Peculiarities of a symbol classification and recognition of expression structures system development

Taras Basyuk[1,*,†], Andrii Vasyliuk[1,*,†]

[1] *Lviv Polytechnic National University, Bandera str.12, 79013, Lviv, Ukraine*

Abstract

The article analyzes the existing methods and approaches used in the pattern recognition process, which made it possible to identify the peculiarities of their application and outlined the range of problems that arise. A symbol and relation classifiers were designed, which allowed to categorize symbols into one of four classes: 'small', 'descending', 'increasing', 'variable range', and to determine what is the relationship between them. A one-pass algorithm was implemented, including search with returns, which provided fast structure recognition of the expression using the algebra of algorithms. The software system was designed using the object approach and displaying the created diagrams in accordance with the UML language. Sequence and class diagrams are given, and their detailed description is carried out. The study presents the structure of basic actions, which allowed to display in more detail the process of classification and expression structures recognition. An applied software system has been developed using the Java object-oriented programming language, which implements the process of symbol classification and recognition of image structures. With the help of the developed system, it is possible to recognize the structure of expressions and classify symbols. As for now, the software solution works in the form of a prototype.

Further research will be directed to testing and improving the system, eliminating conflicts, and expanding functionality in accordance with the specified requirements.

Keywords

classification system, expression recognition, functional model, object-oriented design

1. Introduction

Computer vision is an activity in which statistical methods are used to obtain data and models, built using geometry, physics, and learning theory [1,2]. Computer vision is used quite widely, both in old fields (for example, mobile robot control, military applications, industrial surveillance) and in new ones (human/computer interaction, image retrieval in libraries, medical image analysis and realistic rendering of simulated scenes in computer graphics). A distinctive feature of computer vision is the extraction of descriptions from images or sequences of images. For example, such an aspect of computer vision as the detection of structure by motion allows you to get an idea from a series of images about how the camera moves and what is depicted in the picture. In the entertainment industry, similar methods are used to filter out

MoDaST-2024: 6th International Workshop on Modern Data Science Technologies, May, 31 - June, 1, 2024, Lviv-Shatsk, Ukraine

[*] Corresponding author.

[†] These authors contributed equally.

Taras.M.Basyuk@lpnu.ua (T. Basyuk); Andrii.S.Vasyliuk@lpnu.ua (A. Vasyliuk)

0000-0003-0813-0785 (T. Basyuk); 0000-0002-3666-7232 (A. Vasyliuk)

motion and build three-dimensional computer models of a building with structural integrity. With the help of a few photos, you can get good, simple, accurate and convenient models.

This technology is now at a special point of its development. Only relatively recently appeared the opportunity to create practical programs using the ideas of computer vision. The speed of modern digital devices and the possibility of parallel calculations enable the implementation of many algorithms for working with libraries of digital images. Thus, conducting important research and solving many everyday tasks (for example, organizing a collection of photographs, creating a three-dimensional model of the surrounding world, managing, and making changes to a collection of video recordings) is now possible with the help of computer vision methods.

Character recognition is performed by classic OCR methods, for example, using support vector methods, pattern matching. Analysis of structure is carried out by means of geometric reasoning based on implicit rules or grammatical rules.

These rules take character identity into account, and only a few studies have attempted to separate structure analysis from character recognition. Uncertainty in mathematical expressions, especially in handwritten ones, is accepted. This may be uncertainty about the structure symbol meaning, and only a few studies address this issue, maintaining multiple interpretations about the symbol meaning until the structure resolves the ambiguity [3]. Therefore, considering the mentioned features, an urgent task is analyzing symbol classification in known systems and recognition of expression structures with further development of a software tool for solving the given task.

1.1. Analysis of recent researches and publications

1.1.1. Analysis of known approaches to problem-solving

The problem created is not new, and therefore there are many studies related to it. We will analyze the most famous scientific works.

