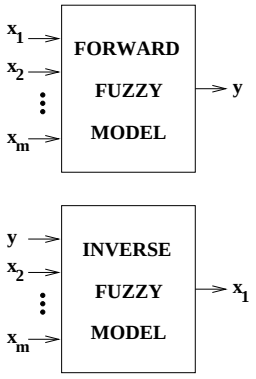


Fig. 5.2 Forward and inverse model

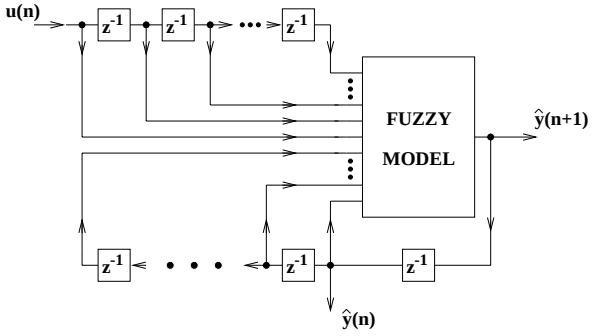


Fig. 5.3 Dynamic system model

Recently very interesting methods have been reported for exactly inverting certain type of fuzzy models [7]. For the case of the standard Mamdani fuzzy model [12] with singletons in the rule consequents exact inverse can be derived. Obviously the general conditions of invertibility must be met: the forward fuzzy model should be strictly monotone with respect to the input which is considered to be replaced by the output. If the forward model implements a noninvertible function it must be decomposed into invertible parts and should be inverted separately. There are,

however, several other prerequisites of the exact inversion concerning both antecedent and consequent sets, rule base, implication and T-norm, as well as the defuzzification method. The construction of the inverse proved to be relatively simple at the price of strong limitations.
There are other promising approaches ([8], [10]) where the inversion is solved on linguistic level, i.e. rule base inversion is performed. These techniques can accommodate various fuzzy concepts but must be combined with fuzzy rule base reduction algorithms if the number of the input sets is relatively high.

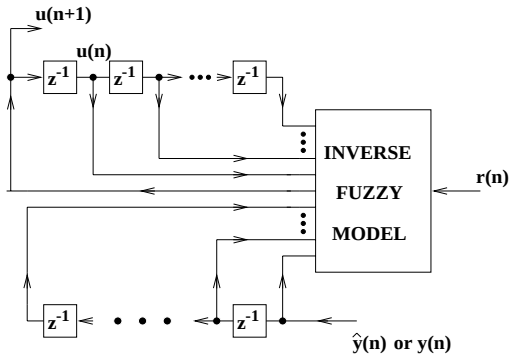


Fig. 5.4. Inverse model

5.3 Observer based iterative inversion

Theorem 5.1. Based on the observer concept, globally convergent MISO fuzzy model inversion can be achieved under the assumption of model observability ([S37], [S49]).

Proof.
See Methods 5.1. and 5.2.

Method 5.1.

Step 1. Initial direct model building. The general concept of the observer is represented by Fig. 5.1. The physical system produces output y and we suppose that its behavior can be described by a dynamic system model e.g. like the structure given in Fig. 5.3.
The fuzzy model can be created based on observed input-output pairs of the system, probably with support of human expert knowledge. This system description becomes the inherent part of the measurement procedure and is forced to behave similarly to the physical system. A more detailed description of this idea for static nonlinear fuzzy model is given by Fig. 5.5. As an example, here a three-input one-output fuzzy model can be shown.

Step 2. Iteration. Repeat steps 3-5 till convergence or stopping criteria is reached.

Step 3. Estimation of the output-error based cost function. Input x_1 is considered unknown and therefore to be observed via comparing the output of the physical system and that of the model.

Step 4. Evaluation of the adaptation (forcing) mechanism.

Step 5. Correction of the unknown model input estimation with the help of step 3. If the correction (forcing) mechanism is appropriate the observer will converge to the required state and produce the estimate of the unknown input. The strength of this approach is that this iterative evaluation is easy to implement, e.g., using standard digital signal processors. The complete system of Fig. 5.5 can be embedded into a real-time environment, since the necessary number of iterations to get the inverse can be performed within one sampling time slot of the measurement or control application.

For the correction several techniques can be proposed based on the vast literature of numerical methods (see e.g. [13]) since the proposed iterative solution is nothing else than the numerical solution of a single variable nonlinear equation. The iteration is based on the following general formula

$$\hat{x}(n+1) = \hat{x}(n) + correction[y - \hat{y}, \hat{x}(n), f(), \mu] \tag{5.1}$$

Where y is the output of the unknown system to be inverted, $\hat{y}$ denotes the estimation of the output produced by the (direct fuzzy) model, $f(\,)$ stands for the nonlinear function to be inverted and μ for the step size. The convergence properties of the inversion depend on the convergence properties of the applied correction technique, i.e. if locally convergent method is used, then only local convergence of the iteration can be ensured. Although, after the convergence of a step, in most of the cases, the output remains within a predictable distance of the previous output if the input change is under a limit and vice versa.
One of the simplest solutions is if Newton iteration is applied. In this case, (5.1) has the form of

$$\hat{x}(n+1) = \hat{x}(n) + \mu \frac{(y - \hat{y})}{f'(\hat{x}(n))} \tag{5.2}$$

where $f'(\,)$ denotes the derivative of the $f(\,)$. This latter must be evaluated locally using simple numerical technique:

$$f'(\hat{x}(n)) \cong \frac{f(\hat{x}(n) + \Delta) - f(\hat{x}(n) - \Delta)}{2\Delta} \tag{5.3}$$

The computational complexity of this iterative procedure depends mainly on the complexity of the forward fuzzy model itself. It is anticipated, however, that after the first convergence, if the input of this observer changes relatively smoothly, then in the majority of the cases only a few iterations will be required to achieve an acceptable inverted value.
In principle the proposed method can be generalized even for two or more (forward model) inputs (see Fig. 5.5). For this generalization, however, it must make clear that to calculate two or more inputs at least the same number of ("measured") outputs is required. With static nonlinear MISO models this is not possible. If the fuzzy model in Fig. 5.5 is replaced by a dynamic, state variable model, where the "inputs" will correspond to the state variables, following the state transitions new outputs can be "measured" and as many values can be collected as required to the iterative solution of the multi input problem. This idea obviously requires the "observability" [2] of the states and the application of state variable models instead of the schemes of Fig. 5.4 and Fig. 5.5. The complexity of the iterative technique to be applied in such cases is under investigation.

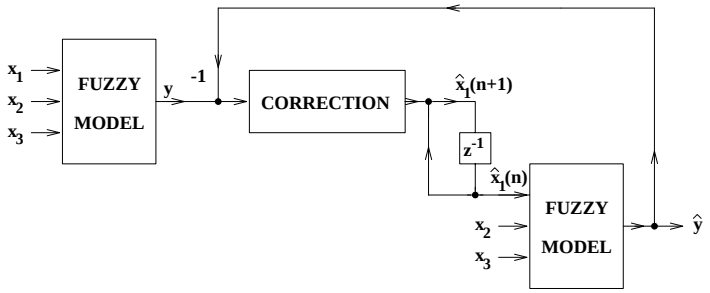


Fig. 5.5 Block diagram of the iterative inversion scheme for a three input one output case

5.4 Genetic algorithms for fuzzy model inversion

Unfortunately, the simple Newton iteration may fail if the nonlinear function represented by the fuzzy model has multiple minima. This is because gradient-based techniques work using only locally available information. If multiple minima may occur global search techniques are to be considered, however usually with a burden of high complexity.

5.4.1 The multiple root problem

Another problem is the case of multiple roots. Fig. 5.6 illustrates these situations. Here a function $y=f(x_1,x_2,...)$ is shown with fixed x_2, … input values. The calculation of the inverse means to find the corresponding roots x_{10i} (i=1,…) for a given $y=y_0$. Unfortunately, in general it is a hard problem to decide which root equals (or approximates) the actual fuzzy model input. At our present knowledge the only thing we can state is that, having the correspondence, small variations of the input should result in small deviation of the estimate. Intuitive techniques based on Taylor series expansion may provide proper orientation, but the general solution is still an open issue.
Since the situation indicated in Fig. 5.6 is not necessarily typical, there are intervals with one-to-one correspondence between input and output. Based on these values the root decision problem can be solved.
It is important to note at this point that if multiple roots may occur in a region, and there is no proper hint available to find the appropriate one, then it is unavoidable to search for all the roots and make the decision afterwards.

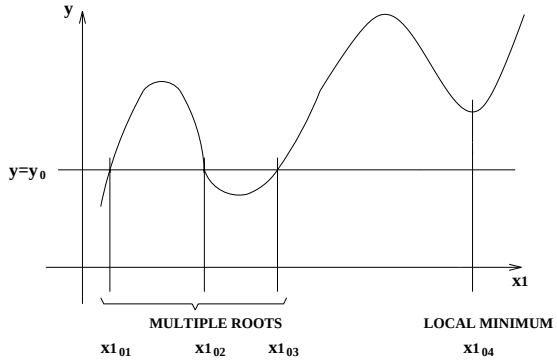


Fig. 5.6. Illustration for the multiple roots-local minimum problem

5.4.2 Genetic algorithm based inversion

Genetic algorithms (GAs) [14],[15] are optimization methods which search in the entire space to find the global optimum, and they are quite resistant against the problem of local minima. The classical versions of GAs are applied to solve discrete optimization problems, however, by applying new forms for representation and for genetic operators more and more continuous optimization problems are solved based on GAs.
In the continuous optimizations the chromosomes should store, instead of the traditional binary strings (as representation of numbers, e.g. [16]), one or more real numbers. Here the so-called direct real coding is used, (see e.g. [17]) which is a more efficient representation and it is immune against several unfortunate effects [15],[17].
The presented methods apply real-coded representation. The chromosome contains basically one real number x that is the candidate for x_1. The fitness-function $ff(x)$ is derived from the difference $d(x) = |f(x) - y|$ as $ff(x) = M - d(x)$ where, M = max $d(x)$ + min $d(x)$, max and min are evaluated over the entire population.

The initialization of the population and the mutation are performed in a similar way. The population is derived from an interval I_n which is updated in every generation. The operators work on random values having uniform distribution over this interval.
If the tracing mode is applied, then the initial population comes from interval I_0. This interval is around the value x_0 with radius r which is derived from the previous x_1 and Δx_1 values: $x_0 = x_1 + \Delta x_1$, $r = \Delta x_1$. As the generations go on, I_n is grown as follows: $I_{n,\min} = I_{0,\min} - nv_1$, $I_{n,\max} = I_{0,\max} + nv_1$, where $v_1 = 0.5\Delta x_1$.
The crossover operator is the same in both the global search and the tracing mode methods. It uses linear interpolation of the function, deriving from parent values x_A, x_B and from $f(x_A)$, $f(x_B)$: $x_{C0} = \frac{x_A f(x_B) - x_B f(x_A)}{f(x_B) - f(x_A)}$. The final children are taken from an interval around x_{C0}, with radius $r = 0.1 \min(|x_A - x_{C0}|, |x_B - x_{C0}|)$.
The GAs use roulette wheel selection. The probability of the crossover is 0.8, the mutation works with 0.05. The population size is 10. The terminating condition depends on the error of the best value: $d(x_b) / |f'(x_b)| < E$ where $f'(x_b)$ is estimated by the nearest x to x_b, E is the estimation error limit.
In our experience the performance of the GAs for iterative fuzzy model inversion is quite acceptable, but its computational complexity is relatively high.
In the followings, as a low complexity global search technique, a new hybrid method is presented which is based on the combination of some low complexity local search technique (like Newton iteration) and globally convergent genetic algorithms.

5.4.3 The new globally convergent hybrid inversion method

To decrease the complexity of the inversion, a combined technique has been elaborated which is basically a Newton method, but if it fails, it is switched to the GAs for one iteration.

Method 5.2. ([S37], [S56])

Our assumption concerning operation is that of associated with the observer mechanism, i.e., a tracking mode, where the iteration is based on a "good", previous estimate, since the inputs and the output do not change drastically. The introduction of genetic search can be justified (1) at the beginning, as we start the iterative procedure, and try to find the first estimate of the input, (2) during continuous operation, when gradient based search techniques fail due to local minima. In other cases the simple Newton iteration might be acceptable.
The major steps of the combined method are as follows:

Genetic algorithm:
Evaluate genetic algorithm
Switch to Newton iteration

Newton iteration:
Set the input variables, evaluate the fuzzy model $y = f(\ )$, $y(n)$
Evaluate equation (5.3)
IF $|y - \hat{y}(n)| / |f'(\hat{x}(n))| < E$, where $E > 0$ is the estimation error limit, THEN stop
ELSE IF $|f'(\hat{x}(n))| < \delta$, where $\delta > 0$ is a small number, OR IF the number of iterations is larger than a predefined value N, THEN switch to the genetic algorithm
ELSE Evaluate equation (5.2)
$n = n + 1$
Switch to Newton iteration
The advantageous, low complexity figures of the proposed hybrid method has been proved by simulations (see also [S56] and Subsection 5.7).

