# Soft Computing Techniques for the Improvement of Signal Processing Algorithms

**Annamária R. Várkonyi-Kóczy**[1,2]
**András Rövid**[1,2]
**Gábor Samu**[1,2]

[1]Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary, koczy@mit.bme.hu
[2]Integrated Intelligent Systems Japanese-Hungarian Laboratory

*Abstract: Authors present an analysis about the consequences of complex measurement and signal processing tasks. They show that measurement and signal processing problems of now-a-days open new dimensions for the interpretation of the basic concepts of measurement and signal processing and make the reevaluation of these concepts necessary. Traditional methods fail in many cases to yield useful solutions, especially when measurement and signal processing problems reveal considerable complexity, involve a wide spectrum of various disciplines and require a multitude of components and methods. Since traditional methods, without a proper resource management supported at the system level, seem inappropriate to solve such problems, qualitatively new methods are needed. The present paper seeks answers to these problems. It gives a brief overview of various imprecise computational methods and discusses their applicability to treat complex measurement and signal processing problems.*

## 1. Introduction

Up to now classical problem solving methods proved to be entirely sufficient in the measurement, signal processing and system engineering. Nowadays, however, measurement science tackles problems of previously unseen spatial and temporal complexity (consider e.g. measurement problems of industrial plants, large scale environmental systems or laboratory tests in health care) and in a large number of cases traditional information processing methods and equipment failed to handle these problems. It became clear that new ideas are required for specifying, designing and implementing sophisticated measurement systems.
Similar problems appeared in numerous other fields of research, and it was natural to explore and adopt the solutions. In the field of Artificial Intelligence (AI), Soft Computing (SC) and Imprecise Computation (IC) several methods have been developed that address the problem of nonnumeric information processing and the rational control of limited resources. In AI so called 'anytime algorithms' [1,2]

offer considerable control over resources, in SC and IC the trade off between accuracy and resource usage is possible [3,4].
In measurements the used model serves as a basis to design information processing methods and to implement them at the equipment level. In case of complex measurements analytical models leading to a well defined numerical optimal information processing are not enough. The complexity of the problem manifests itself not only as a hierarchy of subsystems and relations, but also as the variety of modeling approaches needed to grasp the essence of the modeled phenomena. Analytical models rarely suffice. Frequently numerical information is missing or is uncertain making place for various qualitative or symbolic representation methods.
This situation can be still complicated by the fact that various modeling approaches, expressing different aspects of the problem, should be used together in a well orchestrated integrated way. Even more disturbing is that such traditional metrological concepts, like accuracy, error, scale, unit, etc. are no more applicable in their usual approved sense.
As a consequence researchers drew from new methods and fields to tackle the problem. Artificial Intelligence offered means to handle nonnumeric and vague information. Imprecise Computation and Soft Computing offered a novel view at the computational accuracy as a utility rather than an ultimate aim of the development. In the next section an overview of some kind of such methods is given.

# 2. Overview of Non-Classical Computing Techniques

Recently we meet new efforts to combine the ideas of the traditional and the soft computing methods. Some of the classical concepts and goals have to be analyzed and reevaluated, and the new methods must be placed into the context of the accepted classical frame of measurement and signal processing.

## 2.1. Anytime Systems

Today there is an increasing number of applications where the computing must be carried out on-line, with a 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 signal processing, diagnostics, monitoring, and control 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 the model of the supervised system is used and the evaluation of the system model must be carried out on-line, thus the model must not only be correct, but also treatable by the limited resources

