Software System for Optical Recognition and Symbolic Differentiation of Mathematical Expressions

Ivan Perić, Marko Jocić, Đorđe Obradović
* University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia
{ivanperic, m.jocic, obrad}@uns.ac.rs

Abstract – In this paper we propose a method for optical recognition and symbolic differentiation of printed mathematical expressions. Many areas of science use differential calculus as a mechanism to model and formally represent the dynamics of many processes. Function derivation process can be pretty exhausting, especially if we need to use it too often. Basically, it's based on the set of derivation rules which makes it suitable for automation and we will use that fact. Furthermore, to make the derivation process even faster, we will try to recognize the mathematical expression by taking a photograph of it, so the user will not have to enter complicated mathematical expressions. Maximum potential of this kind of system can be achieved by using it on a mobile platform where the user already has a mechanism to take photographs of mathematical expressions and differentiate them wherever he is. It could be used in scientific purposes to speed up the calculation process. On the other hand, it could be very helpful to students, who could check if their calculations are valid.

I. INTRODUCTION

Computer Vision is a field of artificial intelligence, widely used in robotics, applied medical sciences, physics, industrial processes and many more. It includes methods for acquiring, processing, analyzing, and understanding images in order to produce numerical or symbolical data about a real world. This way, machines are allowed to perceive and "understand" a world around them. Continuous enhancement in a field of computer hardware gives a boost to other fields of computer science, especially the ones that are computationally demanding and at the same time need to be executed in a real-time. That's the reason why computer vision has become very active field of research in previous years. One of the computer vision fields is the optical character recognition (OCR). OCR can be applied in many situations. Optical recognition of mathematical constructions is a special class of OCR problems.

The system proposed in this paper is divided into four subsystems, where each of them encapsulates logically similar operations.

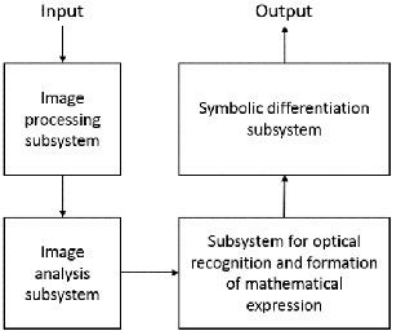

Figure 1 - System architecture

This paper contains formal specification and implementation steps of the proposed system. Computer vision techniques, machine learning, mathematical logic and numerical algorithms are used in the implementation.

II. RELATED WORK

There are many software systems for the optical character recognition. Mathematical expressions represent one problem type in this field of research. The first challenge in optical recognition and symbolic differentiation of mathematical expressions is the image segmentation process. Many different techniques can be used, but if the accuracy is important, those techniques tend to be more complicated and the speed of the system is be reduced. Researchers from Assam University and Manipur University in India proposed an approximation method for adaptive thresholding technique. [1] They managed to keep binarization process times around 200ms, while regular adaptive thresholding techniques went up to 13s for large window sizes. This means that their approach is more than 50 times faster. This is very important for systems that are executed in real-time environments or on mobile platforms, due to limited hardware resources. The second problem is the fact that mathematical expressions are not regular text. It's not easy to process spatial relationships between parts of the mathematical expression. Their complexity can be very high. Alan Sexton from the University of Birmingham in UK proposed one method for spatial relationships processing. [2] He did horizontal and vertical projections on the mathematical expression and tried to divide it. The problem he didn't solve was the fact that mathematical expressions can be rotated. In that case projections won't work. In the example of photographed mathematical expression, the probability of expression being rotated is very high. One possible solution to this problem will be presented in this paper. Commercial products for mathematical problem solving usually just solve mathematical equations, but don't incorporate OCR techniques. For example, there is Wolfram Alpha [3], which can solve equations and parse hand written symbols. This type of problem solving is very helpful to many people and that's why these commercial systems are constantly improving.

III. PRELIMINARIES

Approach proposed in this paper uses adaptive thresholding image segmentation techniques, morphological operators, least-squares approximation algorithm to make linear regression, artificial neural networks with back-propagation training algorithm and

symbolic logic incorporated into Prolog programming language.