5.5 Fuzzy inversion in anytime systems

With the help of the (HO)SVD based complexity reduction (see Section 4.3) fuzzy systems can be operated in an anytime mode. This operational form can be advantageous in model inversion, as well.

Method 5.3. ([S37], [S69], [S84])

The steps of preparing a fuzzy model to be able to work in anytime inversion, are the same as described in Section 4.6., i.e. first a practically "accurate" fuzzy system is to be constructed, for which expert knowledge and/or training data can be used. In this step there is no need to deal with the complexity of the obtained model.
In the second step, with the SVD-based complexity reduction algorithm a reduced, but accurate rule-base can be generated. In case of PSGS fuzzy systems, the model can be transformed to a form which can be evaluated iteratively (Section 4.5).
In all other cases, the modular frame has to be used (Section 4.2.2) and further variations of the rule-base must be constructed, with different accuracy and complexity. An alternative rule-base can be characterized by its complexity (proportional with $n_1^r * \ldots * n_n^r$) and its error that can be estimated by the sum of the discarded singular values.
The different rule-bases can form the different units realizing a given module. The expert system can adaptively change the units - rule-bases - according to the available computing time and resources at the moment. Because the inference algorithm is the same, only the rule-bases - a kind of parameter set - must be changed.
For anytime fuzzy model inversion the forward model within the inversion scheme has to be replaced by the appropriate truncated model corresponding to the actual circumstances, i.e., the temporarily available amount of computational resources and time. The computational need of the inversion based on the reduced forward model is directly proportional to the computational complexity of the used model, thus the necessary model reduction can be computed.
Since convergence can be ensured in the observer scheme, the accuracy of the inverted model will be determined by the accuracy and the transfer characteristics of the forward model. This latter can be approximated locally using e.g. the simple numerical technique in (5.3).

5.6 Inversion of Neural Network Models ([S118])

When we refer to "inversion" of a neural network for the acquisition of certain input parameters, we are actually referring to a constrained inversion. That is to say, given the functional relationship of the NN input to the output, we have the forward and want to have the inverse relationships (see Fig. 5.7), where $x_1, x_2, \ldots, x_n$ is the input set of the forward neural network model and $\underline{y}$ is the output vector of the model. Fig. 5.7b illustrates the inverse NN model, where output vector $\underline{x}^s$ is composed of a subset of the input variables of the forward model. Given the forward relationship described in a neural network form

$$\underline{y} = f(\underline{x}^s, \underline{u}) \tag{5.4}$$

where $\underline{x}^s$ stands for the unknown environmental parameters we wish to obtain, and $\underline{u}$ denotes a vector of the known environmental and system parameters ($\underline{x} = \underline{x}^s \cup \underline{u}$), we wish to find an inverse mapping

$$\underline{x} = g(\underline{y}, \underline{u}) . \tag{5.5}$$

For the inversion, Method 5.2 can be used, however with taking into consideration that we need as many input as many output is to be inverted, i.e. e.g. in case of a multiple-input-single-output

(MISO) system we have to use as many output values as input of the inversion scheme as many input parameters are to be inverted.

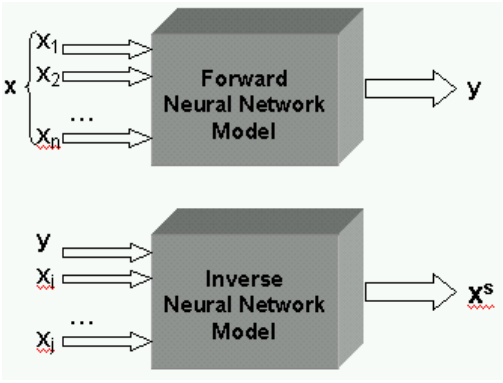


Fig. 5.7 Forward and inverse neural network models

$$\underline{x} = \{x_1,...,x_n\},\ \underline{x}^s \subseteq \underline{x} \equiv \{x_1,...,x_n\} \setminus \{x_i,...,x_j\}$$

5.7 Illustrative examples

In this Section, to illustrate the proposed iterative inversion method, two simple examples are presented. (For more examples see [S37], [S36], [S49], [S56], [S69]). The first one is a three input one output forward fuzzy model. The purpose of this model is to describe the frequency, sound intensity and age dependency of the human hearing system [18]. The output of the model is the sound intensity felt by the person, i.e., the subjective intensity. The inversion of this fuzzy model might be interesting if the input sound intensity is to be estimated from known subjective intensity, frequency, and age information.

The inputs are represented by Gaussian shape membership functions: five-five sets for the frequency and the input intensity, respectively, and another three to represent the age. The characterization of the output is solved by five triangular shape sets (see Fig. 5.8). The rule base of the model consists of 49 rules which is a linguistic equivalent of the widely known, measurement-based plots describing the human hearing system (see Table 5.1). Fig. 5.9 shows the obtained surface as a function of frequency and sound intensity at age of 60 years. The inverse was calculated by two methods. The proposed iterative gradient method was operated with μ=1 and termination condition at $E = 10^{-4}$ or 50 iterations. The error surface given in Fig. 5.10 is defined as the absolute difference of the original and the calculated values of the input variable objective intensity ($|x - \hat{x}(n)|$). The number of iterations can be kept at a relatively low level as it is illustrated by Fig. 5.11. The inversion was carried out also by using GA based inverse search. The obtained results (an average of ten independent runs) for the error surface and for the number of evaluations are shown in Figs. 5.12-5.13.

In the second example (Ex2) the inversion is carried out over a relatively different surface containing several local minima. The five-five Gaussian shape sets for the input variables and the five triangular shape output sets of the two input one output forward fuzzy model are shown on Fig. 5.14. The rule base of the model consists of 25 rules as is given by Table 5.2. The obtained surface can be followed on Fig. 5.15. As it is illustrated by Fig. 5.16 the gradient based iteration suffers from the local minima (see also Fig. 5.17 for the number of iterations). The GA based iteration avoids this problem (see Figs. 5.18-5.19).

To keep the computational complexity on a lower level the proposed combined method can be applied. Fig. 5.20 shows the obtained error surface, while Fig. 5.21 the number of evaluations.

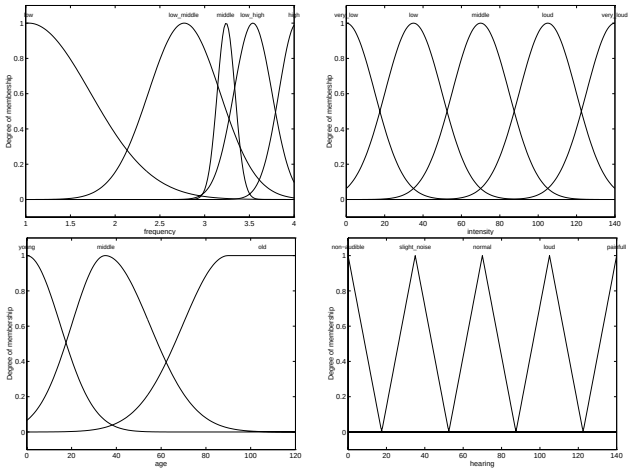


Fig. 5.8 Input and output fuzzy sets for the frequency, intensity, age, and hearing

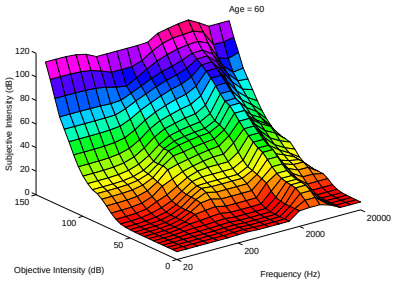


Fig. 5.9 The obtained surface as a function of frequency and sound intensity at age of 60 years

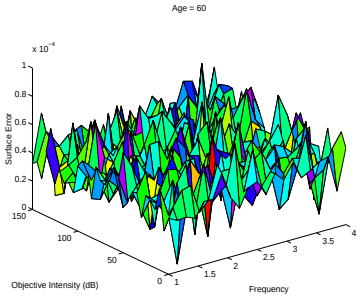


Fig. 5.12 The error surface (GA method)

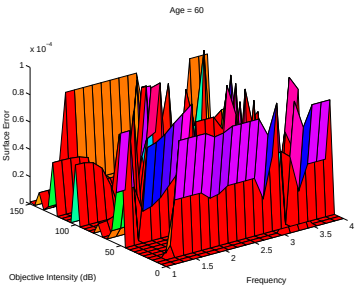


Fig. 5.10 The error surface (gradient method)

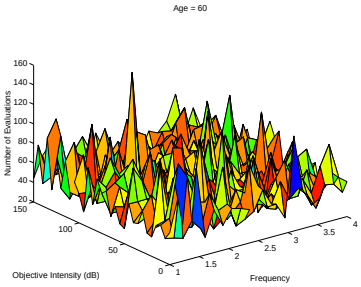


Fig. 5.13 The number of evaluations (GA method)

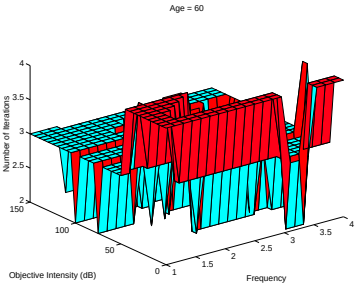


Fig. 5.11 The number of iterations (gradient method)

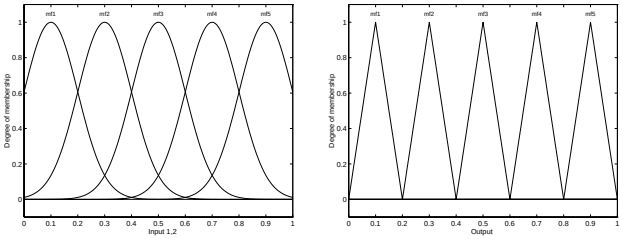


Fig.5.14 Ex2: input fuzzy sets for x_1 and x_2(left); output fuzzy sets (right)

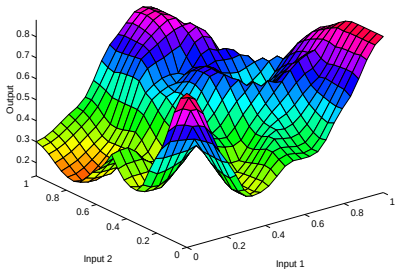


Fig. 5.15 Ex2: The obtained surface

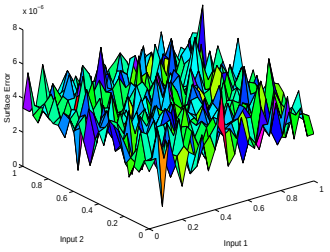


Fig. 5.18 Ex2: The error surface (GA method)

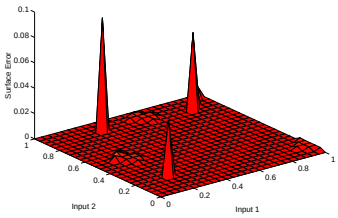


Fig. 5.16 Ex2: The error surface (gradient method)

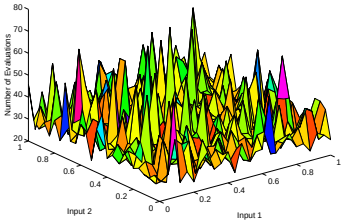


Fig. 5.19 Ex2: The number of evaluations (GA method)

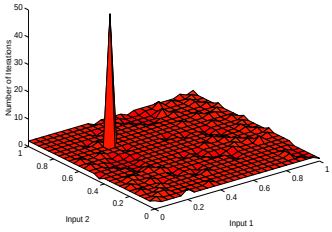


Fig. 5.17 Ex2: The number of iterations (gradient method)

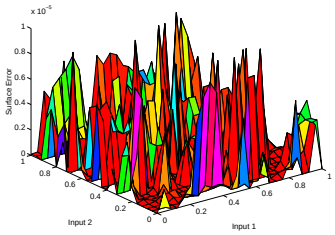


Fig. 5.20 Ex2: The error surface (combined method)

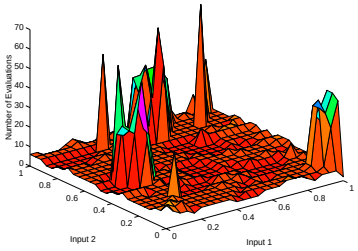


Fig. 5.21 Ex2: The number of evaluations (combined method)

Table 5.1 Rules of hearing fuzzy-model

Hearing (Age = Young)		Objective intensity				
		Very low	Low	Middle	Loud	Very loud
Frequency	Low	H1	H1	H2	H3	H5
	Low middle	H1	H2	H3	H4	H5
	Middle	H2	H3	H4	H5	H5
	Low high	H1	H2	H3	H4	H5
	High	H1	H1	H2	H3	H5

Hearing (Age = Middle)		Objective intensity				
		Very low	Low	Middle	Loud	Very loud
Frequency	Low	H1	H1	H1	H2	H5
	Low middle	H1	H1	H2	H3	H5
	Middle	H1	H2	H3	H4	H5
	Low high	H1	H1	H2	H3	H5
	High	H1	H1	H1	H2	H5

Hearing (Age = Old)		Objective intensity				
		Very low	Low	Middle	Loud	Very loud
Frequency	Low	H1	H1	H1	H2	H5
	Low middle	H1	H1	H2	H3	H5
	Middle	H1	H1	H3	H4	H5
	Low high	H1	H1	H2	H3	H5
	High	H1	H1	H1	H2	H5

H1 = non-audible, H2 = slight noise, H3 = normal, H4 = loud, H5 = painful

Table 5.2 Rules of example 2

Output		Input 2				
		mf1	mf2	mf3	mf4	mf5
Input 1	mf1	mf5	mf1	mf3	mf2	mf5
	mf2	mf1	mf3	mf2	mf4	mf5
	mf3	mf3	mf1	mf5	mf2	mf4
	mf4	mf1	mf3	mf4	mf5	mf2
	mf5	mf2	mf3	mf5	mf4	mf1

5.8 Conclusions

In this part an “on-line” iterative technique has been proposed to solve the inversion of fuzzy and neural network models for measurement and control applications. The derivation of this iterative technique is related to the state observer concept which proved to be very successful in the interpretation of the different techniques applied on this field. Additionally a step toward completely inverted models can also initiated by introducing state variable dynamic models combined with fuzzy logic based components. The proposed inversion scheme can be operated in anytime systems, as well, if as forward model SVD-based fuzzy or NN model is used. The main results presented in Part V are published e.g. in [S37], [S36], [S49], [S56], [S69], [S84], [S118], [S115].