during limited time. Moreover, if some abnormality occurs in the system's behavior it may cause the reallocation of a part of the finite resources from the evaluation of the system model to another task. Also in case of an alarm signal, lower response time may be needed. Having approximate results can also help in making decisions for the further processing.
In these cases, the so-called anytime algorithms and systems [5] can be used advantageously, which are able to provide guaranteed response time and are flexible in respect to the available input data, time, and computational power. Recursive or iterative algorithms are popular tools in anytime systems, because their complexity can be easily and flexibly changed. These algorithms always give some, possibly not accurate result and more and more accurate results can be obtained if the calculations are continued. Unfortunately, the usability of iterative algorithms is limited. Besides the iterative algorithms, in a more general frame, a wide-range of other types of computing methods/algorithms can be applied in anytime systems. This frame means that a modular architecture is used [6]. The system is composed of modules each of them offering several implementations for a given task. These units (implementations within a given module) have uniform interface (same set of inputs, outputs, and solve the same problem) but can be characterized by different attribute-values, i.e. differ in their computational need and accuracy. At a given time, in the knowledge of the temporal conditions (tasks to complete, achievable time/resources, needed accuracy, etc.) an expert system can choose the adequate configuration, i.e. the units from the modules, which will be used. This means the optimization of the whole system instead of individual modules.Anytime processing may have great advantages in signal processing, monitoring, diagnostics, control, and related fields.

### 2.1.1 Block-Recursive Averagers

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

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

For n ≥ N the "block-oriented" linear averaging has the form of

$$X(n-N) = \frac{1}{N} \sum_{k=1}^{N} x(n-k) \tag{2}$$

while the block-recursive average can be written as

$$y(n) = \frac{n-N}{n} y(n-N) + \frac{N}{n} X(n-N) \tag{3}$$

If (3) is evaluated only in every Nth step, i.e. it is maximally decimated, then we can replace (3) with n=mN, m=1,2, ..., by

$$y(mN) = \frac{m-1}{m} y[(m-1)N] + \frac{1}{m} X[(m-1)N] \tag{4}$$

or simply

$$y(m) = \frac{m-1}{m} y(m-1) + \frac{1}{m} X(m-1) \tag{5}$$

where m stands as block identifier. Note the formal correspondence with (1).
If the block identifier m in equation (5) is replaced by a constant Q > 1 then an exponential averaging effect is achieved. 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 paper 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 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

$$y(n) = y(n-1) + \frac{1}{N}[x(n-1) - x(n-N-1)] \tag{6}$$

If in (6) the input samples are replaced by preprocessed data, e.g. as in (2), then a block-recursive form is also possible:

$$y(n) = y(n-N) + [X(n-N) - X(n-2N)] \tag{7}$$

which, however, has no practical meaning, since it gives back (2). But if the window size is integer multiple of N, e.g. MN, then the form

$$y(n) = y(n-N) + \frac{1}{M}[X(n-N) - X(n-(M+1)N) \tag{8}$$

has real importance. If (8) is evaluated only in every Nth step, i.e. it is maximally decimated, then we can replace (8) with n=mN, m=1,2, ..., by

$$y(mN) = y[(m-1)N] + \frac{1}{M}[X((m-1)N) - X((m-M-1)N)] \tag{9}$$

or simply

$$y(m) = y(m-1) + \frac{1}{M}[X(m-1) - X(m-M-1)] \qquad (10)$$

where m stands as block identifier. Note the formal correspondence with (6).

The generalization of these averaging schemes to signal transformations and/or filter-banks is straightforward. Only (2) should be replaced by the corresponding "block-oriented" operation. Figure 1 shows the block diagram of the linear averaging scheme. This is valid also for the exponential averaging except m must be replaced by Q. These frameworks can incorporate a variety of possible

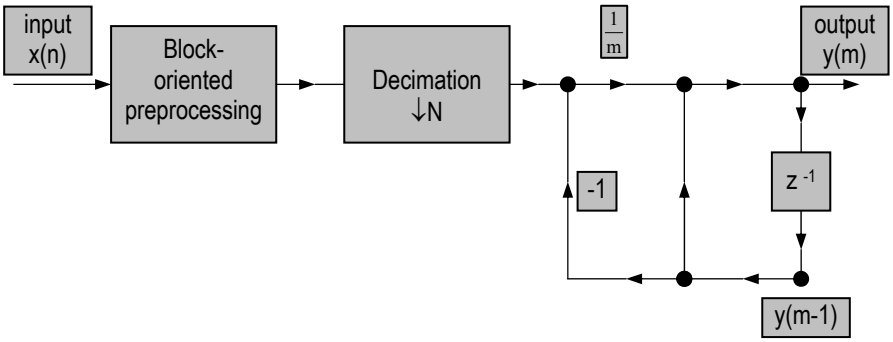



Figure 1: Block-recursive linear averaging signal processing scheme, n=mN

transformations and corresponding filter-banks which permit decimation by the block-size. Standard references, e.g. [7] provide the necessary theoretical and practical background. The idea of transform-domain signal processing proved to be very efficient especially in adaptive filtering (see e.g. [8]). 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. 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 since 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".

