SIMPLE FUZZY NEURAL NETWORK FOR OCR

Josef Kokeš
Vysoká škola manažerské informatiky a ekonomiky, a.s., Vltavská 585/14, Praha
josef.kokes@vsmie.cz, www.vsmie.cz

ABSTRACT:
This paper shows a concrete example of how to use a system of fuzzy logic networks for reading data from a computer screen. The system has been proven particularly when the manufacturer has made deliberate steps that should make it difficult or impossible to OCR from the screen. Because recognition works for a small set of 11 characters, the result is very fast and sufficiently reliable.

KEYWORDS:
OCR, neural networks, defuzzyfication

INTRODUCTION

We often meet with the need to obtain data from various measurements or technological processes, and to keep them for later processing. We had to deal with one such case in our work, but I think that this is a fairly typical situation, and that similar problems researchers encounter quite often. In our case, an original task was - to read actual results from a PC at regular intervals. Soon it became clear that the same sequence can be used for business data processing, such as reading Exchange data.

THE BACKGROUND

We have been testing and debugging an expert system for intraday-trading in the real traffic on the American Stock Exchange [1], [2], [3] for a long time. Intraday-trading is an activity in which a man cannot hold the attention for a long time. Some results of a computerized expert system (ES) look quite promising, because while ES on one hand achieves worse results than a human expert, on the other hand these results are obtained systematically throughout the whole trading hours.

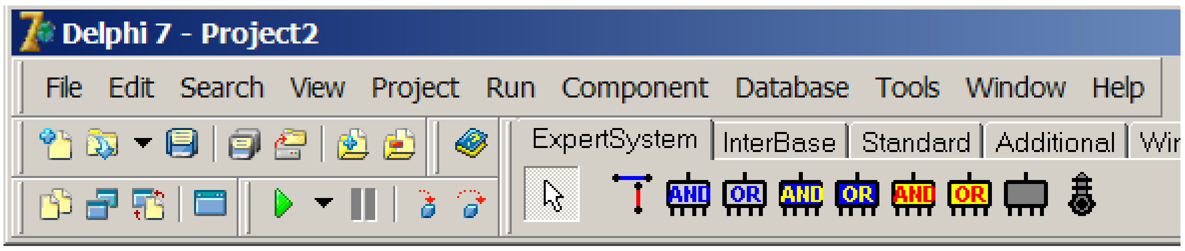

Figure 1. Delphi palette with Boolean, Extended and Fuzzy components

First applications were programmed in Delphi 7 and a full graphic environment for ES rules was introduced [1] , see Figure 1. An additional standard Delphi palette with components like "BoolAND" and "BoolOR" components was created to make graphic programming of ES easy. Later, a new engine based on fuzzy evaluation of expert system rules was created in JAVA Net Beans 6.7 [3].
Internet intraday-trading have been tested in real traffic via online broker TD Ameritrade. Just to mention some interesting results. First, it is clear that the expert system has not conducted all business days. This is partly because every day the situation is not suitable for this type of

trade, partly (and mainly) because in the meantime, the expert system was under maintenance. For the period from 16.10 to 11.12, i.e. for 27 calendar days, the expert system traded 11 days. Overall, the expert system within 11 business days created a net profit of $ 3939.39. The amount invested was $ 20000. (More precisely, there were only invested two-thirds of that amount because the remaining third is a "margin". Because the margin is a complex issue and there is no room for its more detailed explanation, we neglect that fact and do calculations as if invested amount was $ 20000).

Number of calendar days	27
- active days when trading was tested	11
Best one-day performance	2396 USD
Worst one-day performance	-261 USD
Total profit	**3939.39 USD**
Earnings calculated on the% of money invested:	
- for the days when trading	1.79% daily
- extrapolation to 1 year (365.25 days)	654% p.a.
- for calendar days	0.7295% daily
- extrapolation to calendar year	**266.5% p.a.**

This means that this expert system creates a profit on average $ 358.12 daily, which represents 1.791% of money invested. The gain can not be extrapolated to the whole year, because both stock market does not work on weekends and holidays, and as well, not every trading day is suitable for businesses. So we tried to quantify more realistic earnings. Neglecting the fact that the expert system could not run for several days because of its maintenance, its profit reached $ 3939.39 for 27 calendar days. This means that the average income per calendar day is about $ 145.90, representing an average daily value of 0.729%. If this value is extrapolated to the entire calendar year, we went out to an average annual profit of 266.5% p.a.