Part VI Summary of the new results

Result 1: New methods in digital signal processing and data representation

1.1. I presented a new, low-complexity structure-pair based on the Walsh-Hadamard transformation for the synthesis and analysis of multisine signals. I extended this structure to the transformed domain orthogonal signal representations (having fast algorithms).

1.2 I introduced novel, low-complexity realizations of sliding-window and block-recursive filters and filter-banks. I demonstrated via simulations that these filters and filter-banks are able to follow slowly varying signals, and I proved analytically that the new (transformed domain) fast recursive sliding window realizations reduce significantly the processing time of signal samples in case of Overlap-Save and Overlap-Add Methods.

1.3 I constructed a new low-complexity implementation of the Discrete Fourier Transformation, which enables good estimation of the frequencies and amplitudes of the signal components after only a quarter of a complete signal period. Using this estimator, I introduced the Anytime Discrete Fourier Transformation (AnDFT) and its fast algorithm called Anytime Fast Fourier Transformation (AnFFT). I also showed how the new method helps to significantly reduce the delay of certain Adaptive Fourier Analyzer and other signal processing structures.

1.4 I introduced a novel, low-complexity overcomplete signal representation, which is appropriate also for anytime communication: it supports the adaptive adjustment of accuracy of the communicated signals based on the momentarily available capacity of the communication channel, thereby helping to find the optimal tradeoff between cost and accuracy.

1.5 I demonstrated that the applicability of the classical error- and error-propagation models of the measurement standards to nonlinear systems is strongly limited. I introduced novel non-classical data, error, and error-propagation models and demonstrated that they exhibit better behavior in case of nonlinear and non-monotonic systems than the classical ones. I also proposed adequate conversion methods for matching equivalent, classical and non-classical, qualitative and quantitative data and error models.

Result 2: New methods in digital image processing

2.1 I introduced a new, fuzzy-based corner detection method, which assigns also a fuzzy attribute ‘the strengths of cornerness’ to each detected corner. I showed that the new characterization of the corners fulfills the new requirements of pre-processing and offers enhanced support to further processing, such as automatic point correspondence matching of stereo-images.

2.2 I proposed new fuzzy-based methods for the separation of 'useful' and 'secondary' components (i.e. the enhancement and filtering) of visual information in digital images. I showed that by the application of this method, the complexity of the further automated processing (searching for patterns, object recognition, classification, etc.) can be significantly decreased, and their reliability can be improved.

2.3 I presented new high dynamic range (HDR) image reproduction techniques for the extraction of distorted or hidden information. I worked out several fuzzy-supported local anchor based and global tone-reproduction techniques. I also proposed a novel, low-complexity method to measure visual information density, which is used to optimally tune the information extraction algorithms.

2.4. I proposed a new image-synthesis method to generate high dynamic range images with optimal information density, based on a set of different exposure time color images taken from static scenes.

Result 3: Automatic 3D reconstruction and its application in intelligent vehicle system dynamics

3.1 I proposed a new, low complexity autonomous method for point correspondence matching in 2D stereo-images taken of static scenes.

3.2 Based on the previous result and a novel method of automatic camera calibration, I introduced a complete algorithm for automatic 3D reconstruction of static scenes based on 2D images from different viewpoints.

3.3 I worked out the structure and algorithms of an automatic, intelligent vehicle crash analysis system. I also realized the system software and validated it for the vehicle-wall collision experiments of standard crash-tests.

3.4 I proposed a new method for the approximation of high complexity neural networks via fuzzy weighting of small size 'local' neural network models of the system, thereby significantly decreasing the complexity and training time of the whole system model. Based on this hybrid neural network and the novel automatic 3D reconstruction method, I proposed a new technique for automatic energy equivalent speed (EES) based estimation of the collision speeds.

3.5 I worked out two new automatic methods for the estimation of the direction of impact. The first one is based on the displacement of the newly proposed 'energy-center' due to deformations. The second one applies surface fitting of the attached bodies for the determination of the impact direction.

Result 4: Generalization of anytime systems, anytime extension of fuzzy and neural network models

4.1 I introduced the concept of modular anytime framework, which enables the application of a set of non-recursive, not anytime algorithms and processes (based on different models of a system) in anytime mode of operation. I proposed a method for automatic generation of contract-based anytime algorithms in case of existing models of a system, which are not anytime. I introduced a novel description of the algorithms based on anytime error, which is often more efficient than contract-based descriptions. I also proposed a method to increase the adaptivity of existing anytime systems without complexity explosion (complexity-optimal refinement), i.e. to include new information.

4.2 I worked out two new compilation optimization algorithms (referred to as 'hierarchical' and 'output based incremental' algorithm), which fit unconditional and time-scaled quality, performance profiles, and scheduling functions more effectively than the existing methods.

4.3 Based on singular value decomposition (SVD)-based transformation, I gave a better upper bound of the anytime error of Product-Sum-Gravity-Singleton (PSGS) fuzzy systems with linear or nonlinear antecedent (input) sets than the known estimations. I generalized these results to 'nearly' PSGS systems, in which the antecedent sets are not in Ruspini partition. I also extended the method to the estimation of anytime error in generalized fuzzy neural networks.

4.4 I gave a constructive proof of the statement that arbitrary PSGS fuzzy system can be transformed to iterative anytime system.

4.5 I worked out the structure of a novel anytime compiler and scheduler (with the steps of active and passive monitoring processes), the basic framework of an anytime programming language (called ATDL), a development tool, and a (real-time) operating system. I determined the basic setup, the (descriptor, fitting, and scheduling) data-structures, the fundamental principles of operation (optimal fitting, scheduling), and the basic linguistic elements of the framework.

Result 5: Observer-based system inversion

5.1 I proposed a novel, globally convergent, low complexity method supported by genetic algorithms for model parameter tuning based on learning.

5.2 I introduced the observer-based iterative model inversion scheme and applied it to the observer-based, globally convergent inversion of static and dynamic MISO fuzzy and neural network systems. I also extended the inversion method towards anytime operation.

References

Part I

[1] Rabiner, L.R., B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J, 1975.

[2] Blahut, R.E., *Fast Algorithms for Digital Signal Processing*, Addison-Wesley Publ. Comp., Inc., 1985.

[3] Godfrey, K., *Perturbation Signals for System Identification*, Prentice-Hall Int. Ltd., UK, 1993.

[4] Hostetter, G.H., "Recursive Discrete Fourier Transformation," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 28, No.2, pp. 184-190, Apr. 1980.

[5] Péceli, G., "A Common Structure for Recursive Discrete Transforms," *IEEE Trans. on Circuits and Systems*, Vol. 33, pp. 1035-1036, Oct. 1986.

[6] Várkonyi-Kóczy, A.R., "A Recursive Fast Fourier Transformation Algorithm." *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 42, No. 9, pp. 614-616, Sep. 1995.

[7] Crochiere, R.E., L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J, 1983.

[8] Várkonyi-Kóczy, A.R., "A Composite Observer Structure for Adaptive Fourier Analysis." *In Proc. of the 5th IFAC Symposium on Adaptive Systems in Control and Signal Processing, ACASP'95*, Budapest, Hungary, June 14-16, 1995, pp. 171-176.

[9] Várkonyi-Kóczy, A.R., "Parallel Implementation of the Recursive Discrete Fourier Transformation." *In Proc. of the European Conference on Circuits Theory and Design, ECCTD'95*, Istambul, Turkey, Aug. 27-31, 1995, pp. 999-1002.

[10] Tadokoro, Y., Higuchi, T., "Discrete Fourier Transform Computation via the Walsh Transform," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 26, pp. 236-240, June 1978.

[11] Hostetter, G.H., "Recursive Discrete Fourier Transform with fading Memory," *In Proc. of the IEEE Int. Symp. on Circuits and Systems, ISCAS'83,* pp. 80-85, 1983.

[12] Bitmead, R.R., A.C. Tsoi, P.J. Parker, "A Kalman Filtering Approach to Short-Time Fourier Analysis," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 34, No. 6, pp. 1493-1501, Dec. 1986.

[13] Shynk, J.J., „Frequency-Domain and Multirate Adaptive Filtering", *IEEE Signal Processing Magazine*, pp. 15-37, Jan 1992.

[14] Chiang, H.-C., J.-C. Liu, „Fast Algorithm for FIR Filtering in the Transform Domain", *IEEE Transactions on Signal Processing*, Vol. 44. No. 1, pp. 126-129, Jan 1996.

[15] Jones, D.L., „Learning Characteristics of Transpose-Form LMS Adaptive Filters", *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 39., No. 10, pp. 745-749, Oct. 1992.

[16] Padmanabhan, M., K. Martin, G. Péceli, *Feedback-Based Orthogonal Filters*, Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.

[17] Personal discussion with L.A. Zadeh, 2005.

[18] Potter, R.W., "Tracking and Resampling Method and Apparatus for Monitoring the Performance of Rotating Machines," *United States Patent*, Patent Number 4912661, 1990.

[19] Nagy, F., "Measurement of Signal Parameters Using Nonlinear Observer Theory," *IEEE*

Trans. on Instrumentation and Measurement, Vol. 41, No.1, Feb. 1992, pp. 152-155.

[20] Ruzinsky, S.A. and E.T. Olsen, "L_1 and L_∞ minimization Via a Variant of Karmarkar's algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. 37, pp. 245-253, 1989.

[21] Mallat, S. and Z. Zhang, "Matching Pursuit in a Time-Frequency Dictionary," *IEEE Transactions on Signal Processing,* Vol. 41, pp. 3397-3415, 1993.

[22] Ahmed, N. and K.R. Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, New York, 1975.

[22] Goodwin, M.M., Adaptive Signal Models: Theory, Algorithms, and Audio Applications, PhD Thesis, University of California, Berkeley, USA, 1997.

[23] Chen, S.B., L.D. Donoho, "Examples of Basis Pursuit," Wavelet Applications in Signal Processing III, *Proc. of the SPIE*, pt.2, pp.564-574, 1995.

[24] Abdelmarek, N.N., "Solution of Minimum Time Problem and Minimum Fuel Problem for Discrete Linear Admissible Control Systems," *Int. J. Syst. Sci.*, Vol. 8, pp. 849-859, 1978.

[25] Mauris, G., L. Berrah, L. Foulloy, A. Haurat, "Fuzzy handling of measurement errors in instrumentation," *In Proc of the IEEE Instrumentation and Measurement Technology Conf.,* pp. 784-789, Ottawa, Canada, May 19-21, 1997.

[26] Bárdossy, Gy., J. Fodor, "The concept of geological uncertainties and new ways of the geomathematical evaluation," Special papers - Geological Society of America, No. 397, pp. 211-216, 2006.

[27] Kendall, M., A Stuart*, The advanced theory of statistics*, Ed. Griffin and Co., 1977.

[28] International Organization for Standardization: *Guide to the Expression of Uncertainty in Measurement*, First Edition, 1993.

[29] Combettes, P.L., and W.W. Edmonson, "What is a good estimate?," *Signal Processing VI: Theories and Applications*, pp.713-716, 1992.

[30] *MATLAB Statistics Toolbox User's Guide*, The MathWorks Inc., 1997.

[31] Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, 1965, pp. 338-353.

[32] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 3, No. 1, 1973, pp. 28-44.