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, where this positioning can be solved with the so-called complementary DFT which is generated using the Nth roots of -1. This DFT locates its channels into the positions $\pi/N$, $3\pi/N$, 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. 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. 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 ej $2\pi$ n/4N and ej $6\pi$ 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.

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 [9] and [10] are sliding-window transformations, i.e. 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 either the input rate, the maximally decimated one, or any other in between. These techniques are not fast algorithms, however, “produce” less delay as those block-oriented algorithms which start working only after the arrival of the complete input data block.

## 2.2 Anytime Fourier Transformation

The above detailed algorithms can advantageously be applied in anytime systems (see Fig. 2). If the block-recursive linear averager (L=mN) (in case of sliding-window averager MN) is composed of m N-point DFTs then after the arrival of the first N samples we will have a rough approximation of the signal, after 2N samples a better one, etc. The accuracy of the pre-results will not be exact, however the error is in most cases tolerable or even negligible.

In the followings a simple example is presented which illustrates the usability of

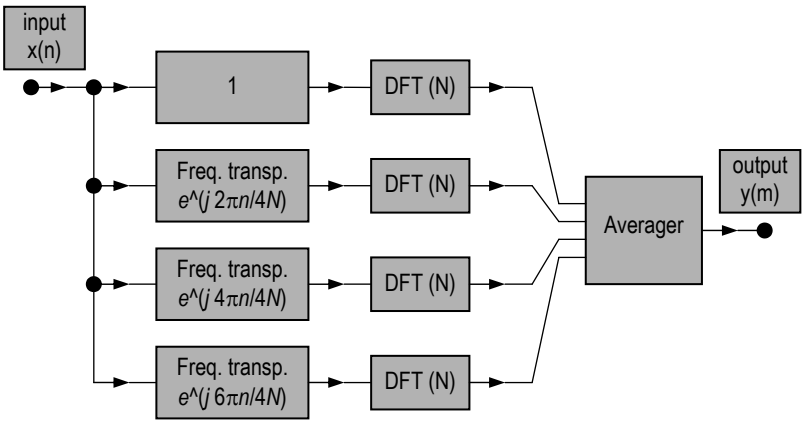



Figure 2: The block structure of the Anytime Fourier Transformation

In the example a 256-channel DFT is calculated recursively with N=64 for m=1,2,8,16. The input sequence was

$$x(n) = \cos\left(\frac{\pi n}{2}\right) + rand - 0.5 \tag{11}$$

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 on Figure 3. The improvement in resolution and noise reduction is remarkable.

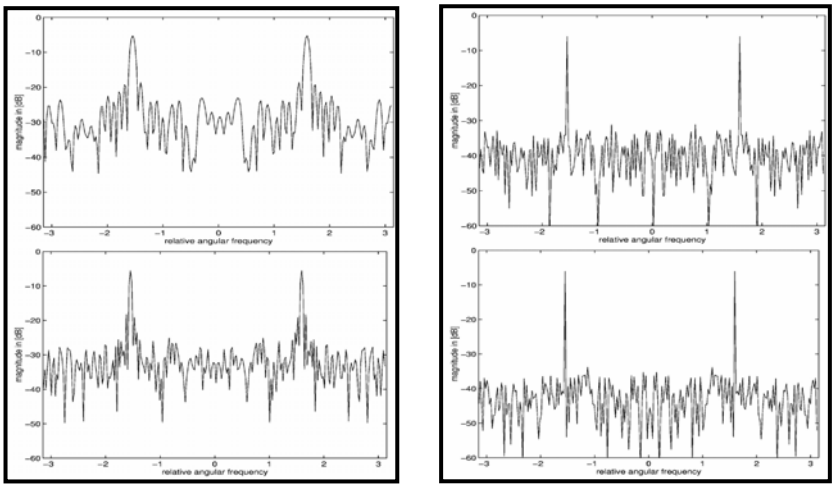



Figure 3: 256 channel DFT of a single sinusoid plus noise. N=64, exactly at a DFT channel (x(n)=cos (πn/2) + rand - 0.5).