Richard Zanibbi's works analyze the baselines present in expressions [4,5]. Consider the dominant baseline, the line on which the expression will be written, and the nested baselines corresponding to the indices. In the first step, a tree is built based on these baselines. Then, using knowledge of mathematical notation properties for some tree transformations, one recognizes that the sequence 'sin' corresponds to the function of the same name, or that in lim $n \longrightarrow 1$, $n \longrightarrow 1$ is the argument of the limit. Further, with their help, a tree is obtained, which represents the content of the equation. To build a tree of baseline structures, you need to recognize individual symbols. To do this, character classes such as upper (e.g., 'd', 'b'), lower (e.g., 'y', 'p'), variable range (e.g., Σ, $\cup$), etc. are defined. Using these classes, areas around the symbol are defined, where subscripts, superscripts, etc. should be found, if any. The specified analysis of baselines was also used in other works [6,7].

In subsequent works, the ability to recognize subscripts and superscripts was improved using fuzzy regions [8,9]. At the same time, they were motivated by the fact that most ambiguities in handwritten mathematical expressions refer to the options' index/line and line/superscript. Additionally, using fuzzy logic allowed them to return to a ranked list of interpretations.

The online recognition method is performed in four stages: stroke recognition, structure recognition, character recognition, and subscript/superscript recognition [10]. After analyzing the baseline, they first recognize dominant symbols such as Σ or fractional lines, which they call parent symbols, and identify the child blocks in which the arguments must be found. The rest of the structure is then recognized using bounding boxes and vertical symbol positions.

Richard A. Tapia and Raúl Rojas González first obtain baselines like Richard Zanibbi and recursively build a minimal tree in which each node is a symbol. Next, the distance between the two symbols is determined depending on whether it is in the range of the other. In this method, they use different symbol classes (top, bottom, centered) and the regions around the symbol are built accordingly with clear boundaries. Studies also consider points of attraction according to the class of operators where the argument usually occurs. For example, the Σ operator usually has top and bottom arguments, so they place the attraction points in the middle of the upper and lower bounds of that symbol.

Qi Xiangwei also uses symbol dominance and minimal spanning tree but performs further analysis of symbol locations to construct groups of symbols more accurately [11,12]. In particular, the main character on the dominant baseline or group of characters representing the function name (for example, 'sin', 'lim') is determined.

Erik G. Miller and Emanuele Viola maintain ambiguity during the character recognition stage. They then calculate the probability of each character being of a certain class (lowercase letter, number, binary operator, etc.) and the probability of being an index, superscript, or linear expression according to character recognition and some location properties. Next, stochastic context-free grammar is used for syntactic analysis of the expression. The system uses a convex hull instead of bounding rectangles, models character positions using a Gaussian distribution, and uses an A-star algorithm to search the space of interpretations.

After character recognition, a set of rules is applied. Three sets of rules are applied, which are used consecutively. Mathematical rules are grammatical rules for parsing and understanding an expression. Sentiment-based rules are used to disambiguate the layout. Experience-based rules handle uncertainty in the semantics of an expression. The last two classes of rules are learnt and modified using feedback. At the end, they use the results to build a location tree and a semantic tree, and then turn them into a Latex tape.

In his writings, Walaa Aly conducts research, the essence of which is the correct recognition of indexes and superstructures [13]. They use normalized bounding rectangles as the main feature of the symbol. It has been proven that with normalized bounding rectangles along with special handling of irregular symbols, it is possible to effectively recognize the relationship using a Bayesian classifier. His work uses the InftyCDB dataset, which contains expressions extracted from more than 70 articles with different settings.

This article [14,15] explains the normalization of bounding rectangles in more detail. In addition, they define more character classes. While there are usually three main classes (top, bottom, small), they define six classes to handle variations in the positioning of some symbols. At the same time, the Bayesian classifier is also used, which allows obtaining positive results on a large sample.

Summarizing the research, it can be concluded that some recognition methods are based only on spatial considerations, such as baselines. Other methods use rule-based systems, such as grammar, and analyze an expression to interpret it [16]. Part of the algorithms considers knowledge of mathematical symbols and operators and their spatial properties. At the same time, not all studies use machine learning methods, so the classification of symbols and recognition of expression structures with the subsequent development of a software tool for solving the given task is an urgent issue.