A. *Image segmentation using adaptive threshold*

Threshold is the image segmentation method in which the specific *threshold* is calculated, and then every pixel value is compared to it. Usually there is one threshold value and all pixels are divided into two groups – below threshold value and above threshold value. Then the one group is classified as background, and the other one as a content of interest. This process is called **binarization**. Thresholding methods are usually applied to grayscale images, which is also the case in this paper. Threshold is a value that must be set or calculated. Most of the methods used to calculate threshold use **histogram** as a data source. Histogram of the grayscale image represents the number of pixels of the specific value on the image. Pixel values of the grayscale image are between [0,255], where 0 is black – zero light, 255 is white – maximum light. As it is shown in the Figure 2, threshold is calculated somewhere around 75. It can be observed that there is a larger number of pixels with low lightning, which means that the whole picture was in darker tone.

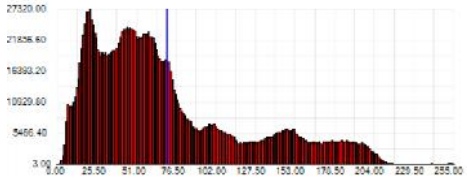

Figure 2 - Histogram for threshold calculation

The main problem with the thresholding methods can be seen if the picture is not uniformly illuminated. When you make a photograph with a flash turned on, there is a possibility that this will happen. One part of the image will be very bright, and the other one will be very dark (Figure 3).

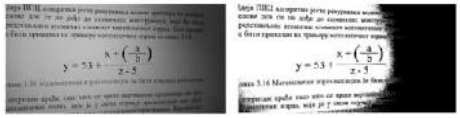

Figure 3 - Global threshold problem

This problem can be fixed by using **adaptive threshold** techniques. Adaptive threshold uses a locality principle, which means that the threshold is calculated for many image segments independently. By this approach results can be improved, but the binarization process will be more complicated and more time-consumable. Threshold calculation needs many statistical characteristics of the input image, which are time-consumable by themselves, and they have to be calculated many times – for each local threshold. An approximation will be used in this paper, and the performance problem will be minimized.

B. *Morphological operators – dilation and erosion*

Definition. Dilation of the set ***X*** by structure element ***S*** is defined as a binary image which represents the set of points [m,n] such that after translation of the structured element to point [m,n] intersection of the sets S_{mn} and ***X*** is not an empty set:

$$S \oplus X = \{m, n: S_{mn} \cap X \neq \emptyset\}. \; [4]$$

Dilation is a morphological operation based on a logical operator OR between the neighboring pixels. If at least one point the neighborhood is black, the current point will be black. [5]

Figure 4 - Dilation operator

Definition. Erosion of the set ***X*** by structure element ***S*** is defined as a binary image which represents the set of points [m,n] such that after translation of the structured element to point [m,n] the whole structured element S_{mn} is included in the set ***X***:

$$X - S = \{m, n: S_{mn} \subseteq X\}. \; [4]$$

Erosion is a morphological operation based on a logical operator AND between neighboring pixels. If all the points in the neighborhood are black, the current point will be black. [5]

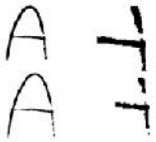

Figure 5 - Erosion operator

C. *Artificial neural networks and Back-Propagation algorithm*

Definition. Artificial neural network can be observed as an artificial replica of the human brain and tends to simulate the learning process. However, they are just a rough approximation of biological neural networks. Artificial neural networks are designed to do a nonlinear mapping of input data to some output data. Formally, artificial neural network can be represented as:

$$f_{ANN}: R^l \rightarrow R^k.$$

Artificial neural network consists of a set of simple process elements, called neurons. Those neurons represent a model of biological neurons. Neural network stores the knowledge in connections between neurons. This knowledge is calculated in a neural network learning process (training).

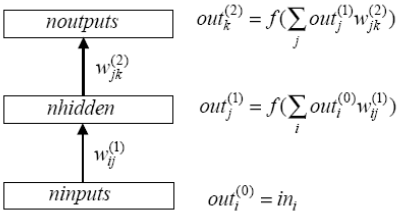

Figure 6 - Multilayer perceptron

Neural network proposed in this paper is trained by a back-propagation algorithm. This algorithm propagates an error value backwards in a neural network topology. Neural network used here is a feed forward neural network and consists of 5 layers. Input layer has 64 neurons, three hidden layers consists of 100 neurons each, and the output layer has 49 neurons. The parameter μ, which represents the speed of learning, is set to 0.05, and the learning process is completed after 10.000 iterations or if the error value drops below 10^{-4}.

The main problem in multilayer perceptron training process in the fact that the target value is known only in the output layer. Algorithm must be able to correct all weights by the same minimization criteria.