## 2.3. Fuzzy based signal processing techniques

Fuzzy logic is rapidly emerging as a powerful resource of instrumentation, signal processing and measurement because fuzzy approach is able to deal with the typical uncertainty, which characterizes any physical system. In the following sections a brief overview of such methods is given[11][12].

### 2.3.1 Fuzzy Based Noise Elimination

A major task in the field of digital processing of measurement signals is to extract information from sensor data corrupted by noise [13][14]. For this purpose we will use a special fuzzy system characterized by an IF-THEN-ELSE structure and a specific inference mechanism. Different noise statistics can be addressed by adopting different combinations of fuzzy sets and rules [13][14].

Let x(r) be the pixel luminance at location r=[r1,r2] in the noisy image where r1 is the horizontal and r2 the vertical coordinate of the pixel. Let N be the set of eight neighboring pixels (see Fig. 4a). The input variables of the fuzzy filter are the amplitude differences defined by:

$$\Delta x_j = x_j - x_0, j = 1,...,8 \qquad (12)$$

where the xj, j=1,…,8 values are the neighboring pixels of the actually processed pixel $x_0$ (see Fig. 4a).

| | | |
|---|---|---|
| $x_1$ | $x_2$ | $x_3$ |
| $x_4$ | $x_0$ | $x_5$ |
| $x_6$ | $x_7$ | $x_8$ |

Fig. 4a: The neighboring pixels of the actually processed pixel x0

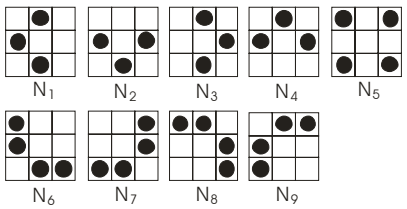



Fig. 4b: Pixel Patterns

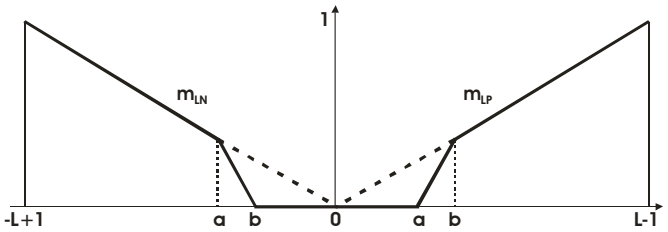



Fig. 5: Membership function $m_{LN}$ and $m_{LP}$. Parameters *a* and *b* are appropriate constant values

Let $y_0$ be the luminance of the pixel having the same position as $x_0$ in the output signal. This value is determined by the following relationship:

$$y_0 = x_0 + \Delta y \tag{13}$$

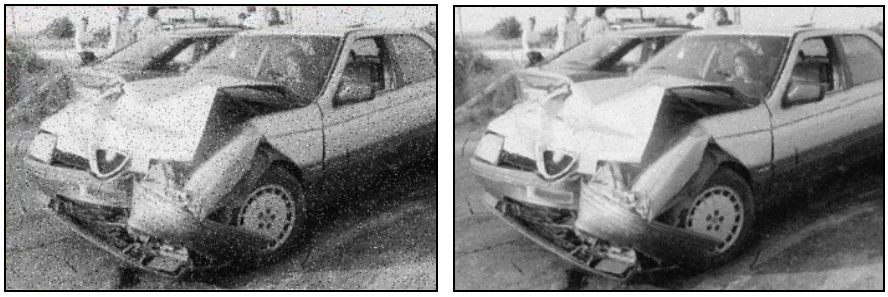
where $\Delta y$ is determined thereinafter (see eq. (16)). Let the rulebase deal with the pixel patterns N1,…,N9 (see Fig. 4b). The value $y_0$ can be calculated, as follows [4]:

$$\lambda = MAX\left\{MIN\left\{m_{LP}(\Delta x_j) : x_j \in N_i\right\}, i = 1,...,9\right\} \tag{14}$$

$$\lambda^* = MAX\left\{MIN\left\{m_{LN}(\Delta x_j) : x_j \in N_i\right\}, i = 1,...,9\right\} \tag{15}$$

$$\begin{aligned} \Delta y &= (L-1)\Delta\lambda \\ y_0 &= x_0 + \Delta y \end{aligned} \tag{16}$$

where $\Delta\lambda=\lambda-\lambda^*$, $m_{LP}$ and $m_{LN}$ correspond to the membership functions and $m_{LP}(u)=m_{LN}(-u)$ (see Fig. 5.). The filter is recursively applied to the input data. An example of the described fuzzy-filter can be seen in Figure 6.



(a) (b)

Figure 6: (a) Original photo of a crashed car corrupted by noise, (b) Fuzzy-filtered image of the photo

### 2.3.2 Fuzzy Based Edge Detection

Edge detection in an image is a very important step for a complete image understanding system. In fact, edges correspond to object boundaries and are therefore useful inputs for 3D reconstruction algorithms. The proposed fuzzy based edge detection [15] can very advantageously be used for this purpose.

Let $x_{i,j}$ be the pixel luminance at location [i,j] in the input image. Let us consider the group of neighboring pixels which belong to a 3x3 window centered on xi,j.

The output of the edge detector is yielded by the following equation [15]:

$$\begin{aligned} z_{i,j} &= (L-1)MAX\{m_{LA}(\Delta y_1), m_{LA}(\Delta y_2)\} \\ \Delta y_1 &= \left| x_{i-1,j} - x_{i,j} \right| \\ \Delta y_2 &= \left| x_{i,j-1} - x_{i,j} \right| \end{aligned} \qquad (17)$$

where $z_{i,j}$ is the pixel luminance in the output image and $m_{LA}$ is the used membership function (see Fig. 7). Pixels $x_{i-1,j}$ and $x_{i,j-1}$ are the luminance values of the left and the upper neighbor of the pixel at location [i,j].

The fuzzy based technique compared to the classical methods provided better results with less (very small) processing time. Figure 8 shows an example of the edge detection results.

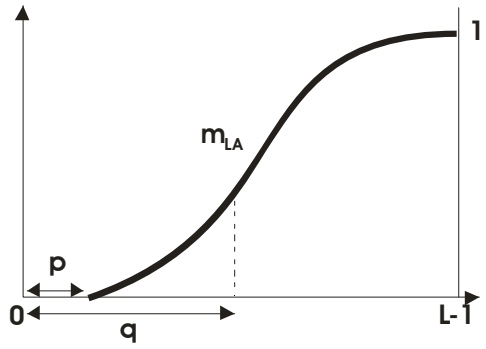
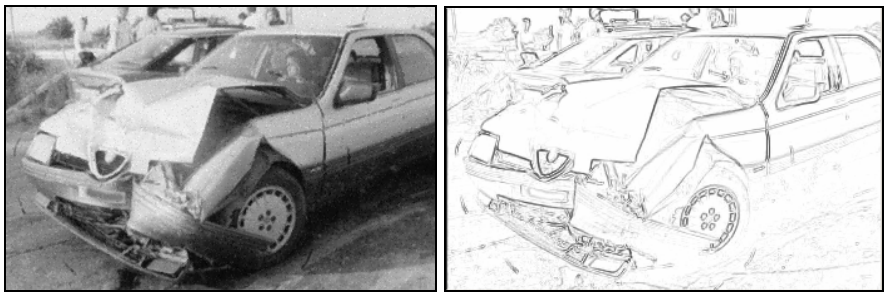



Fig. 7: Membership function $m_{LA}$. Parameters *p* and *q* are appropriate constant values



(a) (b)