Currently the biggest obstacle for further debugging of expert system is the lack of relevant data. ES for the intra-day trading needs data on the basis of instantaneous values, while available data are at most 1-minute maxima and minima (or 1-minute OHLC, open-high-low-close). Live (instantaneous) values are available only online and usually are available only for customers as a paid service. Some brokers and software vendors supply data free of charge (e.g. NinjaTrader Inc.), but only for a limited time period from 1 to 3 months. All this makes debugging ES difficult and obtaining sources of data a must.

For the above reasons, we were trying the OCR from PC screen, originally designed to obtain data from the laboratory experiment, also use for obtaining instantaneous trading values.

OCR FROM THE PC SCREEN

Attempts to read values directly from the PC screen failed totally. A solution based on the idea that the values will be read from the screen using a suitable computer program for optical recognition (OCR), and subsequently stored on disk for further processing proved to be unfeasible. The reason is that the characters displayed on the screen are pre-distorted in

various ways, so their automated reading is not easy. Probably, the data supplier has made deliberate steps that should make it difficult or impossible to OCR from the screen.
Figure 2 shows an example (screenshot) of a display captured from online web trading software. Characters are perfectly readable to humans, but when deploying OCR problems arise. You can see that for some reason, probably intentionally, data and results are visually distorted. This means that one and the same character is displayed on the screen each time differently, in a large number of variants.

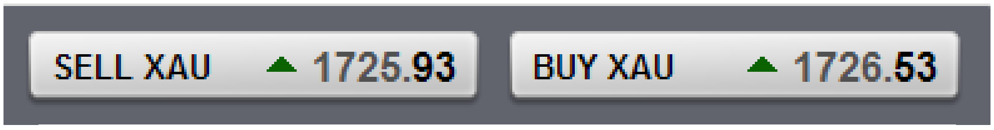

Figure 2. The appearance of text and numbers captured from the PC screen

This is shown in detail on Figure 3, which is an enlarged detail of one section of text. It is obvious that what the human eye appears as a single character or number, in fact, is a very complicated unit composed of pixels of different colors and intensities. However, the color tone and intensity of the display changes.

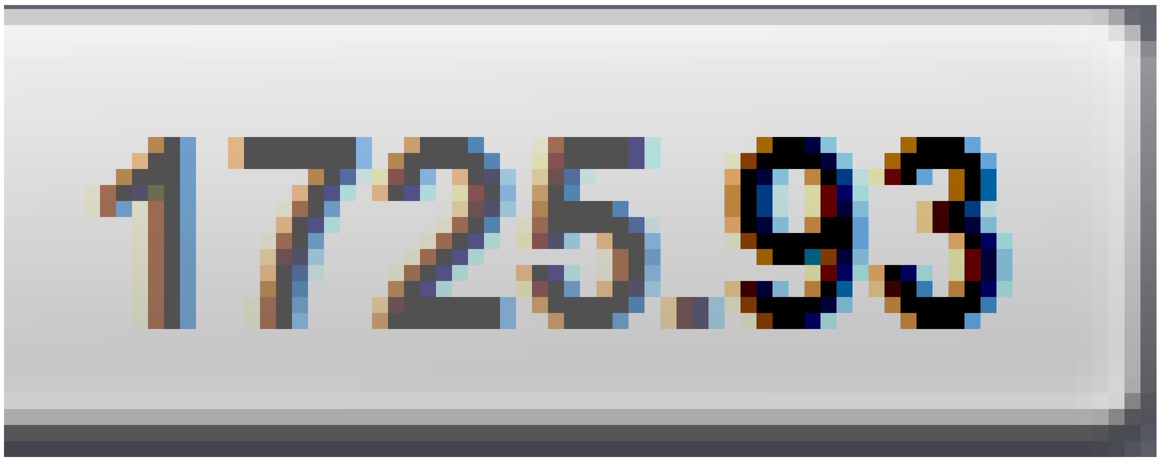

Figure 3. Enlarged details of numbers and characters

In addition, probably intentionally, also character shapes are not always exactly the same. This is why we cannot use simple thresholding, and then converting to black and white scale. Attempts to solve these problems led to the idea, to put on for the OCR neural network and train it to recognize numbers 0-9 and the character "dot".