1.2. The main tasks of the research and their significance

The purpose of this research is to develop a pattern recognition system, or rather a mathematical expression recognition system based on machine learning methods, where symbol classification and structure analysis are separate tasks. The conducted research will provide means for classifying symbols and determining the structure of expressions, including mathematical ones. To achieve the goal, the following tasks must be solved: to analyze the existing approaches and methods used in the process of symbol classification and recognition of the structure of expressions; to determine the main tasks that arise at the same time; to develop the structure and determine the main components of the recognized expression; to formulate an algorithm for symbol classification and recognition of an expression structure, and to carry out its mathematical description using the algebra of algorithms; to design a software system using the object approach; to construct an applied software system that implements the process of symbol classification and expression structure recognition.

The results of the study solve the actual scientific and practical task of analyzing the structure of an expression and classifying symbols, using a small amount of knowledge about the syntax of a mathematical expression.

2. Major research results

Mathematical expression recognition is a task in which an image representing a mathematical expression is interpreted by a computer so that it can be stored, processed, and reused. Expression recognition consists of two stages [17]:

- Symbol recognition: each foreground pixel in the image belongs to a symbol, and each symbol has a value in the expression and conveys some information.
- Pattern recognition: the two-dimensional layout of an expression obeys some rules, and each pattern corresponds to a specific value.

Character recognition. A character recognition task is a procedure by which each character is recognized and classified [18]. This is not an easy task due to the enormous number of characters, given that there is no dictionary of words like for text recognition. The same symbol can appear in different contexts, and it is sometimes important to distinguish between, for example, the summation symbol Σ and the Greek letter Σ. Some different symbols have the same shape, for example, p and P. Recognizing them correctly is not an easy task. Even more problems arise when dealing with handwritten expressions, such as the q-9 problem.

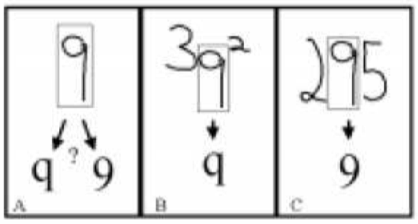


Figure 1: “q-9” problem

This task is usually performed by neural networks or support vector methods.

Structure recognition. Although the former task is easy to understand, pattern recognition can be difficult, depending on what level of interpretation needs to be achieved. In the research process, several goals were defined, along with the format in which the structure should be analyzed.

For example, for simple digitization in Latex, recognizing spatial relationships regardless of their meaning is usually sufficient. For example, in the example presented earlier: xi/x(i), it is not necessary to know that 'i' is a power in the first case and an index in the other. In Latex, x^i and x^{(i)} will simply be written.

It is necessary to understand the meaning of the expression. It is important to know that 1 + 2 is not only a sequence of characters in one line, but also an addition. In this case, the value of operators, both explicit (for example, +) and implicit (for example, multiplication by xy), must be known.

Sometimes a deeper analysis is required. In mathematical expressions, the meaning derives not from the symbols, but from what they represent. As mentioned earlier, the same function (or variable) can have a different name in different contexts. It is possible that f(x) and g(y) are the same in two different documents.

Thus, recognizing the structure of an expression is often a combination of location analysis and interpretation of what symbols and relations represent.

Complex mathematical symbols. As stated earlier, the range of symbols and rules used to write mathematical expressions is not fixed. Common symbols and structural rules are only a subset of an infinite number, since symbols and their new meanings can be invented at any time. Indeed, new fields of science continue to appear, bringing with them the need for new mathematical notations.

Quantum mechanics, for example, defined new uses for < and | to denote quantum states. The symbol <, which represents a comparison of numbers in 1 < 2, can be used as a kind of bracket in ⟨x, y⟩, or in the definition of some mathematical objects in (C, <). These are examples where existing symbols are reused for a different purpose. It is also possible to find completely new symbols. Therefore, it is necessary to consider this fact when recognizing mathematical expressions. A convenient way to solve this problem is to reduce the impact of the symbol. In most systems, character recognition determines how the structure will be recognized. As a result of the study, it turned out that character identification is not necessary for structure recognition. In addition, we believe that symbols can be classified using only their bounding box and context (Fig.2).