[33] Klir, G.J., and T.J. Folger, *Fuzzy Sets, Uncertainty and Information*, Prentice Hall International Inc., Binghamton,1988.

[34] Chen, P., M. Nasu, T. Toyota, "Sequential Self-reorganization Method of Symptom Parameters and Identification Method of Membership Function for Fuzzy Diagnosis," *In Proc. of the IEEE Int. Conference on Fuzzy Systems*, pp. 433-440., Barcelona, Spain, July 1-5, 1997.

[35] Shen Q., and R. Leitch, "Fuzzy Qualitative simulation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 4, July/August 1993.

[36] D. Dubois and h. Prade, *Fuzzy sets, probability and measurement*, European Journal of Operational research, Vol. 40., 1989., pp. 135-154.

[37] M. Reha Civanlar and H. Joel Trussell*, Constructing Membership Functions Using Statistical Data*, Fuzzy Sets and Systems, Vol. 18, 1986, pp. 1-13.

[38] James F. Geer and George J. Klir*, Probability-Possibility Transformations: a Comparison*, Int. J. General Systems, Vol. 21, 1992, pp. 291-310.

[39] James F. Geer and George J. Klir, *A Mathematical Analysis of Information-preserving Transformations between Probabilistic and Possibilistic Formulations of Certainty*, Int. J. General Systems, Vol. 20, 1992, pp. 143-176.

[40] Takács, M., "Uninorm Operations on Type-2 Fuzzy Sets, *In Proc. of the 12th IEEE Int. Conference on Intelligent Engineering Systems, INES'2008,* Miami, USA, Feb.22-29, 2008.

Part II

[1] Russo, F., "Recent Advances in Fuzzy Techniques for Image Enhancement," *IEEE Transactions on Instrumentation and Measurement*, Vol. 47, No. 6, pp. 1428-1434, Dec. 1998.

[2] Xie, X., R. Sudhakar, and H. Zhuang, Corner detection by a cost minimization approach, Pattern Recognition, 26(8) (1993), 1235-1243.

[3] Velastin, S.A., J.H. Yin, A.C. Davies, M.A. Vicencio-Silva, R.E. Allsop, and A. Penn, Automated Measurement of Crowd Density and Motion using Image Processing, 7th IEE Int. Conf. on Road Traffic Monitoring and Control, 26-28 Apr. 1994, London, UK, pp. 127-132

[4] Debevec, P.E., C.J. Taylor, and J. Malik, Modeling and rendering architecture from photographs a hybrid geometry and image based approach, ISIGGRAPH, Aug. 1996.

[5] Grossmann, E., D. Ortin, and J. Santos-Victor, Single and multi-view reconstruction of structured scenes, 5th Asian Conf. on Computer Vision, Jan. 23–25, 2002, Melbourne, Australia.

[6] Harris, C. and M. Stephens, A combined corner and edge detector, 4th Alvey Vision Conf., 1988, pp. 189-192.

[7] Förstner, W., A feature based correspondence algorithm for image matching, Int. Arch. Photogramm. Remote Sensing, 26 (1986), 150-166.

[8] Smith, S.M and M. Brady, SUSAN - a new approach to low level image processing, Int. Journ. of Computer Vision, 23(1) (1997), 45-78.

[9] Russo, F., Fuzzy Filtering of Noisy Sensor Data, IEEE Instrumentation and Measurement Technology Conf., Brussels, Belgium, 4-6 June 1996, pp. 1281-1285.

[10] Catté, F., P.-L. Lions, J.-M. Morel, and T. Coll, Image selective smoothing and edge detection by nonlinear diffusion, SIAM Journ. on Numerical Analysis, 32(1992), 1895-1909.

[11] Felsberg, M., Low-Level Image Processing with the Structure Multivector, PhD Dissertation, Inst. of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, TR no. 0203, 2002.

[12] F. Long, H. Zhang, D. Dagan Feng, "Fundamentals of Content Based Image Retrieval," Proof. p. 26.

[13] J. Assfalg, A. D. Bimbo, P. Pala, "Using multiple examples for content-based retrieval," In Proc. of the Multimedia and Expo, ICME 2000, Vol.1, 2000, pp. 335-338.

[14] C. Lu, Y. Cao, D. Mumford, "Surface Evolution under Curvature Flows", Submitted for the special issue on Partial Differential Equations (PDE's) in Image Processing," Computer Vision, and Computer Graphics, p. 19, 2002.

[15] Gray, A. "The Gaussian and Mean Curvatures" and "Surfaces of Constant Gaussian Curvature," §16.5 and Ch. 21 in Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed. Boca Raton, FL: CRC Press, pp. 373-380 and 481-500, 1997.

[16] Adelson, E. H., A.P. Pentland, "The Perception of Shading and Reflectance," *In D. Knill and W. Richards (eds.), Perception as Bayesian Inference*, New York: Cambridge University Press, pp. 409-423, 1996.

[17] Adelson, E.H., "Lightness perception and lightness illusions," *In the Cognitive Neurosciences*, 2nd ed., Cambridge, MA: MIT Press, pp. 339-351, 2000.

[18] Palmer, S., Vision Science: Photons to Phenomenology, *The MIT Press, Ch. 3.3, Surface-Based Color Processing*, 1999.

[19] Scheel, A., M. Stamminger, H.-P. Seidel, "Tone Reproduction for Interactive Walkthroughs," *EUROGRAPHICS'2000*, Saarbrücken, Germany, Vol. 19, , No. 3, 2000.

[20] Kawahito, S., "An ultimate dynamic range imaging device – from star light to sun light," *In Proc. of the Inter-Academia'2005*, Sep. 19-22, 2005, Wuppertal, Germany, Vol. 1, pp.105-113.

[21] Reinhard, E., M. Stark, Shirley, P.J. Ferwerda, "Photographic Tone Reproduction for Digital Images," *In Proc. of the 29th Annual Conf. on Computer Graphics and Interactive Techniques*, 2002, San Antonio, Texas, pp. 267 – 276.

[22] Ukovich, A., G. Impoco, G. Reamponi, "A tool based on the co-occurance matrix to measure the performance of dynamic range reduction algorithms," *In Proc. of the IEEE Int. Workshop on Imaging Systems and Techniques, IST'2005*, May 13, 2005, pp. 36-41.

[23] Gilchrist, A., C. Kossyfidis, F. Bonato, T. Agostini, J. Cataliotti, X. Li, B. Spehar, V. Annan, E. Economou, "An anchoring theory of lightness perception," *Psychol. Rev*., Vol. 106, No. 4, pp. 795-834, Oct. 1999.

[24] Rudd, M. E., I.K. Zemach, "The highest luminance anchoring rule in lightness perception," *Journal of Vision*, Vol. 3, No. 9, 56a, 2003.

[25] Theiler, J., G. Gisler, "A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation," *In Proc. SPIE* 3159., 1997, pp. 108-118.

[26] Krawczyk, G., K. Myszkowski, H.P. Seidel, "Lightness Perception in Tone Reproduction for High Dynamic Range Images," *In Proc. of EUROGRAPHICS '05 (Computer Graphics Forum*, *vol. 24)*, 2005.

[27] Ward, G., "A contrast-based scalefactor for luminance display," *In Graphics Gems IV*, P. Heckbert, Ed. Academic Press, Boston, 1994, pp. 415–421.

[28] Lou Haskell's photos, www.easyhdr.com

Part III

[1] Taylor, C., P. Debevec, J. Malik, "Reconstructing Polyhedral Models of Architectural Scenes from Photographs," *Computer Vision - ECCV'96*, Lecture Notes in Computer Science, Vol. 1065, No. II, pp 659-668, 1996.

[2] Hartley, R., "Euclidean reconstruction from uncalibrated views," in J.L. Mundy, A. Zisserman, and D. Forsyth (eds.), *Applications of Invariance in Computer Vision*, Lecture Notes in Computer Science, Vol. 825, Springer-Verlag, pp. 237-256, 1994.

[3] Hartley, R., A. Zisserman, "*Multiple View Geometry in Computer Vision*," Cambridge University Press, 2000.

[4] Zhang, Z., R. Deriche, O. Faugeras, Q.-T. Luong, "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry," *Artificial Intelligence*, 1995, pp. 87-119.

[5] Faugeras, O., T. Vieville, E. Theron, J. Vuillemin, B. Hotz, Z. Zhang, L. Moll, P. Bertin, H. Mathieu, P. Fua, G. Berry, C. Proy, "*Real time correlation-based stereo: algorithm, implementations and applications,*" Research Report 2013, INRIA Sophia-Antipolis, France, August 1993, p. 45.

[6] Lee, C.-Y., D.B. Cooper, D. Keren, "Computing Correspondence Based on Region and Invariants without Feature Extraction and Segmentation," *in Proc. CVPR'93*, New York, USA, 1993, pp. 655-656.

[7] Russo, F., "Fuzzy Filtering of Noisy Sensor Data," *In Proc. of the IEEE Instrumentation and Measurement Technology Conference*, Brussels, Belgium, 4-6 June 1996, pp. 1281-1285.

[8] Russo, F., "Edge Detection in Noisy Images Using Fuzzy Reasoning," IEEE Transactions on Instrumentation and Measurement, Vol. 47, No. 5, Oct. 1998, pp. 1102-1105.

[9] Russo, F., "Recent Advances in Fuzzy Techniques for Image Enhancement," *IEEE Transactions on Instrumentation and Measurement*, Vol. 47, No. 6, Dec. 1998, pp. 1428-1434.

[10] Rogers D. F., *Procedural elements for Computer Graphics,* McGraw Hill, New York, 1985.
[11] Xu, G., Z. Zhang, *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*, Kluwer Academic Publishers, Norwell, MA, 1996.
[12] O. Faugeras, "What can be seen in three dimensions with an uncalibrated stereo rig", *Computer Vision - ECCV'92*, Lecture Notes in Computer Science, Vol. 588, Springer-Verlag, pp. 563-578, 1992.
[13] Hartley, R., "In defence of the 8-point algorithm," *In Proc. of the 5th International Conference on Computer Vision,* Cambridge, Massachusetts, USA, pp. 1064-1070, June 1995.
[14] Bogdan, G., P. Meer, "Point Matching under Large Image Deformations and Illumination Changes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 6, June 2004, pp. 674-688.
[15] Happer A., M. Araszewski, *Practical Analysis Technique for Quantifying Sideswipe Collisions*, 1999.
[16] Bokor J., P. Michelberger, A. Keresztes, P. Várlaki, "Statistical identification of nonlinear vehicle vibrating structures," *IFAC Preprints on Identification and System Parameter Estimation*, Vol. 1 (9th IFAC/IFORS Symposium), Budapest, Hungary, 1991, pp. 358-362.
[17] Melander, A., "Finite element simulation of crash testing of laser welded joints," *Swedish Institute for Metals Research report,* no: IM-2000-062 (2000).
[18] Lenard, J.A., B. Hurley, P.D. Thomas, "The accuracy of CRASH3 for calculating collision severity in modern European cars," *In Proc. 16th International Conference on the Enhanced Safety of Vehicles*, May 31-June 4th, Windsor, Ontario, Canada, June 1998, 1242-1249, ISBN DOT HS 808 759
[19] Chen, H.F., C.B. Tanner, N.J. Durisek, D.A. Guenther, "Pole Impact Speeds Derived from Bilinear Estimations of Maximum Crush for Body-On- Frame Constructed Vehicles," Paper # 2004-01-1615, Society of Automotive Engineers, Warrendale, Pennsylvania, 2004.
[20] Mercedes Benz, "Die Bedentung der Energy Equivalent Speed (EES) für die Unfallrekonstruktion und die Ferletzungsmechanik," Mercedes-Benz Pkw, 1992.

Part IV

[1] Billings, S.A, "Identification of Nonlinear Systems – A Survey," *IEE Proc.* Vol. 127, Pt.D, No. 6, Nov. 1980, pp. 272-284.
[2] Klir, G.J., T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall International, Inc., 1988.
[3] Wang, L.X., *Adaptive Fuzzy Systems and Control*, Prentice-Hall International, Inc., New Jersey, USA, 1994.
[4] Babuska, R., J. Sousa, H.B. Verbruggen, "Model-Based Design of Fuzzy Control Systems," *In Proc. of the Third European Congress on Intelligent Techniques and Soft Computing EUFIT'95*, Aachen, Germany, pp. 837-841.
[5] Chen, S., S.A. Billing, W. Luo, "Orthogonal Least Squares Methods and Their Application to Non-Linear System Identification," *Int. Journal of Control*, Vol. 50, No. 5, 1989, pp. 1873-1896.
[6] Fuller, R., Neural Fuzzy Systems, Lecture Notes, Abo Akademi, Helsinki, Finland, 1997.
[7] Rauma, T., M. Kurki, P. Alahuhta, „An Approach of Using Fuzzy Control in Fault Diagnostics," *European Conf. on Fuzzy and Intelligent Systems, EUFIT'96*, Aachen, Germany, 1996.