NEURAL NETWORKS

Most researchers implement their neural-network application in Matlab. So it is difficult to embed it into PC program, also it would require user's knowledge about Matlab. In this paper we try to use Java program language to implement of Fuzzy logic using table lookup scheme for OCR. Our tool is easy to use and get a high accuracy results. This is why used it also later, for predicting future data.

The general idea of neural network for character recognition is shown on Figure 4. Each character image is divided to 9x12 = 108 pixels and as such it is loaded. Unfortunately, the color of each pixel differs randomly. Of course, each color component can be used as a separate component of the input signal, but it would lead to overly complex structure and

learning time would be prohibitively long. As it turned out better, first convert color to grayscale and resulting intensities express using fuzzy values.
Unified values obtained are used as inputs to the neural network. The neural network is trained to recognize characters, so in theory it would be sufficient to identify the outputs 11 of the numerals 0..9 and decimal point.
Unfortunately, in reality it is not so simple because the variation of colors and intensities of pixels is too big. This led to a relatively frequent occurrence of errors.
But that application is indicated as extremely sensitive to exact digit recognition and any error would be associated with a substantial financial loss. This is why is used a slightly higher number of output signals (16 in final version). Additional signals are trained to outputs like "upper horizontal line inside character", "closed circle in lower part of character" (to distinguish 5 from 6) and so on.
Only these output signals, which are also in the range compatible with fuzzy logic, are used for fuzzy rule-part process. Here, the system applies the rules as e.g.

"if there is a closed ring at the bottom, it cannot be number 5",
"if there is an upper horizontal line inside character, it cannot be 8" etc.

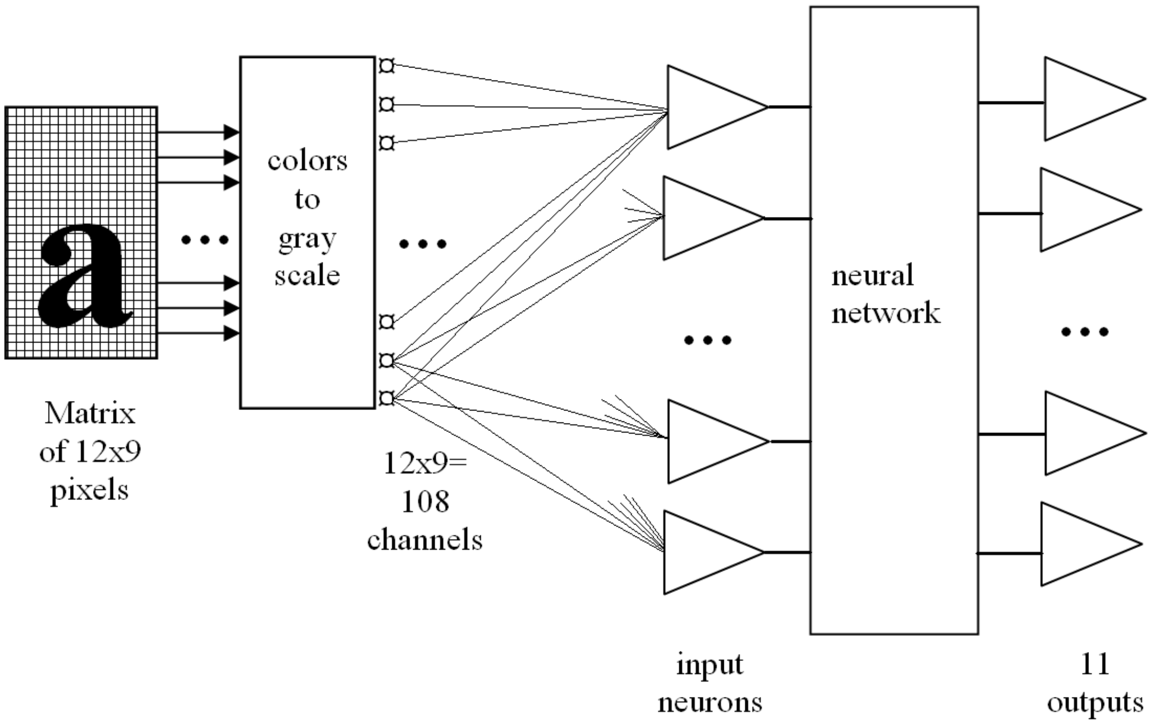

Figure 4. General scheme of neural recognition

TRAINING OF FUZZY LOGIC SYSTEM

Training of Fuzzy Logic system using a table – lookup scheme is one of the most important parts of our character recognizer. Assume, we have a set of input and output data pairs:

$$(x_1^1, x_2^1; y^1), (x_1^2, x_2^2; y^2), ... \qquad (1)$$

The task here is to generate a set of fuzzy IF-THEN rules from the given input-output pairs of (1), and then use them to determine logic system. The approach [4] divides into five steps as follows:

- **Step 1**: Divide each domain interval of input and output variable into 2*N+1 regions (N can be different for each variable and the width of each region can be equal or unequal), denote by SN (Small N), …, S1 (Small 1) , CE (Center), B1 (Big 1), …, BN (Big N) and then assign each region to one fuzzy membership function.

For example: We divide domain of x1, x2 and y into 5 regions (N=2) as Figure 4 below:

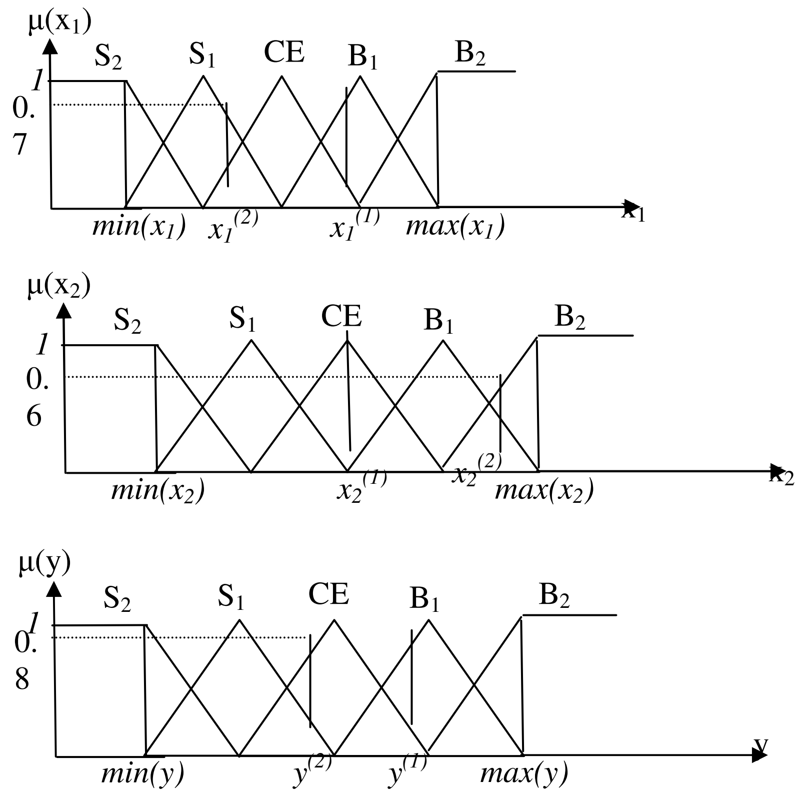

Figure 5. Input-output regions and corresponding member ship functions

- **Step 2**: This step further divides into three small steps. First, calculate the degree of given in different region. Second, assign the given to region with maximum degree. Finally, obtain one IF-THEN rule from one pair of given input-output data. For the example above, x11 has degree 0.2 in CE and 0.8 in B1, equals to zero in all other regions. Similarly, x21 has degree 1 in CE, equals to zero in all other regions, y1 has 0.2 degree in CE, 0.8 degree in B1 and equals to zero in all other regions. So x11 is belonged to regions B1 with 0.8 degree, x22 is belonged to region CE with 1.0 degree, y1 is belonged to region B1 with 0.8 degree. Finally, we obtain rule 1: IF x1 is B1 AND x2 is CE, THEN y is B1 from data pair (x11, x21 →y1). Similarly, we obtain second rule 2 IF x1 is S1 AND x2 is B2, THEN y is CE from data pair (x12,x22 →y2).

- **Step 3**: Calculate degree of each rule base on product strategy.

$$D(rule) = \mu_A(x_1) \times \mu_B(x_2) \times \mu_C(y) \qquad (2)$$

For example, to calculate degree of rule: IF x1 is S1 and x2 is B2 THEN y is CE:

$$D(rule) = \mu_{S1}(x_1) \times \mu_{B2}(x_2) \times \mu_{CE}(y) = 0.7 \times 0.6 \times 0.8 = 0.336$$

where, x1 has degree of 0.7 in S1, x2 has degree of 0.6 in B2 and y has degree of 0.8 in CE.