Fig. 8: (a) Original photo, (b) After fuzzy based edge detection

**2.3.3 Fuzzy Based Corner Detection**

Corner detection should satisfy the following requirements:

- All the true corners should be detected
- No false corners should be detected
- Corner points should be well localized
- Corner detector should be robust with respect to the noise

Förstner determines corners as local maxima of function H(x,y) [16].

$$H(x,y) = \frac{\left(\frac{\partial I}{\partial x}\right)^2\left(\frac{\partial I}{\partial y}\right)^2 - \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right)^2}{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{18}$$

Starting from the algorithm of Förstner a new improved corner detection algorithm can be developed by combining it with fuzzy reasoning. This is used for the characterization of the continuous transient between the localized and not localized corner points, as well. The algorithm consists of the following steps. First, the picture, in which we have to find the corners, is preprocessed. As a result of the preprocessing procedure the noise is eliminated. For this purpose we apply the intelligent fuzzy filters described in [13] and [14]. If necessary the image is also smoothed before taking the derivatives. After noise-filtering, the first

derivatives of the intensity function I(x, y) are calculated in each image point. For this purpose we apply the following convolution masks:

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

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

For increasing the effectiveness of the corner detection it is proposed to smooth each of the entries $I_x^2$, $I_y^2$, $I_xI_y$, in eq. (18), which correspond with the first partial derivates of the intensity function I(x,y) (here x,y denote the 2D coordinates of the pixels). This can be done by applying a Gaussian 6x6 convolution kernel with $\sigma$=1 [17]. As the following step, the values H(x,y) are calculated for each image point with the help of the previously determined $I_x^2$, $I_y^2$ and $I_xI_y$ smoothed values. If the detected corners are neighbors, then we should keep only the corner having the largest calculated value H(x,y). The others are to be ignored. In most cases we can not unambiguously determine that the analyzed image point is a corner or not with only the help of a certain concrete threshold value, therefore in the proposed algorithm fuzzy techniques are applied for the calculation of the values (corners) which increases the rate of correct corner detection. By the score of the membership function (see Fig. 9) of fuzzy set "corners" we can determine a weighting factor, which characterizes the rate of the corner's membership. The value of the membership function mc is 1 for those image points for which the calculated value H equals or is larger than the given threshold value. With the help of the parameters *p*, *q* (see Fig. 9) the shape of the membership function can be modified and so the sensitivity of the described detector can be changed. Finally the output of the proposed corner detector is yielded by the following relation:

$$C_{x,y} = (L-1)m_c(H), \tag{19}$$

where $C_{x,y}$ represent the gray-level intensity values of the output image, x and y are the horizontal and vertical coordinates of the processed image point, *L* is the largest gray level intensity value, and *H* stands for the calculated H(x,y) values.

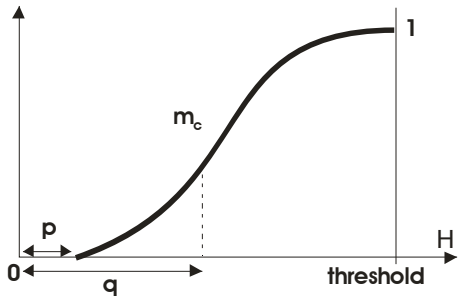
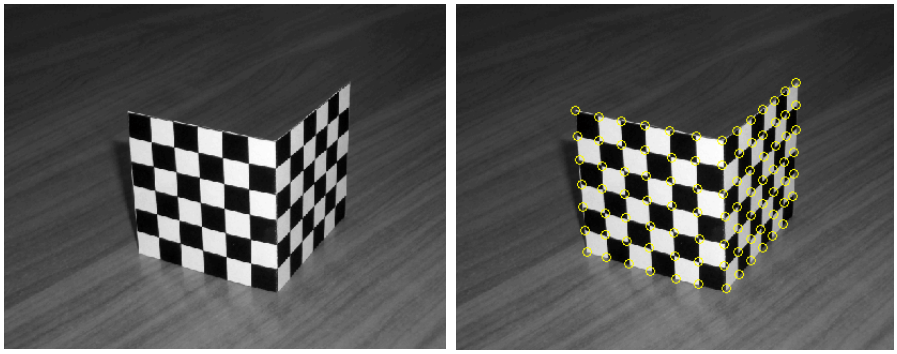



Fig. 9: Membership function of fuzzy set corner (mc). The axis H is the axis of the calculated H(x,y) values.