[8] Russo, F., "Recent Advances in Fuzzy Techniques for Image Enhancement," *IEEE Trans. on Instrumentation and Measurement*, Vol. 47, No. 6, Dec. 1998, pp. 1428-1434.
[9] Várkonyi-Kóczy, A.R. and T.P. Dobrowiecki, "Imprecise Methods in Measurement," *In Proc. of the 1997 IEEE Instrumentation & Measurement Technology Conference, IMTC'97*, Ottawa, Canada, May 19-21, 1997, pp. 790-795.
[10] Mitchell, T., *Machine Learning*, McGraw Hill, New York, USA, 1997.
[11] Lei, K.F., Y. Yam, P. Baranyi, "Neuro-Fuzzy Based Experiments on a Shape Memory Allow Positioning System, *" American Control Conference*, Arlington, USA, 2001, pp. 3861-3865.
[12] Bhatnagar, R.K., L.N. Kanal, "Handling uncertain information: a review of numeric and non-numeric methods," In *Uncertainty in Artificial Intelligence*, Elsevier Science Publishers, 1986, pp. 3-26.
[13] Baron, C., J.-C. Geffroy, G. Motet ed., *Embedded System Applications*, Kluwer Academic Publishers, 1997.
[14] Zilberstein, S., „Using Anytime Algorithms in Intelligent systems," *AI Magazine*, Vol. 17, No. 3, 1996, pp. 73-83.
[15] Zilberstein, S., *Operational Rationality through Compilation of Anytime Algorithms*, PhD Thesis, 1993.
[16] Boddy, M., T.L. Dean, "Solving time-dependent planning problems," *In Proc. of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989, pp. 979-984.
[17] Hansen, E.A., S. Zilberstein, Monitoring the Progress of Anytime Problem-Solving. In Proc. of the Thirteenth National Conference on Artificial Intelligence, AAAI-96, 1996.
[18] Zilberstein, S., F. Charpillet, P. Chassaing, "Optimal Sequencing of Contract Algorithms," *Annals of Mathematics and Artificial Intelligence*, 2002.
[19] Garvey, A., V. Lesser, "Design-to-Time Scheduling with Uncertainty," *SIGART Bulletin*, Jan. 9, 1995.
[20] Grass, J., S. Zilberstein, "Anytime Algorithm Development Tools," UM-CS-1995-094.
[21] Simon, Gy., T. Kovácsházy, G. Péceli, Transient Management in Reconfigurable Control Systems, *Technical Report*, Budapest University of Technology and Economics, 2002.
[22] Péceli, G., T. Kovácsházy, "Transients in Reconfigurable Systems," *In Proc. Of the 1998 IEEE Instrumentation & Measurement Technology Conf., IMTC'98*, St. Paul, USA, May, 1998, pp. 919-922.
[23] Dorf, R.C., *Modern Control Systems*, Addison-Wesley Publ. Comp., USA, 1987.
[24] Yam, Y., "Fuzzy Approximation via Grid Sampling and Singular Value Decomposition," *IEEE Trans. on Systems, Men, and Cybernetics*, Vol. 27, No. 6, 1997, pp. 933-951.
[25] Takagi, T., M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Trans. on Systems, Men, and Cybernetics*, Vol. 15, 1985, pp. 116-132.
[26] Yam, Y., P. Baranyi, C.T. Yang, "Reduction of Fuzzy Rule Base Via Singular Value Decomposition," *IEEE Trans. on Fuzzy Systems*, Vol. 7., No. 2., Apr. 1999, pp.120-132.
[27] Larsen, P.M., "Industrial Application of Fuzzy Logic Control," *Int. Journal Man-Machine Studies*, Vol. 12, 1980, pp. 3-10.
[28] Baranyi, P., Y. Yang, "Singular value-based approximation with non-singleton support," *In Proc. of the Seventh Int. IFSA World Congress*, Prague, June 25-29, 1997, pp. 127-132.
[29] Baranyi P., Y. Yang, "Singular value-based approximation with non-singleton support," Seventh Int. IFSA World Congress, Prague, June 25-29, 1997, pp.127-132.

[30] Yen, J., L. Wang, "Simplifying Fuzzy Rule-based Models Using Orthogonal Transformation Methods," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 29: Part B, No. 1, Feb. 1999, pp. 13-24.
[31] Chen, J., R.J. Patton, *Robust Model Based Fault Diagnosis For Dynamic Systems,* Kluwer Academic Publishers, 1999, ISBN 0-7923-8411-3.
[32] Wang, O.H., K. Tanaka, M.F.P. Griffin, "Parallel distributed compensation of non-linear systems by Takagi and Sugeno fuzzy models," *In Proc. FUZZ-IEEE/FES'95*, 1995, pp. 531-538.
[33] Tikk, D., "On nowhere denseness of certain fuzzy controllers containing pre-restricted number of rules," *Tatra Mountain Math. Publ.,* 16, 1999.
[34] Baranyi P., K. Lei, Y. Yam, "Complexity reduction of singleton based neuro-fuzzy algorithm," *In Proc. of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 8-11, 2000, Nashville, USA, Vol. 4, pp. 2503-2508.
[35] Takács, O., I. Nagy, "Error-bound of the SVD Based Neural Networks," *In Proc. of the IFAC Symp. on Artificial Intelligence in Real-Time Control, AIRTC'2000*, Budapest, Hungary, Oct 2-4, 2000, pp. 139-144.
[36] Zilberstein, S., *Operational Rationality through Compilation of Anytime Algorithms*, PhD Thesis, 1993.
[37] Klement, E.P, L.T. Kóczy B. Moser, "Are fuzzy systems universal approximators?", *Int. Jour. General Systems*, Vol. 28, No. 2-3, 1999, pp. 259-282.
[38] Tanaka, K., Wang, H.O, *Fuzzy Control Systems Design and Analysis*, John Wiley & Sons, Inc. New York, 2001.
[39] Tanaka, K., T. Taniguchi, H.O. Wang, "Robust and Optimal Fuzzy Control: A Linear Matrix Inequality Approach," Proc. of the 1999 IFAC World Congress, Beijing, July 1999, pp. 213-218.

Part V

[1] Patton, R., P. Frank, R. Clark, *Fault Diagnosis in Dynamic Systems*, Prentice Hall International (UK) Ltd., 1989.
[2] Anderson, B.D.O., J.B. Moore, *Optimal Filtering*, Prentice-Hall, Inc., Englewood Cliffs, N.J. 1979.
[3] Boyd, S.P., C.H. Barratt, *Linear Controller Design*, *Limits of Performance*, Prentice Hall, Inc. Englewood Cliffs, N.J. 1991.
[4] Widrow, B., E. Walach, *Adaptive Inverse Control*, Prentice Hall, 1996.
[5] S.A. Billings, "Identification of Nonlinear Systems – A Survey," *IEE Proc*. Vol. 127, Pt.D, No. 6, Nov. 1980., pp. 272-284.
[6] Klir, G.J., T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall International, Inc., 1988.
[7] Babuska, R., J. Sousa, H.B. Verbruggen, "Model-Based Design of Fuzzy Control Systems," *Proc. of the Third European Congress on Intelligent Techniques and Soft Computing EUFIT'95*, Aachen, Germany, pp. 837-841.
[8] Baranyi, P., I. Bavelaar, L.T. Kóczy, A. Titli, "Inverse Rule Base of Various Fuzzy Interpolation Techniques," *Proc. of the 7th Int. Fuzzy Systems Association World Congress, IFSA'97*, June 25-29, 1997, Prague, pp. 121-126.
[9] Baranyi, P., P. Korondi, H. Hashimoto, M. Wada, "Fuzzy Inversion and Rule Base Reduction," *Proc. of the 1997 IEEE Int.. Conf. on Engineering Systems, INES'97*, Sep. 12-15, 1997, Budapest, Hungary, pp. 301-306.
[10] Baranyi, P., I.M. Bavelaar, R. Babuska, L.T. Kóczy, A. Titli, H.B. Verbruggen, "A Method to Invert a Linguistic Fuzzy Model," *Int. Journal of Systems Science,* 1998, Vol. 29, no. 7, pp. 711-721.

[11] Batur, C., A. Shrinivasan, Chieng-Chung Chan, "Inverse Fuzzy Model Controllers," *Proc. of the American Control Conf.*, June 1993, San Francisco, California, pp. 772-774.
[12] Jager, R., *Fuzzy Logic in Control*, PhD Thesis, TU Delft, 1995.
[13] Acton, F.S., *Numerical Methods That Work*, The Mathematical Association of America, Washington D.C., 1990.
[14] Whitley, D., *A Genetic Algorithm tutorial,* Technical report, Colorado State University, 1993.
[15] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading 1989.
[16] Ko, M-S., T-W. Kang, Ch-S. Hwang, Function Optimization Using an Adaptive Crossover Operator Based on Locality, *Engineering Applications of Artificial Intelligence*, Vol. 10, No. 6. pp. 519-524, 1997.
[17] Djurisic, A.B., Elite "Genetic Algorithms with Adaptive Mutations for Solving Continuous Optimization Problems – Application to Modeling of the Optical Constants of Solids," *Optics Communications,* 151, pp. 147-159, 1998.
[18] Moore, B.C.J., *An Introduction to the Psychology of Hearing*, Third Edition, Academic Press London, 1995.

Publications of the author

Books, book chapters

[S1] Balas, V., J. Fodor, A.R. Várkonyi-Kóczy (eds.), Soft Computing Based Modeling in Intelligent Systems (Ser. Studies in Computational Intelligence), Springer Verlag, 2008.

[S2] Várkonyi-Kóczy, A.R., "Discrete Orthogonal Transforms," in Horváth, G. (ed.), Parallel Digital Signal Processing, Miskolc, 1997, 49 p.

[S3] Baranyi, P., A.R. Várkonyi-Kóczy, P. Várlaki, P. Michelberger, and R.J. Patton, "Singular Value Based Model Approximation." In N. Mastorakis (ed.) Problems in Applied Mathematics and Computational Intelligence (Mathematics and Computers in Science and Engineering), World Scientific and Engineering Society Press, Danvers, 2001, pp. 119-124.

[S4] Takács, O. and A.R. Várkonyi-Kóczy, „Iterative Evaluation of Anytime PSGS Fuzzy Systems." In P. Sincak, J. Vascak, K. Hirota (eds.) Quo Vadis Machine Intelligence? - The Progressive Trends in Intelligent Technologies, (Ser. Advances in Fuzzy Systems – Applications and Theory, Vol. 21) World Scientific Press, Heidelberg, 2004, pp. 93-106.

[S5] Várkonyi-Kóczy, A.R, "Model Based Anytime Soft Computing Approaches in Engineering Applications." In Balas, V., J. Fodor, A.R. Várkonyi-Kóczy (eds.), Soft Computing Based Modeling in Intelligent Systems (Ser. Studies in Computational Intelligence), Springer Verlag, 2008, 28 p.

Journal papers

[S6] Várkonyi-Kóczy, A.R., "Efficient Polyphase DFT Filter Banks with Fading Memory." IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 44, No. 8, pp. 670-673, Aug. 1997.

[S7] Várkonyi-Kóczy, A.R., "Synchronized Multi-Sine Measurements via DSP Methods." IEEE Trans. on Instrumentation and Measurement, Vol. 46, No. 4, pp. 929-932, Aug. 1997.

[S8] Várkonyi-Kóczy, A.R., "Anytime Fourier Analysis," Periodica Polytechnica Ser. Electrical Engineering, Vol. 50, No. 3-4, pp.269-280, Nov.-Dec. 2006.

[S9] Várkonyi-Kóczy, A.R., "Fuzzy Logic Supported Corner Detection," Journal of Intelligent and Fuzzy Systems, Vol. 19, No. 1, pp. 41-50, Jan. 2008.

[S10] Várkonyi-Kóczy, A.R., "CASY - an intelligent car crash analysis system," International Journal of Advanced Intelligence Paradigms (IJAIP), Vol. 1, No. 1, pp. Jan. 2008.

[S11] Várkonyi-Kóczy, A.R., "Intelligent autonomous primary 3D feature extraction in vehicle system dynamics analysis: Theory and Application," Acta Polytechnika Hungarica, Vol. 5, No. 1, pp. 5-29, Jan. 2008.

[S12] Várkonyi-Kóczy, A.R., "State Dependant Anytime Control Methodology for Non-linear Systems," International Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 12, No. 2, pp. , March, 2008, in print.

[S13] Várkonyi-Kóczy, A.R., "A Universal Autonomous Robot Navigation Method," International Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 12, No. 2, pp. , March, 2008, in print.

[S14] Várkonyi-Kóczy, A.R., Gy. Simon, L. Sujbert, and M. Fék, "A Fast Filter-Bank for Adaptive Fourier Analysis." IEEE Trans. on Instrumentation and Measurement, Vol. 47, No. 5, pp. 1124-1128, Oct. 1998.