- **Step 4**: Create combine Fuzzy rule base. This step avoids conflict rules and two identical rules exist in the combine fuzzy rule base. Conflict rules are rule which have same IF part but different THEN part. To solve conflict rules is to accept only the rule from conflict group that has maximum degree. We use table – lookup presents a fuzzy rule base. We fill cells of rule base by the rules; if there is more than one rule on one cell of fuzzy rule base, use the rule that has maximum degree.

- **Step 5**: Determine mapping bases on the Combined Fuzzy Rule Base. We use the following defuzzification strategy to determine the output control y for given input (x1,x2). First, we combine the antecedents of the ith fuzzy rule using product operations to determine the degree, of the output control corresponding to (x1,x2); that is,

$$\mu^{i}_{O^i} = \mu_{I^i_1}(x_1) \times \mu_{I^i_2}(x_2) \tag{3}$$

where Oi denotes the output region of rule i, and denotes the input region of rule i for the jth component; for example, rule 2 gives

$$\mu^{1}_{CE} = \mu_{S1}(x_1) \times \mu_{B2}(x2) \tag{4}$$

Then we use the center average defuzzification formula to determine the output

$$y = \frac{\sum_{i=1}^{M} \mu^{i}_{O^i} \times y^{-i}}{\sum_{i=1}^{M} \mu^{i}_{O^i}} \tag{5}$$

where y^{-i} denotes the center value of region Oi and M is number of rule in the combined fuzzy rule base.

IMPLEMENTATION

To implement this algorithm, we use Hash Map (HM) in Collection Frame Work of Java to implement fuzzy rule base. Each element of HM consists of two parts. The first part is the key and second part is the value correspond this key. To avoid conflicting rules, we choose the key that is the antecedent of the original rule. For easy defuzzification when comparing degree of two rules that have same IF parts, we choose value of object that is created from class Rule as illustrated by figure 6.