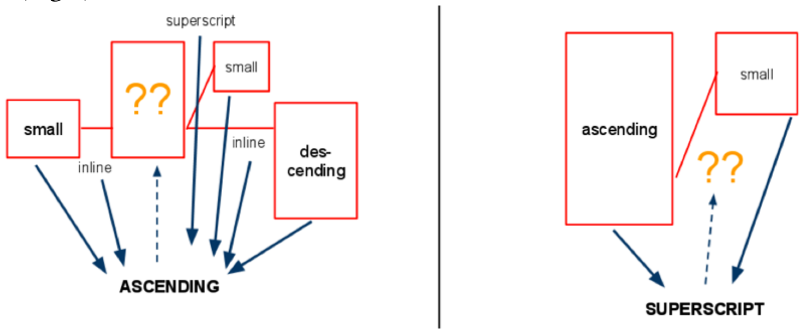


Figure 2: Hypotheses. Left: Context helps classify symbols. Right: character identification is not required for structure recognition

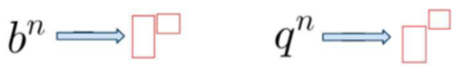


Figure 3: Variable superscript position

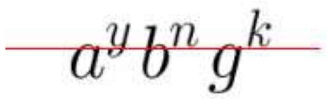


Figure 4: Arguments are on one line

When we analyze different symbols separately, we can see different spatial properties. For example, the relative position of the superscript, as seen in Fig. 3, will not be locally the same, with the symbol ‘b’ or ‘q’. However, at the global level, all superscripts must be written in one line (Fig. 4). At the same time, the input format is an image of a *handwritten mathematical expression*. Tables 1 and 2 show the character classes and relations used, respectively.

Table 1
Character classes used

Character classes	Example
Small symbols	a, e, r, u, o, s, m, x, c, n
Descending symbols	y, p, q, g
Rising symbols	A-Z, 0-9, t, d, h, k, l
Variable range symbols	∑, ∏, ∪ , ∩

Table 2
The relations used

Character classes	Example
Linear	xy, tan, 42, 10x, ∑n
Superscripts	p^n, b^a, x^y
Subscripts	p_a, b_n, x_3∑, ∏, ∪ , ∩

The goal of classification and recognition is to find the relationship between characters and determine their classes. This can be thought of as linking symbols together and adding information to an existing structure. At the beginning, a tree is created from the initial list.

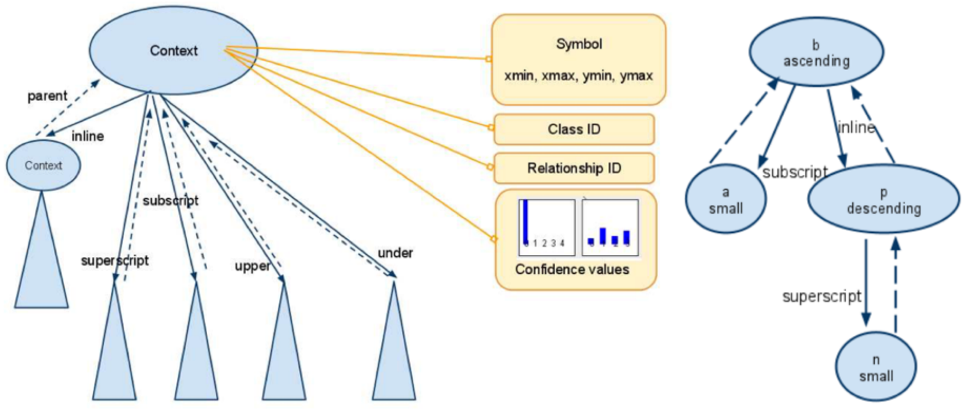


Figure 5: An abstract expression tree and an example of a typical expression tree

Therefore, in the process of research, it is necessary not only to recognize the structure, but also to try to find character classes using this structure. High order means a lot of contexts, which should make classification easier. However, as expressions become more complex, pattern recognition also becomes more difficult. That said, there is a trade-off between the challenges of complexity and necessity in context, so recognizing all kinds of symbols and relationships remains a challenge [19].