Multilayer perceptron training tends to minimize error function

$$E\left(w_{ij}^{(n)}\right) = \frac{1}{2}\sum_{p}\sum_{j}\left(targ_j^p - out_j^{(N)}(in_i^p)\right)^2$$

and corrections given by the gradient formula are used:

$$\Delta w_{kl}^{(m)} = -\eta\frac{\partial E\left(w_{ij}^{(n)}\right)}{\partial w_{kl}^{(m)}}$$

In the first step, training set is formed and the neural network architecture is defined. After neural network formation, initial values of weights between neurons must be set. Values are usually defined in a random manner. Also, error function $E(w_{ij})$ must be chosen, and also a training speed η. In the second step, the input data vector is sent to the neurons in neural network's input layer. In the third step the neural network output is calculated. The whole weight correction process is represented as an optimization process of the error function.

Output value formula for the neural network with multiple hidden layers looks like this:

$$delta_k^{(N)} = \left(targ_k - out_k^{(N)}\right)\cdot f'\left(\sum_j out_j^{(1)} w_{jk}^{(N)}\right) = \left(targ_k - out_k^{(N)}\right)\cdot out_k^{(N)}\cdot\left(1 - out_k^{(N)}\right)$$

formula for the hidden layers:

$$delta_k^{(n)} = \left(\sum_k delta_k^{(n+1)} w_{lk}^{(n+1)}\right)\cdot f'\left(\sum_j out_j^{(n-1)} w_{jk}^{(n)}\right) = \left(\sum_k delta_k^{(n+1)} w_{lk}^{(n+1)}\right)\cdot out_k^{(n)}\cdot\left(1 - out_k^{(n)}\right)$$

and a formula for weights correction:

$$\Delta w_{hl}^{(n)} = \eta\sum_p delta_l^{(n)} out_h^{(n-1)}$$

Algorithm continues by correcting all weights, for every input pattern from the training set. This step is repeated until the error is not low enough. When error drops below certain value, training process is finished. [6] [5]

D. Prolog

Many problems in the real world are calculated much faster by machines than by the human brain, but some of them are not. Sometimes it's easier to represent a problem by a set of rules and facts, which is more similar to a human perception. Prolog is a declarative programming language that has a knowledge base, which consists of a set of **facts** and a set of **rules**. Fundamentals of Prolog are based on a predictive logic. Prolog is very suitable for problem solving where problem consists of objects and relations between them.

In the first step, the knowledge base is formed. When a query is passed to Prolog interpreter, decision tree is made by using facts and rules from knowledge base. Then the result is calculated by following the tree branches.

IV. PROPOSED ALGORITHM

Software system proposed in this paper is divided into four subsystems (image processing subsystem, image analysis subsystem, subsystem for optical recognition and formation of the mathematical expression and a subsystem for symbolic differentiation). Every subsystem does a special set of tasks. Input to the system is a digital image of the mathematical expression, and an output is a derivative of that mathematical expression in a form of text.

1. *Image processing subsystem*

This subsystem is the first in the series of subsystems, and it's input is also an input of the whole system.

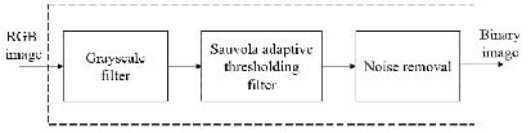

Figure 7 - Image processing subsystem block diagram

Color image, represented in a RGB color system, first needs to be converted into grayscale color image. Mathematical expressions are usually printed on a very bright surfaces (white paper), and the text is usually black, so it's enough to look every pixel's luminosity level. Grayscale color space shows the luminosity level of every pixel and that's why this approximation is applied. In the next step, grayscale image is binarized and every pixel is classified as a background or a content of interest. This is done by adaptive thresholding technique. The main problem of adaptive threshold techniques is a performance time. Adaptive threshold approximation will be proposed in this paper, by using **integral images**.

Definition. An integral sum image **g** of an input image *I* is defined as an image in which the intensity at a pixel position is equal to the sum of the intensities of all the pixels above and to the left of that position in the original image. [1]

Integral image intensity at *(x,y)* can be calculated as:

$$g(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{y} I(i,j)$$

The integral sum image of any grayscale image can be efficiently computed in a single pass as:

$$g(1,y) = I(1,y) + g(1,y-1), \quad y = 2..n$$

$$g(x,1) = I(x,1) + g(x-1,1), \quad x = 2..m$$

$$g(x,y) = I(x,y) + g(x,y-1) + g(x-1,y) - g(x-1,y-1), \quad x = 2..m, \quad y = 2..n$$