[S15] Román, Gy. and A.R. Várkonyi-Kóczy, "On the Consequences of the Complexity of Tasks in Measurement." Engineering, Vol. 4, No.1, pp. 53-62, March 1998.
[S16] Várkonyi-Kóczy, A.R., T. Kovácsházy, O. Takács, and Cs. Benedecsik, "Anytime Evaluation of Regression-Type Algorithms." International Journal of Advanced Computational Intelligence, Vol. 5, No. 1, pp. 2-7, Feb. 2001.
[S17] Takács, O. and A.R. Várkonyi-Kóczy, "Information Processing Based on Mixed – Classical and Fuzzy – Data Models." International Journal of Advanced Computational Intelligence, Vol. 5, No. 1, pp. 44-50, Feb. 2001.
[S18] Takács, O. and A.R. Várkonyi-Kóczy, "Soft Computing Tools in Anytime Systems (Szoft számítási eszközök anytime rendszerekben)," Hiradástechnika, Vol 16, No. 2, pp. 55-60, May 2001. (In Hungarian)
[S19] Várkonyi-Kóczy, A.R. and M. Fék, "Recursive Overcomplete Signal Representations." IEEE Trans. on Instrumentation and Measurement, Vol. 50, No. 6, pp. 1698-1703, Dec. 2001.
[S20] Baranyi, P., Y. Yam, A.R. Várkonyi-Kóczy, R.J. Patton, P. Michelberger, and M. Sugiyama, "SVD Based Complexity Reduction to TS Fuzzy Models," IEEE Trans. on Industrial Electronics, Vol. 49, No.2, pp. 433- 443, Apr. 2002.
[S21] Baranyi, P. and A.R. Várkonyi-Kóczy, "Adaptation of SVD Based Fuzzy Reduction via Minimal Expansion," IEEE Trans. on Instrumentation and Measurement, Vol. 51, No. 2, pp. 222-226, Apr. 2002.
[S22] Takács, O. and A.R. Várkonyi-Kóczy, "SVD Based Complexity Reduction of Rule Bases with Non-Linear Antecedent Fuzzy Sets," IEEE Trans. on Instrumentation and Measurement, Vol. 51, No. 2, pp. 217-221, Apr. 2002.
[S23] Baranyi, P., Y. Yam, A.R. Várkonyi-Kóczy, and R.J. Patton, "SVD Based Reduction to MISO TS Fuzzy Models," IEEE Trans. on Industrial Electronics, Vol. 50. No. 1, pp. 232-242, Feb. 2003.
[S24] Várkonyi-Kóczy, A.R., G. Samu, "Anytime System Scheduler for Insufficient Resource Availability," International Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 8, No. 5, pp. 488-494, Sep. 2004.
[S25] Várkonyi-Kóczy, A.R., A. Rövid, "Soft Computing Based Point Corresponding Matching for Automatic 3D Reconstruction," Acta Polytechnica Hungarica (Special Issue on Computational Intelligence), Vol. 2, No. 1, pp. 33-44, Jan. 2005.
[S26] Baranyi, P., A.R. Várkonyi-Kóczy, Y. Yam, and R.J.Patton "Adaption of TS Fuzzy Models without Complexity Expansion: HOSVD Based Approach," IEEE Trans. on Instrumentation and Measurement, Vol. 54, No. 1, pp. 52-60, Feb. 2005.
[S27] Takács, O. and A.R. Várkonyi-Kóczy, "Iterative-type Evaluation of PSGS Fuzzy Systems for Anytime Use," IEEE Trans. on Instrumentation and Measurement, Vol. 54, No. 1, pp. 391-397, Feb. 2005.
[S28] Baranyi, P., A.R. Várkonyi-Kóczy, TP Transformation Based Dynamic System Modeling for Nonlinear Control," IEEE Trans. on Instrumentation and Measurement, Vol. 54, No. 6, pp. 2191-2203, Dec. 2005.
[S29] Várkonyi-Kóczy, A.R., A. Rövid, P. Várlaki, „Fuzzy Based Brightness Compensation for High Dynamic Range Images," International Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 10, No. 4, pp. 549-554, July. 2006.
[S30] Várkonyi-Kóczy, A.R., A. Rövid, M.G. Ruano, "Soft Computing Based Car Body Deformation and EES Determination for Car Crash Analysis Systems," IEEE Trans. on Instrumentation and Measurement, Vol. 55, No. 6, pp. 2300-2308, Dec. 2006.
[S31] Várkonyi-Kóczy, A.R., A. Rövid, "High Dynamic Range Image Reproduction Methods," IEEE Trans. on Instrumentation and Measurement, Vol. 56, No. 4, pp. 1465-1472, August 2007.

[S32] Várkonyi-Kóczy, A.R., A. Rövid, Sz. Balogh, T. Hashimoto, Y. Shimodaira, High Dynamic Range Image Based on Multiple Exposure Time Synthetization, Acta Polytechnika Hungarica, Vol. 4, No. 1, pp. 5-15, 2007.

Conference papers published in proceedings

[S33] Várkonyi-Kóczy, A.R., "Multi-Sine Synthesis and Analysis via Walsh-Hadamard Transformation." In Proc. of the 1996 IEEE Int. Symp. on Circuits and Systems, ISCAS'96, Atlanta, USA, May 12-15, 1996, Vol 2, pp. 457-460.

[S34] Várkonyi-Kóczy, A.R., "Synchronized Multi-Sine Measurements via DSP Methods." In Proc. of the 1996 IEEE Instrumentation & Measurement Technology Conference, IMTC'96, Brussels, Belgium, June 4-6, 1996, pp. 1056-1060.

[S35] Várkonyi-Kóczy, A.R., "Characterization of the Synchronized Multi-Sine Analysis and Synthesis." In Proc. of the Int. Symp. on New Measurement and Calibration Methods of Electrical Quantities and Instruments, 8th TC-4 Symp., IMEKO TC4, Budapest, Hungary, Sep. 16-17, 1996, pp. 188-191.

[S36] Várkonyi-Kóczy, A.R., "Observer Based Iterative Fuzzy Model Inversion Techniques." In Proc. of the Int. Symp. of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 2-3, 2000, pp. 49-68.

[S37] Várkonyi-Kóczy, A.R., "Observer Based Fuzzy Model Inversion," In Proc. of the 2001 IEEE Int. Conference on Intelligent Engineering Systems, INES'2001, Helsinki, Finland, Sep. 16-18, 2001, pp. 17-22 (invited keynote paper).

[S38] Várkonyi-Kóczy, A.R., "Fuzzy Logic Supported Frequency Range Estimation for Adaptive Fourier Analysis," 2nd IEEE International Workshop on Soft Computing Applications, SOFA'2007, Gyula, Hungary and Oradea, Romania, Aug. 21-23, 2007, pp. 25-31.

[S39] Várkonyi-Kóczy, A.R., "Autonomous 3D Model Reconstruction and Its Intelligent Applications in Vehicle System Dynamics Part I: Theory," 5th IEEE Int. Symposium on Intelligent Systems and Informatics, SISY'2007, Subotica, Serbia, Aug.24-24, 2007, pp. 13-18. (invited plenary paper)

[S40] Várkonyi-Kóczy, A.R., "Autonomous 3D Model Reconstruction and Its Intelligent Applications in Vehicle System Dynamics Part II: Applications," 5th IEEE Int. Symposium on Intelligent Systems and Informatics, SISY'2007, Subotica, Serbia, Aug.24-24, 2007, pp. 19-24. (invited plenary paper)

[S41] Várkonyi-Kóczy, A.R., "Anytime Fuzzy Fast Fourier Transformation," In Proc. of the 12th IEEE Int. Conference on Intelligent Engineering Systems, INES'2008, Miami, USA, Feb.22-29, 2008, accepted.

[S42] Várkonyi-Kóczy, A.R., "Anytime Adaptive Fourier Transformation," 2008 IEEE International Instrumentation and Measurement Technology Conference, I^2MTC'2008 Victoria, Canada, May 12-15, 2008, accepted.

[S43] Várkonyi-Kóczy, A.R., T.P. Dobrowiecki, and B. Pataki, "Imprecise Computation Methods in Measurement." In Proc. of the 5th Biennial Baltic Electronics Conference, BEC'96, Tallinn, Estonia, Oct. 7-11, 1996, pp. 149-152.

[S44] Várkonyi-Kóczy, A.R. and S. Theodoridis, "Fast Sliding Transforms in Transform-Domain Adaptive Filtering." In Proc. of the 1997 IEEE Int. Conference on Acoustics, Speech, and Signal Processing, ICASSP'97, Munich, Germany, Apr. 20-24, 1997, Vol. 3, pp. 2009-2012.

[S45] Péceli, G. and A.R. Várkonyi-Kóczy, "Block-Recursive Filters and Filter Banks." In Proc. of the 1997 IEEE Int. Conference on Acoustics, Speech, and Signal Processing, ICASSP'97, Munich, Germany, Apr. 20-24, 1997, Vol. 3, pp. 2001-2004.

[S46] Várkonyi-Kóczy, A.R. and T.P. Dobrowiecki, “Imprecise Methods in Measurement.” In Proc. of the 1997 IEEE Instrumentation & Measurement Technology Conference, IMTC'97, Ottawa, Canada, May 19-21, 1997, pp. 790-795.

[S47] Várkonyi-Kóczy, A.R., Gy. Simon, and M. Fék, “Improved Multi-Sine Synthesis and Analysis.” In Proc. of the Int. Symp. on Electrical Instruments in Industry, IMEKO TC-4, Glasgow, United Kingdom, Sep. 8-9, 1997, pp. 13-16.

[S48] Várkonyi-Kóczy, A.R., T.P. Dobrowiecki, and G. Péceli, “Measurement Uncertainty: A Soft Computing Approach.” In Proc. of the 1997 IEEE Int. Conference on Intelligent Engineering Systems, INES'97, Budapest, Hungary, Sep. 15-17, 1997, pp. 485-490.

[S49] Várkonyi-Kóczy, A.R., G. Péceli, T.P. Dobrowiecki, and T. Kovácsházy, “Iterative Fuzzy Model Inversion.” In Proc. of the 1998 IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'98, Anchorage, Alaska, USA, May 5-9, 1998, Vol. 1, pp. 561-566.

[S50] Várkonyi-Kóczy, A.R. and Gy. Román, “The Role and Power of Soft Computing in Complex Measurement Sytems.” In CD-ROM Proc. of the 1998 World Automation Congress, WAC’98, Anchorage, Alaska, USA, May 9-14, 1998, ISBN: 1-889335-07-X, pp. ISSCI-005.1-6.

[S51] Várkonyi-Kóczy, A.R., Gy. Simon, L. Sujbert. and M. Fék, “A Fast Filter-Bank for Adaptive Fourier Analysis.” In Proc. of the 1998 IEEE Instrumentation & Measurement Technology Conference, IMTC'98, St. Paul, Minnesota, USA, May 18-20, 1998, Vol. 2, pp. 915-918.

[S52] Várkonyi-Kóczy, A.R. and T. Kovácsházy, “Anytime Algorithms in Embedded Signal Processing Systems.” In Proc. of the IX. European Signal Processing Conference, EUSIPCO-98, Rhodes, Greece, Sep. 8-11, 1998, Vol. 1, pp. 169-172.

[S53] Várkonyi-Kóczy, A.R., G. Péceli, T.P. Dobrowiecki, and O. Takács, “Measurement Technology of Intelligent Systems.” In Proc. of the 1998 IEEE Int. Conference on Intelligent Engineering Systems, INES'98, Vienna, Austria, Sep. 17-19, 1998, pp. 281-284.

[S54] Takács, O. and A.R. Várkonyi-Kóczy, “Some Aspects of Representing Uncertainty in Nonlinear Systems.” In Proc. of the Int. Symp. on Intelligent Systems in Control and Measurement, INTCOM'98, Miskolc, Hungary, Nov. 21-27, 1998, pp. 168-176.

[S55] Takács, O. and A.R. Várkonyi-Kóczy, “Fuzzy Handling of Uncertainty in Nonlinear Systems.” In Proc. of the Joint EUROFUSE-SIC'99 Conference, Budapest, Hungary, May 25-28, 1999, pp. 22-27.

[S56] Várkonyi-Kóczy, A.R., A. Álmos, and T. Kovácsházy, “Genetic Algorithms in Fuzzy Model Inversion.” In Proc. of the 8th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'99, Seoul, Korea, Aug. 22-25, 1999, Vol. 3, pp. 1421-1426.

[S57] Baranyi, P., Y. Yam, Ch-T. Yang, and A.R. Várkonyi-Kóczy, “Complexity Reduction of a Rational General Form.” In Proc. of the 8th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'99, Seoul, Korea, Aug. 22-25, 1999, Vol. 1, pp. 366-371.

[S58] Várkonyi-Kóczy, A.R., T. Kovácsházy, and Gy. Román, “Anytime Algorithms in Embedded Signal Processing,.” In Proc. of the Jubilee Int. Conference, Budapest, Hungary, Sep. 1-2, 1999, pp. 65-68.