The next stage of the research was the system's design according to the object approach and the UML language [20]. A class diagram is a structural diagram type in the UML (Unified Modelling Language) used to visualize the structure of classes and their relationships in a software system [21-23].

It allows you to show classes, their attributes and methods, and relationships between classes. An extended description of the application of the class diagram includes the following aspects: requirements analysis (used to analyze system requirements, namely, understanding the structure of data and components allows you to analyze which classes are needed to implement the functionality of the system and how they are related to each other); communication with project participants (is an important communication tool between different project participants and allows developers, architects, managers and other interested parties to understand the structure of the system and its components); identifying potential problems (can help identify potential problems and flaws in the system design, such as circular dependencies between classes, excessive complexity, or insufficient modularity); documentation (serves as an important means of documenting the project, namely providing clear and specific information about the structure of the system for future developers and support personnel).

The class diagram of the designed system is presented in Fig. 6.

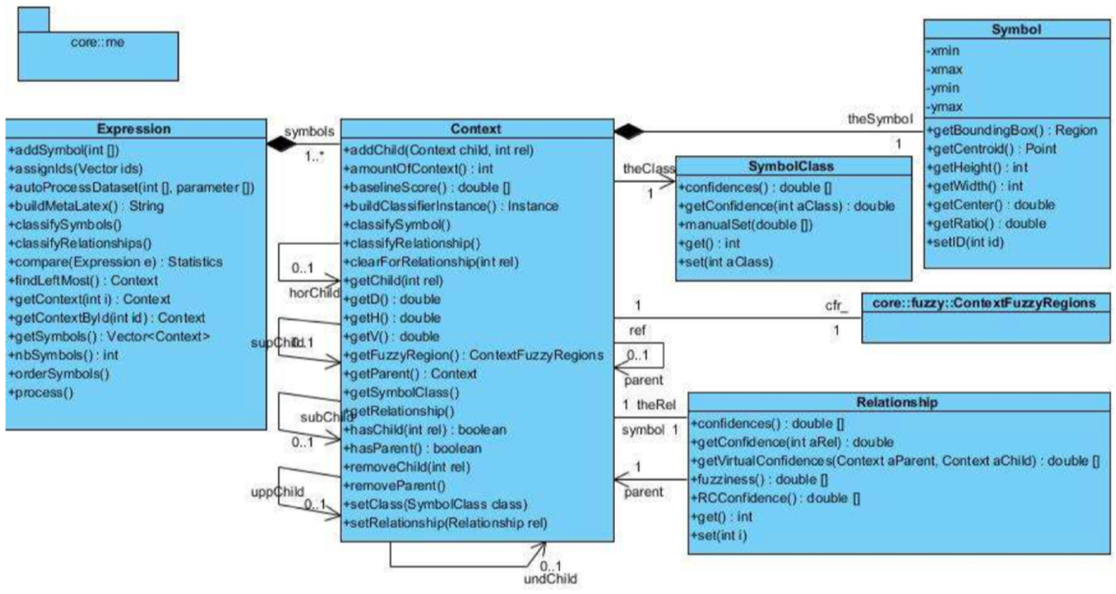


Figure 6: Class diagram

Expressions are represented by the symbols of which they are composed, considered in their context.

Symbol class. Objects of this class are representations of symbols. They contain information about the symbol itself, out of context. A Symbol object is created from a bounding rectangle. It provides geometric information such as the height, width, position, or center of a rectangle. The character identifier is particularly useful when testing the system.

The SymbolClass class. A SymbolClass object is a symbol classification. It contains the results of all the classifiers involved in the classification and the methods of providing the final probability values.

Relationship class. A Relationship object represents a classification of relationships that exist between a parent and subsidiary element. It stores the output of different classifiers and the final probability values.

The Context class. An object in this class is referred to by the word 'symbol' throughout the work. The described approach recognizes the structure and uses it to determine the character class. Therefore, the context of the symbol is as important as the symbol itself. A Context object is a node in a finite expression tree because it can be linked to other Context objects. A node also contains a SymbolClass object and a Relationship object. Its key role is to present the recognition result.