Definition. In Sauvola's binarization method [1], the threshold ***T(x,y)*** is calculated using the mean ***m(x,y)*** and standard deviation ***δ(x,y)*** of the pixels within a window of size ***ω×ω*** as:

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{\delta(x,y)}{R} - 1\right)\right]$$

where R is the maximum value of the standard deviation (R = 128 for a grayscale document), and k is a bias, which takes positive values in the range [0.2, 0.5]. [1]

However, standard deviation is calculated for every local thresholding window and that consumes a lot of time. That's why mean standard deviation is used, and the process is much faster. After this approximation, threshold formula looks like this:

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{\partial(x,y)}{1 - \partial(x,y)} - 1\right)\right]$$

where $\boldsymbol{\partial(x,y) = I(x,y) - m(x,y)}$ is the mean deviation, and **k** is a bias which can control the level of adaptation varying threshold value. Also $k \in [0,1]$. [1]

Calculation of the *m(x,y)* requires that the local sum ***s(x,y)*** is calculated before.

Definition. The local sum ***s(x,y)***, which is the center of the local window of size $\omega \times \omega$ of an image *I* is the sum of all the pixel intensities within the local window. It can be calculated in two passes as:

$$s(x,y) = \sum_{i=x-c}^{x+c} \sum_{j=y-c}^{y+c} I(i,j)$$

where $c = \frac{\omega-1}{2}$, since ω is an odd number. [1]

Once we have the integral sum image **g**, the local sum ***s(x,y)*** of any window size can be computed simply by using two addition and one subtraction operations without depending on window size as:

$$s(x,y) = [g(x+d-1, y+d-1) + g(x-d, y-d)] - [g(x-d, y+d-1) + g(x+d-1, y-d)]$$

where $d = round\left(\frac{\omega}{2}\right)$.

Definition. The local arithmetic mean ***m(x,y)*** at *(x,y)* is the average of the pixels within the window of $\omega \times \omega$ of the image I. It can be calculated as:

$$m(x,y) = \frac{s(x,y)}{\omega^2}$$

In this way the local mean can be calculated efficiently in a single pass without depending on local window size using integral sum image. [1]

Adaptive thresholding gives very good results without a lot of noise after binarization process. However, there is a noise removal filter to handle those situations. In this case it was enough to use just morphological operators – dilation and erosion.

2. *Image analysis subsystem*

When binary image is filtered, regions of interest can be found. Every region of interest is a list of black pixels.

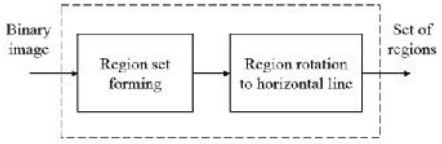

Figure 8 - Image analysis subsystem block diagram

Region formation starts with the one pixel that belongs to it, and then a ***Depth first search – DFS*** is performed to collect all the other points of that region. The process continues until all pixels are classified to their corresponding regions.

If the image is rotated, regions will be rotated too. This is going to be a problem in the next subsystem, and that's why a rotation angle of the whole expression needs to be calculated. Then, every region will be rotated and placed on a horizontal line.

First we need to find a rotation angle **φ**. Region centers will be taken, and then linear regression is used to find a line that best fits an angle of the expression.

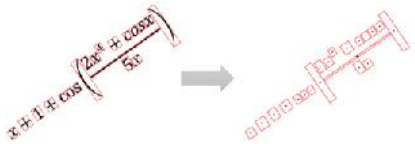

Figure 9 - Preparation for the approximation

When regression line is found, it's easy to find a rotation angle φ.

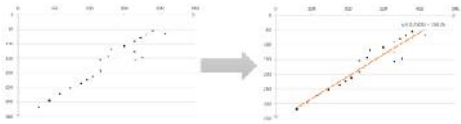

Figure 10 - Angle approximation process

$$\varphi = \tan^{-1}(k).$$

Now it's possible to rotate the expression. Point of the rotation is going to be an intersection of a y-axis and a

Figure 11 - Rotation of the expression

regression line. If a negative value is calculated, all the regions are translated to the positive part of y-axis so the digital image coordination system is preserved. [7]

Every point of every region is rotated around rotation point. Original point with coordinates ***(x,y)*** is transformed into point with coordinates ***(r_x,r_y)***, where

$$r_x = \cos(\varphi) * (x - m_x) - \sin(\varphi) * (y - m_y) + m_x$$

$$r_y = \sin(\varphi) * (x - m_x) + \cos(\varphi) * (y - m_y) + m_y$$

3. *Subsystem for optical recognition and formation of mathematical expression*

This subsystem is very important, because it will do the optical recognition of every symbol in the mathematical expression, and it will do the processing of spatial relationships between regions. These two functionalities will be placed into two different modules.