(a) (b)

Figure 10: (a) Original photo, (b) After corner detection

**Conclusions**

The increased complexity of measurement systems caused classical problem solving methods, especially in resource-bounded applications, to fail to produce 'usable' solutions. This led to focus on knowledge representation, information handling, and different ways of expressing uncertainty. Soft Computing methods are serious candidates for handling many of the theoretical and practical limitations and, in many cases, are the best if not the only alternatives for emphasizing significant aspects of system behavior with a burden of less precision. The real power of such methods can only be exploited, however, only when they are embedded in a framework that provides efficient means for communication and information sharing, thus providing firm basis for using methods rather different in nature together.

**Acknowledgement**

This work was sponsored by the Hungarian Fund for Scientific Research (OTKA T 035190) and the Hungarian-Portugese Intergovern. S&T Cooperation Programme (P-24/03).

**References**

[1] S. Zilberstein,S. J. Russel: "Reasoning about optimal time allocation using conditional profiles", Proc. of AAAI-92 Workshop on Implementation of Temporal Reasoning, pp. 191-197, San Jose, California, 1992
[2] S. Zilberstein,S. J. Russel: Constructing utility-driven real-time systems using anytime algorithms", in Procs. of the IEEE Workshop on Imprecise and Approximate Computation, pp. 6-10, Phoenix, Arizona, 1992
[3] J.W.S. Liu, et al.: Imprecise Computations, Proc. of the IEEE, Vol. 82, No.1, Jan 1994, pp. 83-93.
[4] L. Zadeh: Fuzzy Logic, Neural Networks, and Soft Computing, Communications of the ACM, March 1994, Vol. 37, No.3, pp. 77-83.
[5] Zilberstein, S., "Using Anytime Algorithms in Intelligent Systems," AI Magazine, Vol. 17, No. 3, 1996, pp. 73-83.
[6] Várkonyi-Kóczy, A.R., A. Ruano, P. Baranyi, O.Takács, "Anytime Information Processing 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.
[7] Crochiere, R.E., L.R. Rabiner, Multirate Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N.J, 1983.
[8] Shynk, J.J., "Frequency-Domain and Multirate Adaptive Filtering," IEEE Signal Processing Magazine, Jan. 1992, pp. 15-37.
[9] Péceli, G., "A Common Structure for Recursive Discrete Transforms," IEEE Trans. on Circuits and Systems, Vol.33, Oct. 1986, pp. 1035-1036.
[10]Padmanabhan, M., K. Martin and G. Péceli, Feedback-Based Orthogonal Filters, Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.
[11] I. J. Rudas, Á. Szeghegyi, J. F. Bitó, G. Geary: Non Monotone Generalized Fuzzy Operations for Fuzzy Logic Controllers, 5th IEEE International Workshop on Robotics in Alpe-Adria-Danube Region, June 1996, Budapest, pp. 529-533.
[12] Imre J. Rudas, M. O Kaynak, János F. Bitó, Ágnes Szeghegyi:
New Possibilities in Fuzzy Controllers Design Using Generalized Operators. 5th International Conference on Emerging Technologies and Factory Automation, November 1996. Hawaii, pp. 513-517.
[13]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.
[14]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.
[15]C. Harris and M. Stephens, "A combined corner and edge detector", Proc. 4th Alvey Vision Conference, pp. 189-192, 1988.
[16]W. Förstner, "A feature based correspondence algorithm for image matching," Int. Arch. Photogramm. Remote Sensing, vol. 26, pp. 150-166, 1986.
[17] F. Catté, P.-L. Lions, J.-M. Morel, T.Coll, "Image selective smoothing and edge detection by nonlinear diffusion," SIAM Journal on Numerical Analysis, 32:1895-1909, 1992.