Expression class. An Expression object defines a list of all characters, sorted from left to right. Provides a straightforward way to get information about the entire expression, such as its interpretation, dimensions, leftmost character (the root of the tree after recognition), or to apply any action to all characters (for example, to classify characters).

The main stages of the specified process include:

- Image capture. An initial handwritten image of a mathematical expression is fed to the input of the recognition system.
- Image processing. The image is subject to processing such as downsizing, smoothing and noise removal to improve quality and facilitate further analysis.
- Structure analysis. The resulting text is divided into individual characters and groups of characters corresponding to subexpressions. This stage determines the structure of the mathematical expression and establishes relationships between symbols.
- Classification of symbols. Provides the classification of the symbols of the handwritten formula to the specified class.
- Context recognition. Given the context of the mathematical expression, such as the order of operations, actions with parentheses, and operator priority, the received symbols are analyzed to correctly recognize the expression.
- Generation of printed expressions. Based on the created expression tree, a printed mathematical expression is built in the appropriate format.

The next stage was the creation of a functioning model using the algebra of algorithms [24].

The first stage of the implementation of the algebra of algorithms is the description of unit terms and the synthesis of sequences, which is given below.

Formed uniterms: $S(i)$ – uniterm of capturing the input image; $P(s)$ - is the uniterm of image pre-processing; $A(s)$ - uniterm of structure analysis; $C(s)$ - is the uniterm of classification of symbols; $R(f)$ - uniterm of context recognition; $G(f)$ - uniterm of generation of printed expression; u_1 – check whether the symbol is classified. As a result of the use of the apparatus of the algebra of algorithms, the following sequences and eliminations were synthesized:

S_1 – the sequence of system operation when the symbol is classified:

$$S_1 = \overparen{\overparen{\left(\begin{matrix} S(i) \\ , \\ P(i) \end{matrix}\right., \left(\begin{matrix} A(s) \\ , \\ C(s) \end{matrix}\right.}, \overparen{R(f),\ G(s)}}$$

S_2 – the sequence of system operation when the symbol is not classified:

$$S_2 = \overparen{\overparen{\left(\begin{matrix} S(i) \\ , \\ P(i) \end{matrix}\right., \left(\begin{matrix} A(s) \\ , \\ C(s) \end{matrix}\right.}, \overparen{A(s),\ C(s)}}$$

L_1 – elimination of the symbol classification check:

$$L_1 = \underset{S_1\ ;\ S_2\ ;\ u_1 - ?}{\vdash\!\overline{\qquad\qquad\qquad}\!\dashv}$$

The next stage is the substitution of the corresponding sequences in the elimination.

$$L_1 = \underset{\left(\left(\begin{matrix} \left(\begin{matrix} S(i) \\ , \\ P(i) \end{matrix}\right. \\ , \\ \left(\begin{matrix} A(s) \\ , \\ C(s) \end{matrix}\right. \end{matrix}\right. \\ , \\ \left(\begin{matrix} R(f) \\ , \\ G(s) \end{matrix}\right. \right. ;\ \left(\left(\begin{matrix} \left(\begin{matrix} S(i) \\ , \\ P(i) \end{matrix}\right. \\ , \\ \left(\begin{matrix} A(s) \\ , \\ C(s) \end{matrix}\right. \end{matrix}\right. \\ , \\ \left(\begin{matrix} A(s) \\ , \\ C(s) \end{matrix}\right. \right. ;\ u_1 ?}{\vdash\!\overline{\qquad\qquad\qquad\qquad}\!\dashv}$$

As a result of using the properties of the algebra of algorithms, we subtract the common unit terms by the sign of the elimination operation and obtain the following formula of the algebra of algorithms:

$$\left(\begin{matrix}\left(\begin{matrix}\left(\begin{matrix}S(i)\\,\\P(i)\end{matrix}\right.\\,\\\left(\begin{matrix}A(s)\\,\\C(s)\end{matrix}\right.\end{matrix}\right.\\,\\\vdash\!\!\overline{\qquad\qquad\qquad}\!\!\dashv\\\left(\begin{matrix}R(f)\\,\\G(s)\end{matrix}\right.;\left(\begin{matrix}A(s)\\,\\C(s)\end{matrix}\right.;u_{\bar{1}}\,?\end{matrix}\right.$$