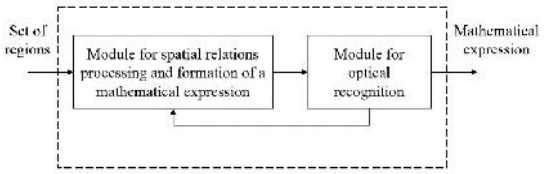

Figure 12 - Block diagram of the subsystem for optical recognition and formation of mathematical expression

Module for optical recognition consists of a feed-forward neural network which is trained by back-propagation algorithm. This module functionalities will be used by the other module.

Module for spatial relationships processing and formation of a mathematical expression makes a tree that represents a mathematical expression, and then transforms it to plain text. The projection profile cutting algorithm – PPC, is used in this module to process spatial relationships of regions.

PPC algorithm recursively divides the image space and tries to get atomic elements of the mathematical expression. Along all the way, a tree that represents a mathematical expression is created. PPC algorithm does a vertical projection and divides a space into vertical segments. Then, every vertical segment is divided horizontally. Now every horizontal subsegment is divided vertically and so on. The algorithm is finished when every subsegment has only one element.

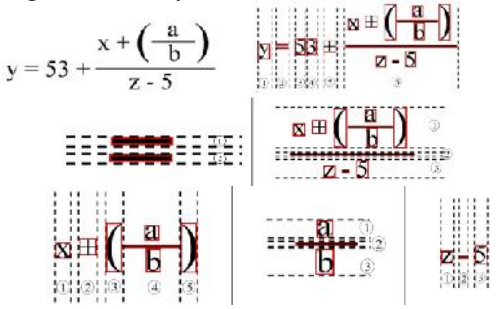

Figure 13 - PPC algorithm

PPC algorithm creates a parsing tree that represents the mathematical expression. A parsing tree for the expression shown above is shown in a Figure 14.

This tree has to be transformed into text in order to be sent to the Prolog interpreter. Here we propose a method in which the parsing tree is modeled by Composite design pattern.

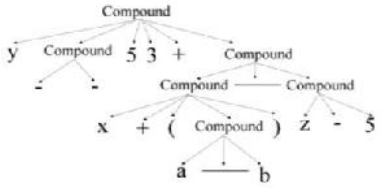

Figure 14 – A parsing tree of mathematical expression

The instance of a parsing tree is a Singleton object which has a reference to the root node. Every node, including the root, has references to all it's successors. Every node can be an instance of CompoundExpression, which represents a complex structure in a tree, or an instance of AtomicExpression, which represents just one element in the mathematical expression. Atomic expressions are leafs in the parsing tree, and compound expressions can be divided into other tree nodes. Compound expression can encapsulate many other compound or atomic expressions. This recursive structure allows this parsing tree to fit any complexity of mathematical expression. Atomic expressions are transformed into text just by returning the symbol that represents that tree node. In the compound expression, every child node is transformed into text independently, surrounded by parenthesis to preserve operator priorities is case of fractions and functions with complex arguments.

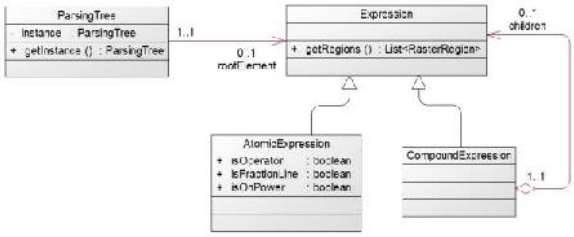

Figure 15 - Parsing tree model by Composite design pattern

4. *Symbolic differentiation subsystem*

This subsystem is designed to do the symbolic differentiation. This functionality is achieved by using Prolog programming language. Knowledge base consists of a differentiation rules for main function types, as well as rules for differentiation of complex functions and differentiation of the sum, subtraction, division and multiplication of two functions. This subsystem receives a mathematical expression in a form of text, and returns differentiated mathematical expression, also as text.

V. RESULTS

Software system proposed in this paper is tested from two different perspectives – accuracy and performance. Performance testing is done on two different machines.