[S59] Várkonyi-Kóczy, A.R., T. Kovácsházy, O. Takács, and Cs. Benedecsik, “Anytime Evaluation of Regression-Type Algorithms.” In Proc. of the IEEE Int. Workshop on Intelligent Signal Processing, WISP’99, Budapest, Hungary, Sep. 4-7, 1999, pp. 34-38.

[S60] Takács, O. and A.R. Várkonyi-Kóczy, “Information Processing Based on Mixed – Classical and Fuzzy – Data Models.” In Proc. of the IEEE Int. Workshop on Intelligent Signal Processing, WISP’99, Budapest, Hungary, Sep. 4-7, 1999, pp. 23-28.

[S61] Sujbert, L., Gy. Simon, and A.R. Várkonyi-Kóczy, “An Improved Adaptive Fourier Analyzer.” In Proc. of the IEEE Int. Workshop on Intelligent Signal Processing, WISP’99, Budapest, Hungary, Sep. 4-7, 1999, pp. 182-187.

[S62] Baranyi, P., Y. Yam, Ch-T. Yang, and A.R. Várkonyi-Kóczy, “Practical Extension of the SVD Based Reduction Technique for Extremely Large Fuzzy Rule Bases.” In Proc. of the

IEEE Int. Workshop on Intelligent Signal Processing, WISP'99, Budapest, Hungary, Sep. 4-7, 1999, pp. 29-33.

[S63] Fék, M., A.R. Várkonyi-Kóczy, and J-M. Boucher, "Reduction of the Calculational Delay in the OLA Method." In Proc. of the Gretsy'99, Vannes, France, Sep. 13-17, 1999, pp. 91-95. (In French)

[S64] Takács, O. and A.R. Várkonyi-Kóczy, "The Use of Mixed – Classical and Fuzzy – Data Models in Intelligent Systems." In Proc. of the Int. Symp. on Intelligent Systems in Control and Measurement, INTCOM'99, Budapest, Hungary, Oct. 9-13, 1999, pp. 68-73.

[S65] Baranyi, P. and A.R. Várkonyi-Kóczy, "Adaption of SVD Reduction in Restricted Rule Base Size," In Proc. of the 11th Soft Science Workshop, SOFT, Kanazawa, Japan, Apr. 2000, pp. 32-35.

[S66] Várkonyi-Kóczy, A.R. and M. Fék, "Recursive Overcomplete Signal Representations." In Proc. of the 2000 IEEE Instrumentation and Measurement Technology Conference, IMTC/2000, Baltimore, Maryland, USA, May 1-4, 2000, pp.1090-1094.

[S67] Baranyi, P., Y. Yam, Ch-T. Yang, and A.R. Várkonyi-Kóczy, "SVD Based Reduction for Subdivided Rule Bases." In Proc. of the 9th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2000, San Antonio, Texas, May 7-10, 2000, pp. 712-716.

[S68] Várkonyi-Kóczy, A.R., T. Kovácsházy, O. Takács, and Cs. Benedecsik, "Anytime Algorithms in Intelligent Measurement and Control." In CD-ROM Proc. of the 2000 World Automation Congress, WAC 2000, Maui, USA, June 11-16, 2000, ISBN: 1-889335-10-X, pp. ISIAC-156.1-6.

[S69] Takács, O. and A.R. Várkonyi-Kóczy, "Fuzzy Tools in Anytime Systems." In Proc. of the 2000 IEEE Int. Conference on Intelligent Engineering Systems, INES'2000, Portoroz, Slovenia, Sep. 17-19, 2000, pp. 315-320.

[S70] Takács, O. and A.R. Várkonyi-Kóczy, "Anytime Soft Computing Methods for Intelligent Measurement, Diagnosis, and Control." In Proc. of the IFAC Symp. on Artificial Intelligence in Real-Time Control, AIRTC'2000, Budapest, Hungary, Oct 2-4, 2000, pp. 163-168.

[S71] Baranyi, P., A.R. Várkonyi-Kóczy, P. Várlaki, P. Michelberger, and R.J. Patton, "Singular Value Based Model Approximation," In. Proc. of the 2001 WSES Int. Conference on Fuzzy Sets and Fuzzy Systems, FSFS'01, Puerto de la Cruz, Spain, Febr. 11-15, 2001, pp. 5451-5456.

[S72] Várkonyi-Kóczy, A.R., A. Ruano, P. Baranyi, and O.Takács, "Anytime Information Processsing Based on Fuzzy and Neural Network Models," In Proc. of the 2001 IEEE Instrumentation and Measurement Technology Conference, IMTC/2001, Budapest, Hungary, May 21-23, 2001, pp. 1247-1252.

[S73] Takács, O. and A.R. Várkonyi-Kóczy, "Error-Bound for the Non-Exact SVD-Based Reduction of the Generalized Type Hybrid Neural Networks with Non-Singleton Consequents," In Proc. of the 2001 IEEE Instrumentation and Measurement Technology Conference, IMTC/2001, Budapest, Hungary, May 21-23, 2001, pp. 1607-1613.

[S74] Benedecsik, Cs., and A.R. Várkonyi-Kóczy, "Structure and Frequency Adaptation of the Recursive Adaptive Fourier Analyzer Based on the Walsh-Hadamard Transformation," In Proc. of the 2001 IEEE Instrumentation and Measurement Technology Conference, IMTC/2001, Budapest, Hungary, May 21-23, 2001, pp. 1148-1153.

[S75] Baranyi, P. and A.R. Várkonyi-Kóczy, "Adaption of SVD Based Fuzzy Reduction via Minimal Expansion," In Proc. of the IEEE Int. Workshop on Intelligent Signal Processing, WISP'2001, Budapest, Hungary, May 24-25, 2001, pp. 171-176.

[S76] Takács, O. and A.R. Várkonyi-Kóczy, "SVD Based Complexity Reduction of Rule Bases with Non-Linear Antecedent Fuzzy Sets," In Proc. of the IEEE Int. Workshop on

Intelligent Signal Processing, WISP'2001, Budapest, Hungary, May 24-25, 2001, pp. 183-187.

[S77] Várkonyi-Kóczy, A.R., M. Sugiyama, and H. Asai, "Complexity Reduction to Non-Singleton Fuzzy-Neural Network," Joint 9th IFSA World Congress and 20th NAFIPS International Conference, IFSA / NAFIPS 2001, Vancouver, Canada, July 25-28, 2001, pp. 2523-2528.

[S78] Baranyi, P., A.R. Várkonyi-Kóczy, Y.Yam, and P.Michelberger, "HOSVD based Computational Complexity Reduction of TS Fuzzy Models," Joint 9th IFSA World Congress and 20th NAFIPS International Conference, IFSA / NAFIPS 2001, Vancouver, Canada, July 25-28, 2001, pp. 2482-2485.

[S79] Baranyi, P., A.R. Várkonyi-Kóczy, Y. Yam, P. Várlaki, and P.Michelberger, "An Adaptation Technique to SVD Reduced Rule Bases," Joint 9th IFSA World Congress and 20th NAFIPS International Conference, IFSA / NAFIPS 2001, Vancouver, Canada, July 25-28, 2001, pp. 2488-2493.

[S80] Fék, M., A.R. Várkonyi-Kóczy, and J-M. Boucher, "Joint Speech and Audio Compression Combining Sinusoidal Modeling and Wavelet Packets, In Proc. of the EUROSPEECH'2001, Aalborg, Denmark , Sep. 3-7, 2001, pp. 2311-2314.

[S81] Baranyi, P., A.R. Varkonyi-Koczy, C. J. Lopez-Toribio and R J Patton, "Fuzzy Model Approximation and its SVD Reduction," In Proc. of the European Control Conference, ECC2001, Porto, Portugal Sep. 4-7, 2001, pp. 7-12. (invited paper).

[S82] Takács, O. and A.R. Várkonyi-Kóczy, "SVD Based Fuzzy and Neuro Systems for Anytime Use," In Proc. of the Int. Symp. of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 12-13, 2001, pp. 59-73.

[S83] Takács, O., A.R. Várkonyi-Kóczy, and P. Várlaki, "Non-exact Complexity Reduction of Generalized Neuro-Fuzzy Networks," In Proc. of the 10th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2001, Melbourne, Australia, Dec. 2-5, 2001, pp. 980-983.

[S84] Várkonyi-Kóczy, A.R. and O. Takács, "Anytime Extension of the Iterative Fuzzy Model Inversion," In Proc. of the 10th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2001, Melbourne, Australia, Dec. 2-5, 2001, pp. 976-979.

[S85] Baranyi, P., A.R. Várkonyi-Kóczy, Y. Yam, and P. Michelberger, "HOSVD and LMI Based Methodology in the Control of Nonlinear Systems," In Proc. of the 4th Int. Conference on Non-linear Problems in Aviation and Aerospace, ICNPAA'02, FIT, Florida, USA 2002, pp. 43-50.

[S86] Baranyi, P. and A.R. Várkonyi-Kóczy, "Adaption without Rule Base Size Expansion: HOSVD Based Approach," In Proc. of the 2002 IEEE Instrumentation and Measurement Technology Conference, IMTC/2002, Anchorage, USA, May 21-23, 2002, pp. 221-226.

[S87] Várkonyi-Kóczy, A.R., P. Baranyi, and A.E. Ruano, S. Győri, Z.Petres, P. Légrády, "SVD Based Modeling of Nonlinear Systems," In Proc. of the 2002 IEEE Instrumentation and Measurement Technology Conference, IMTC/2002, Anchorage, USA, May 21-23, 2002, pp. 887-892.

[S88] Takács, O. and A.R. Várkonyi-Kóczy, "Iterative-type Evaluation of PSGS Fuzzy Systems for Anytime Use," In Proc. of the 2002 IEEE Instrumentation and Measurement Technology Conference, IMTC/2002, Anchorage, USA, May 21-23, 2002, pp. 233-238.

[S89] Takács, O. and A.R. Várkonyi-Kóczy, "Improved Error-bound for the SVD based complexity reduction," In Proc. of the 2002 IEEE Int. Conference on Intelligent Engineering Systems, INES'2002, Opatija, Croatia, May 26-28, 2002, pp. 215-218.

[S90] Fék, M., A.R. Várkonyi-Kóczy, and J-M. Boucher, "Comparing Scalar and Z_n Lattice Based Encoding of Wavelet Coefficients in Sinusoidal Plus Wavelet Packet Coding," EUSIPCO'2002, Toulouse, France, Sep. 3-6, 2002, Vol. I, pp. 401-404.

[S91] Várkonyi-Kóczy, A.R. and P. Baranyi, "SVD Based Modeling and Control of Nonlinear Systems," In Proc. of the 1st Int. Conf. On Global Research and Education, Inter Akademia'2002, Sep. 23-26, 2002, Bratislava, Slovakia, pp. 160-163.

[S92] Várkonyi-Kóczy and A.R., P. Baranyi, "Soft Computing Based Modeling of Nonlinear Systems," 1st Hungarian-Korean Symposium on Soft Computing and Computational Intelligence, Budapest, Hungary, Oct 2-4, 2002, pp. 139-148.

[S93] Várkonyi-Kóczy, A.R., P. Baranyi, and O. Takács, "SVD Based Anytime Modeling and Control of Nonlinear Systems," In Proc. of the 3rd Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 14-15, 2002, pp. 135-146.

[S94] Várkonyi-Kóczy, A.R., P. Baranyi, P. Várlaki, and L. Kiss, "Model Based Anytime Control of Complex Systems," In Proc. of the 7th IEEE Int. Conference on Intelligent Engineering Systems, INES'2003, Assiut-Luxor, Egypt, March 4-6, 2003, pp. 394-399.

[S95] Várkonyi-Kóczy, A.R., P. Baranyi, and R.J. Patton, "Anytime Fuzzy Modeling Approach for Fault Detection Systems", In Proc. of the 2003 IEEE Instrumentation and Measurement Technology Conference, IMTC/2003, Vail, USA, May 20-22, 2003, pp. 1611-1616.

[S96] Várkonyi-Kóczy, A.R., P. Baranyi, L. Kiss, and P. Várlaki, "State Dependant Anytime Control Methodology for Prototypical Aeroelastic Wing Section with Structural Non-linearity," In CD-ROM Proc. of the 2003 IEEE Int. Symposium on Industrial Electronics, ISIE'2003, Rio de Janeiro, Brazil, June 9-12, 2003, ISBN: 0-7803-7912-8, pp. BF-003362.1-6.

[S97] Samu, G. and A.R. Várkonyi-Kóczy, "Intelligent Monitor for Anytime Systems," In Proc. of the IEEE Int. Symposium on Intelligent Signal Processing, WISP'2003, Budapest, Hungary, Sep. 4-6, 2003, pp. 277-282.