The next stage was the development of the system, using modern software tools. The developed system is presented as a desktop application. The application was created using Java programming in the macOS operating system environment [25,26]. The power and flexibility of the Java language make it an optimal choice for programmers in various fields. From support for object-oriented programming to platform-independent capabilities through JVM to automatic memory management, Java provides developers with powerful tools that enable them to efficiently build complex applications [27]. With the developed system's help, it is possible to classify symbols and determine the expression's structure. Structure recognition is performed by the parentAndRelate() method of the RelationshipFinder object. It uses classifiers as described earlier and two data structures. To easily retrieve the last characters on each baseline, we defined the BaselineStructure class. The most recently viewed symbols are available via the stack.

A BaselineStructure object. The purpose of the BaselineStructure object is to access the symbols on each baseline without having to go through the entire expression tree. It consists of a symbol and a list of nested baselines.

The classes that implement the graphical interface are in the 'GUI' package and are built using web technologies. They can be divided into two components [28-30]:

- An input component that facilitates quick input of mathematical expressions.
- An output component that displays the recognition result in the form of a two-dimensional image and a Latex expression.

DrawView class. An object of the DrawView class allows the user to draw (handwrite) a mathematical expression.

ResultImageView class. An object of the ResultImageView class is used to display a mathematical expression as a two-dimensional image.

ResultTextView class. An object of the ResultView class is a block containing a text area that allows displaying results in a Latex message format [31,32].

At the same time, the symbol can only be a descendant of the last symbol of the baseline. For example, in ab^cd , c and d cannot be a subsidiary element of a, but they can be a subsidiary element of b, which is the last character on its base. When a nested subsidiary element is found, a new base structure is created containing an empty list of nested baselines and replaces the parent's base structure. The entire basic structure is simplified and ensures that the complexity does not grow exponentially with each symbol. When an unbuilt subsidiary element is found, a BaselineStructure is created and added to its parent's nested baselines.

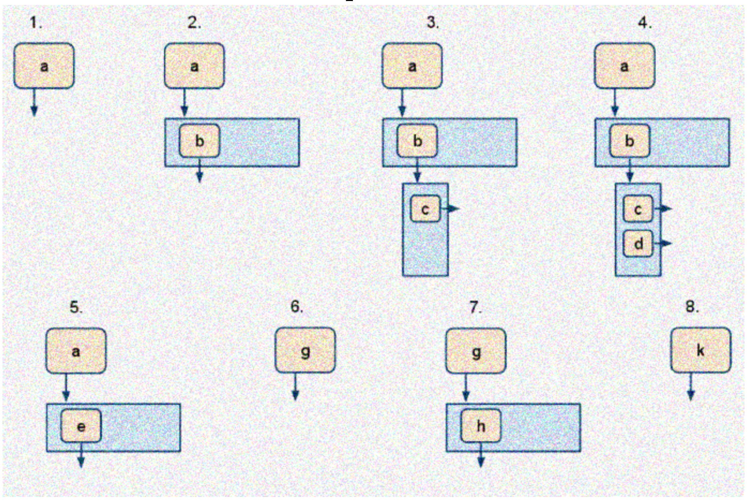
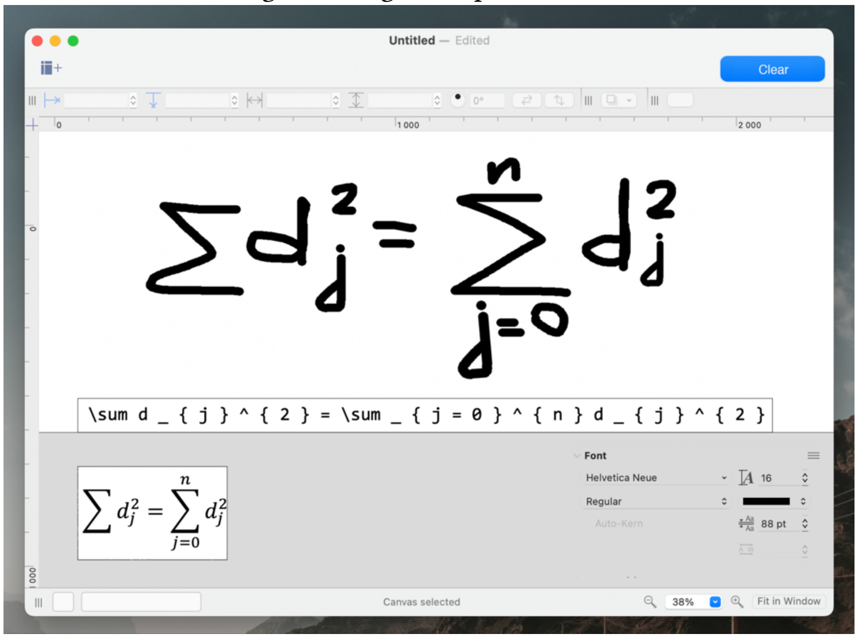