	Machine 1	Machine 2
Processor	Intel Core i7 (QuadCore) @ 2.1GHz	Intel Celeron (SingleCore) @ 2.1GHz
RAM	8 GB	2GB
Graphic card	AMD Radeon 7670	Intel HD Graphics 4000

Table 1 - Testing machines specifications

a) Accuracy evaluation

Evaluation of the accuracy is done in two phases. In the first one, simple mathematical expressions are evaluated (simple fractions, no rotation, no square roots, etc). These expressions are simple and system should work accurately.

Mathematical expression	Accuracy
$\mathbf{sinx + cos5x}$	100%
$\mathbf{(x+5)*(60*x)+50}$	100%
$\mathbf{x+7*\left(\frac{4}{5}+3*x\right)}$	100%

Table 2 - Simple mathematical expressions accuracy evaluation

Now more complex mathematical expressions will be evaluated (complex fractions, rotated expressions, complex functions, etc). Some expressions are rotated, so the rotation algorithm will be evaluated.

Mathematical expression	Accuracy
$sin_x + cos 5_x$	100%
$\sin\frac{180x}{3} - 5*x+5$	100%
$(x+5)*(60*x)+50$	100%
$x+7*\left(\frac{4}{5}+3*x\right)$	100%
$\sqrt{2x+1} - \frac{2x}{7}$	90% (1)
$x^3 + 2x^2 + x + 2 + 2\tan x$	100%
$x+1+\sin\left(\frac{2x^3+\sin x}{5x}\right)$	100%
$x^2 + 2x + \frac{5+x^2}{x+5x}$	94% (2)
$\sqrt{x+3} + \frac{x}{y} - \sqrt{x^2+2}$	100%

Table 3 - Complex mathematical expressions accuracy evaluation

(1) Subtraction sign (-) recognized as a letter „i“.

(2) First element x^2 recognized as x^x.

After accuracy evaluation, it can be observed that system accuracy is very high, and somewhere around 98%. All errors were made by neural network, and other subsystems worked well.

b) Performance evaluation

Performance evaluation is done on the expression set from the second phase of the accuracy testing (complex mathematical expressions) in the same order they appeared in the Table 3. The slowest performance time on the old generation machine was around 250ms, and around 75ms for a modern machine. System can be classified as really fast. Performance time of 250ms average user sees as almost instant response from the system. All performance times are shown in a Figure 16.

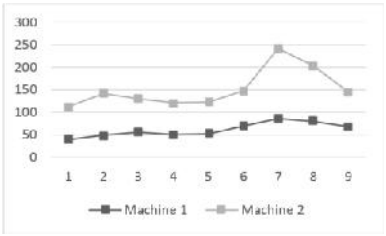

Figure 16 - Performance test results

VI. CONCLUSION

In this paper we proposed a system for optical recognition and symbolic differentiation of mathematical expressions. Many algorithms from computer vision, artificial intelligence, calculus and mathematical logic were used and system resulted to be fast and accurate. This type of system would certainly be a great help to people who do calculations every day, and also to students who could check their calculations. Future research would provide the proposed algorithm of handling images made from perspective, where parts of the mathematical expression are skewed. Furthermore, Prolog segment could be improved to be able to return a step by step solution of the symbolic differentiation process.

REFERENCES

[1] R. T. Singh, S. Roy, T. Sinam and M. K. Singh, "A new local adaptive thresholding technique in binarization," in *Interantional Journal of Computer Science Issues, Vol.8, Issue 6, No2, ISSN(Online): 1694-0814*, 2011.

[2] A. Sexton, A. Raja, M. Rayner and V. Sorge, "Towards a Parser for Mathematical Formula Recognition," University of Birmingham, Birmingham, United Kingdom, 2014.

[3] "Wolfram Alpha," [Online]. Available: (http://www.wolframalpha.com/).

[4] S. Graovac and V. Papić, "Digitalna obrada slike," Elektrotehnički fakultet, Univerzitet u Beogradu, Beograd, 2013.

[5] M. Jocić, Đ. Obradović, Z. Konjović and E. Pap, "2D fuzzy spatial relations and their applications to DICOM medical images," in *Intelligent Systems and Informatics (SISY), 2013 IEEE 11th International Symposium on Intelligent Systems and Informatics*, 2013.

[6] J. Freeman and D. M. Skapura, "Neural networks: algorithms, applications, and programming techniques," MA: Addison-Wesley, Reading, 1991.

[7] S. Weisberg, Applied Linear Regression, Minneapolis, Minnesota: Wiley, 2005.