[S98] Takács, O. and A.R. Várkonyi-Kóczy, "SVD-based Complexity Reduction of "Near PSGS" Fuzzy Systems," In Proc. of the IEEE Int. Symposium on Intelligent Signal Processing, WISP'2003, Budapest, Hungary, Sep. 4-6, 2003, pp. 31-36.

[S99] Baranyi, P., and A.R. Várkonyi-Kóczy, "Asymptotic Stabilization of Aeroelastic Systems via TP Transformation," In Proc. of the IEEE Int. Symposium on Intelligent Signal Processing, WISP'2003, Budapest, Hungary, Sep. 4-6, 2003, pp. 143-148.

[S100] Várkonyi-Kóczy, A.R. and G. Samu, "Anytime System Scheduler for Insufficient Resource Availability," In Proc. of the IEEE International Conference on Computational Cybernetics, ICCC'2003, Siófok, Hungary, Aug. 29-31, 2003, pp. 205-210.

[S101] Várkonyi-Kóczy, A.R. and P. Baranyi, "Model Based Numerical Anytime Control Design of a Prototypical Aeroeleastic Wing Section," In Proc. of the 2nd Int. Conf. On Global Research and Education, Inter-Akademia'2003, Warsaw, Poland, Sep. 8-12, 2003, pp. 377-386.

[S102] Takács, O. and A.R. Várkonyi-Kóczy, "Soft-computing Tools in Anytime Environment," In Proc. of the 2nd Int. Conf. On Global Research and Education, Inter-Akademia'2003, Warsaw, Poland, Sep. 8-12, 2003, pp. 185-194.

[S103] Samu, G. and A.R. Várkonyi-Kóczy, "Intelligent Monitoring System for the Optimization of the Operation of Systems in Resource Insufficient Environment," In Proc. of the 4th Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 13-14, 2003, pp. 71-87.

[S104] Baranyi,P., P. Michelberger, and A.R. Várkonyi-Kóczy, "Numerical Control Design for Aeroelastic Systems," In Proc. of the 2nd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2004, Herlany, Slovakia, Jan. 16-17, 2004, pp. 43-50.

[S105] Rövid, A., A.R. Várkonyi-Kóczy, M.G. Ruano, P. Várlaki, P. Michelberger, "Soft Computing Based Car Body Deformation and EES Determination for Car Crash Analysis Systems," In Proc. of the 2004 IEEE Instrumentation and Measurement Technology Conference, IMTC/2004, Como, Italy, 18-20 May, 2004, pp. 1674-1679.

[S106] Rövid, A., A.R. Várkonyi-Kóczy, P. Várlaki, "Intelligent Methods for Car Deformation Modeling and Crash Speed Estimation," In Proc. of the 1st Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, SACI'2004, Timisoara, Romania, May 25-26, 2004, pp. 75-84.
[S107] Rövid, A., A.R. Várkonyi-Kóczy, P. Várlaki, "3D Model Estimation from Multiple Images," In Proc. of the 13th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2004, Budapest, Hungary, July 25-29, 2004, Vol. 3, pp. 1661-1666.
[S108] Rövid, A., A.R. Várkonyi-Kóczy, M.G. Ruano, "Corner Detection in Digital Images Using Fuzzy Reasoning," In Proc. of the 2nd IEEE International Conference on Computational Cybernetics, ICCC'2004, Vienna, Austria, Aug. 30-Sep.1, pp. 95-99.
[S109] Rövid, A., A.R. Várkonyi-Kóczy, Car Body Deformation Modeling Based on Soft Computing Techniques and Image Processing," In Proc. of the 3rd Int. Conf. On Global Research and Education in Intelligent Systems, Inter-Akademia'2004, Budapest, Hungary, Sep. 6-9, 2004, pp. 75-84.
[S110] Várkonyi-Kóczy, A.R., G. Samu, "Anytime Fourier Analysis," In Proc. of the 3rd Int. Conf. On Global Research and Education in Intelligent Systems, Inter-Akademia'2004, Budapest, Hungary, Sep. 6-9, 2004, pp. 476-485.
[S111] Rövid, A., A.R. Várkonyi-Kóczy, M.G. Ruano, "Automatic 3D Modeling Based on Soft Computing Techniques," In Proc. of the 8th IEEE Int. Conference on Intelligent Engineering Systems, INES'2004, Cluj-Napoca, Romania, Sep. 19-21, 2004, pp. 199-204.
[S112] Várkonyi-Kóczy, A.R., A. Rövid, G. Samu, "Soft Computing Techniques for the Improvement of Signal Processing Algorithms," In Proc. of the 2nd Serbian-Hungarian Joint Symposium on Intelligent Systems, SISY'2004, Subotica, Serbia and Montenegro, Oct. 1-2, 2004, pp. 201-214.
[S113] Várkonyi-Kóczy, A.R. , A. Rövid, G. Ruano, "Fuzzy Logic Supported Point Correspondence Matching for 3D Reconstruction," In Proc. of the 5th Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 11-12, 2004, pp. 47-58.
[S114] Rövid, A., A.R. Várkonyi-Kóczy, P. Várlaki, "An Application of Intelligent 3D Reconstruction in Vehicle System Dynamics," In Proc. of the 2nd IEEE Int. Conference on Artificial Intelligence in Science and Technology, AISAT 2004, Tasmania, Hobart, Australia, Nov. 21-25, 2004, pp. 257-262.
[S115] Várkonyi-Kóczy, A.R, A. Rövid, "Iterative Neural Network Model Inversion," In Proc. of the 3rd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2005, Herlany, Slovakia, Jan. 24-25, 2005, pp. 163-174.
[S116] Várkonyi-Kóczy, A.R., A. Rövid,, "Point Correspondence Matching for 3D Reconstruction Using Fuzzy Reasoning," In Proc. of the 3rd IEEE International Conference on Computational Cybernetics, ICCC'2005, Mauritius, Apr. 13-16, 2005, pp. 87-92.

[S117] Várkonyi-Kóczy, A.R., P. Várlaki, "Uncertainty Representation in Nonlinear and Hybrid Systems," In Proc. of the 3rd IEEE International Conference on Computational Cybernetics, ICCC'2005, Mauritius, Apr. 13-16, 2005, pp. 313-318.
[S118] Várkonyi-Kóczy, A.R., A. Rövid, "Observer Based Iterative Neural Network Model Inversion," In Proc. of the 14th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2005, Reno, USA, May 22-25, 2005, pp. 402-407.
[S119] Várkonyi-Kóczy, A.R., A. Rövid, "Improved Fuzzy Based Corner Detection Method," In Proc. of the IEEE Int. Workshop on Soft Computing Applications, SOFA'2005, Szeged-Arad, Hungary-Romania, Aug 27-30., 2005, pp. 237-242.
[S120] Várkonyi-Kóczy, A.R., M.G. Ruano, "Anytime Fourier Transformation," In CD ROM Proc. of the IEEE Int. Symposium on Intelligent Signal Processing, WISP'2005, Faro, Portugal, Sep. 1-3, 2005, ISBN: 0-7803-9031-8.

[S121] Várkonyi-Kóczy, A.R., A. Rövid, P. Várlaki, "Fuzzy Based Brightness Compensation for High Dynamic Range Images," In Proc. of the 9th IEEE Int. Conference on Intelligent Engineering Systems, INES'2005, Mediterranean Sea, Greece, Turkey, Sep. 16-19, 2005, pp. 245-248.

[S122] Várkonyi-Kóczy, A.R., A. Rövid, "Dynamic, Brightness and Contrast Tuned Fuzzy Based Image Processing Algorithms," In Proc. of the 4th Int. Conf. On Global Research and Education in Intelligent Systems, Inter-Akademia'2005, Wuppertal, Germany, Sep. 19-22, 2005, pp. 477-484.

[S123] Várkonyi-Kóczy, A.R., A. Rövid, P. Várlaki, "Fuzzy Based Corner Detector for Point Correspondence Matching," In. Proc. of the 2nd Int. Symposium on Computational Intelligence and Intelligent Informatics, ISCIII'05, Tunis-Gammarth, Tunisia, Oct. 14-16, 2005, pp. 148-153.

[S124] Várkonyi-Kóczy, A.R., A. Rövid, "Fuzzy Logic Supported High Dynamic Range Image Reproduction Techniques," In Proc. of the 6th Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 18-19, 2005, pp.117-129.

[S125] Várkonyi-Kóczy, A.R., A. Rövid, "A New Corner Detector Supporting Feature Extraction for Automatic 3D Reconstruction," In Proc. of the 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2006, Herlany, Slovakia, Jan. 20-21, 2006, pp. 195-208.

[S126] Várkonyi-Kóczy, A.R., A. Rövid, "High Dynamic Range Image Reproduction Methods," In CD-Rom Proc. of the 2006 IEEE Instrumentation and Measurement Technology Conference, IMTC/2006, Sorrento, Italy, April 24-27, 2006, ISBN:0-7803-9360-0.

[S127] Várkonyi-Kóczy, A.R., A. Rövid, "CASY - an Intelligent Car Crash Analysis System," in Proc. of the Int. Symposium on Research and Education in Innovation Era, Arad, Romania Nov. 16-18, 2006, pp.. I.-XI. (invited plenary paper).

[S128] Várkonyi-Kóczy, A.R., A. Rövid, Sz. Balogh, T. Hashimoto, Y. Shimodaira, "Hygh Dynamic Range Imaging Technique Based on the Merging of Maximum Dense Information Regions," in Proc. of the 7th Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. 24-25, 2006, pp. 57-68.

[S129] Rövid, A., T. Hashimoto,Y. Shimodaira, A.R.Várkonyi-Kóczy, "Gradient Based Synthesized Edge Image Using Multiple Exposure Images," In Proc. of the 13th Int. Display Workshops, Otsu, Japan, Dec.6-8, 2006, pp. 21-24.

[S130] Várkonyi-Kóczy, A.R., A. Rövid, "Intelligent 3D Deformation Modeling in Vehicle System Dynamics," In Proc. of the 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2007, Poprad, Slovakia, Jan. 24-26, 2007, pp. 59-70.

[S131] Rövid, A., T. Hashimoto, A.R. Várkonyi-Kóczy, Y. Shimodaira, "Information Enhancement Method for Image Retrieval and Object Recognition," 3rd Int. Symposium on Computational Intelligence and Intelligent Informatics, ISCIII'2007, Agadir, Morocco, March 28-30, pp. 25-29.

[S132] Várkonyi-Kóczy, A.R., P. Várkonyi, A. Rövid, "Intelligent 3D Car-Body Deformation Modeling," In CD-ROM Proc. of the Second International Conference on Systems, ICONS'2007, Sainte-Luce, Martinique, April 22-28, 2007, ISBN: 978-0-7695-2807-6.

[S133] Rövid, A., A.R.Várkonyi-Kóczy, T. Hashimoto,Sz. Balogh, Y. Shimodaira, "Gradient Based Synthesized Multiple Exposure Time HDR Image," In CD-ROM Proc. of the 2007 IEEE Instrumentation and Measurement Technology Conference, IMTC'2007 Warsaw, Poland, May 1-3, 2007, ISBN: 1-4244-0589-0.

[S134] Rövid, A., A.R. Várkonyi-Kóczy, T. Hashimoto, Y. Shimodaira, "Gradient Based Synthesized Maximum Dense High Dynamic Range Image," In Proc. of the 11th IEEE Int.

Conference on Intelligent Engineering Systems, INES'2007, Budapest, Hungary, June 29-July 1, 2007, pp. 121-126.

[S135] Ribeiro, L., A.E.B. Ruano, M.G. Ruano, P. Ferreira, A.R. Várkonyi-Kóczy, "Improving the Diagnosis of Ischemic CVA's through CT Scan with Neural Networks," 2nd IEEE International Workshop on Soft Computing Applications, SOFA'2007, Gyula, Hungary and Oradea, Romania, Aug. 21-23, 2007, pp. 39-43.

[S136] Várkonyi-Kóczy, A.R., A. Rövid, T. Hashimoto, Y. Shimodaira, "Fuzzy Logic Supported Multiple Exposure Time Coloured HDR Image Synthesization," In Proc. of the 6th Int. Conf. On Global Research and Education in Intelligent Systems, Inter-Akademia'2007, Hamamatsu, Japan, Sep.26-30, 2007, pp. 168-177.

[S137] Várkonyi-Kóczy, A.R., A. Rövid, T. Hashimoto, HDR Colored Information Enhancement Based on Fuzzy Image Synthesization, In Proc. of the 5th IEEE Int. Symposium on Intelligent Signal Processing, WISP'2007, Alcala de Henares, Spain, Oct. 3-5, 2007, pp. 187-192.

[S138] Várkonyi-Kóczy, A.R., A. Rövid, "Fuzzy Logic Supported Coloured Image Information Enhancement," In Proc. of the 7th Int. Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, Nov. , 2007, pp. 189-200.

[S139] Várkonyi-Kóczy, A.R., A. Rövid, "Fuzzy Logic Supported Primary Edge Extraction in Image Understanding," In Proc. of the 17th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2008, Hong Kong, China, June 1-6, 2008, accepted.