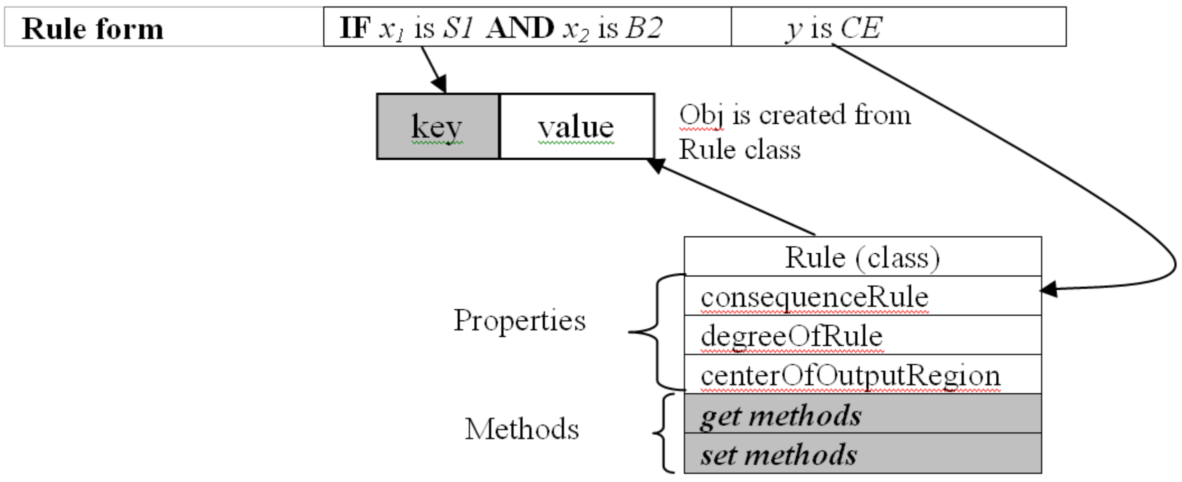

Figure 6. Implementation of Fuzzy rule based system

For better understanding of this algorithm, see fragment of pseudo-program below:

```
------------------------------------------------------
Input: Training data set
Output: Fuzzy rule base system
------------------------------------------------------
Assign fuzzy rule base (hm) is empty.
i=0;
WHILE NOT a last sample in training set DO
          Find antecedents of the rule and calculate all properties of object that is created from Rule
          class from input – output pair ($x_{1+i}$, $x_{2+i}$, ... ,$x_{n+i}$→$x_{n+i+1}$).
          IF hm contains the antecedents of the rule as the key THEN
                    GET object FROM hm with the key equals to antecedents of the rule
                    IF degree of current rule > degree of stored rule THEN
                              UPDATE conclusionRule, degreeOfRule and centerOfOutputRegion of stored
                              rule with value of current rule
                    END_IF
          ELSE
                    APPEND current rule to hm
          END_IF
          i = i + 1
END_WHILE
----------------------------------------------------------------
```

Figure 7. The pseudo code of training process

CONCLUSION

Described application of fuzzy sets resolved all problems with the OCR reading texts on the PC screen. The system is fully proven. Nowadays, we use it to read and record the values of www.forex.com .

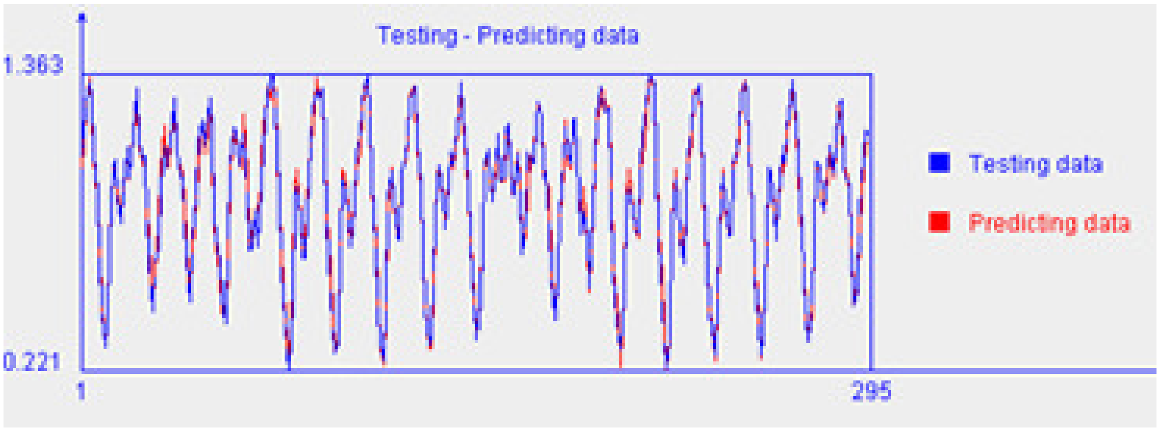

Figure 8. Predicting CZK to USD ratio

Similar approach with neural network, calculated by identical Java library, shows an interesting probability of success, when used for predicting time series of data. Figure 8 above shows result of experiment with CZK to USD ratio.

As an example, we took the relation of CZK against the USD in time. We took 700 historical values in the period from 04.06.2008 to 22.03.2011 and we used them as a training signal. Then, after the network was fully trained, we made prediction of 300 data points and compared them with the actual course. The results are shown in Figure 8.
It can be seen that the results are very promising.

Příspěvek má vztah k projektu s podporou z grantu FR-TI2_643_v2.

LITERATURE

[1] Verner, M., Kokeš, J.: Napojení expertního systému na internet. Sborník konference Inteligentní systémy pro praxi, Bohdaneč, 2006, AD&M Ostrava, str. 47-50. ISBN 80-239-6535-2.
[2] Kokeš, J.: Hilbert-Huangova transformace aplikovaná v expertním systému. Sborník semináře Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky, pořadatel Ústav přístrojové a řídicí techniky ČVUT v Praze, Jindřichův Hradec, 28. až 29. května 2009
[3] Nghien N. B. – Kokeš, J.: Implementation of Fuzzy logic system using Table Lookup Scheme by Java language to predict time series data International Review of Automatic Control (I.RE.A.CO.), May 2008
[4] Wang, Y.F., “Predicting stock price using fuzzy grey prediction system”, Expert system with application 22 (2002) 33-39.
[5] Smejkal, J.: “Virtuální úloha pro analýzu drsnosti povrchu optickou cestou”, Diploma work, FME CTU in Prague, 2011.