Figure 7: Baselines structure

In Fig. 8 it is shown the evolution of the main base structure $a^{b^c_d e}g_{h^k}$. *The stack of last characters.* Each time a symbol is a parent, the created base structure is added to the last symbol stack. This stack contains *BaselineStructure objects* rather than context objects to simplify updating the entire structure during the recognition process.



```
\sum d _ { j } ^ { 2 } = \sum _ { j = 0 } ^ { n } d _ { j } ^ { 2 }
```

$$\sum d_j^2 = \sum_{j=0}^{n} d_j^2$$

Font
Helvetica Neue
Regular
Auto-Kern
16
88 pt

Figure 8: Graphical user interface

The system was tested with 1, 2, 5, 7 and 10 iterations. The result was that 440 out of 570 characters (77.19%) were successfully classified. This means that, on average, the symbol is well classified, or the actual class is considered the second most likely. Accuracy is given in Table 3.

Table 3
Evaluation of symbol recognition

Symbol classes	Number of symbols	Correctly classified	Evaluation
Small symbols	225	212	94.22%
Descending symbols	68	57	83.82%
Rising symbols	218	137	62.84%
Variable range symbols	59	34	57.63%

There were 32 parentage recognition errors (94.39% success rate) and 52 relationship recognition errors (9.12% errors). At the same time, the recognition of connections is determined at the level of (0.797).

Conclusion

As a result of the research, the existing methods and approaches used in the process of classifying symbols and determining the structure of expressions were analyzed, which made it possible to identify the peculiarities of their application and outlined the range of problems that arise. An iterative algorithm was developed to use the mutual constraints between the structure and the type of symbols. A one-pass algorithm was implemented, which includes search with returns, which ensured fast recognition of the expression structure, and its representation was carried out using the algebra of algorithms. The next stage was the design of the software system using the object approach and the display of the created diagrams in accordance with the UML language. An applied software system was developed that implements symbol classification and expression structure recognition. With the help of the developed system, it is possible to recognize mathematical expressions by classifying symbols and analyzing the structure of expressions. At the current moment, the software solution works in the form of a prototype.

Further research will be directed to testing and improving the system, eliminating conflicts, and expanding functionality in accordance with the specified requirements.

References

[1] F. Olaoye, K. Potter, L. Doris. Computer Vision and Image Recognition Techniques. Journal of Scientific Conference Proceedings. -2024.pp.343-360.

[2] X. Luo. Review of object tracking algorithms in computer vision based on deep learning. Applied and Computational Engineering. 2024, vol. 32. pp.22-27. https://doi.org/10.54254/2755-2721/32/20230178.

[3] A. Antonacopoulos, C. Clausner, C. Papadopoulos, S. Pletschacher. Competition on Historical Book Recognition. 12th International Conference on Document Analysis and Recognition. – 2013. – №12. – pp. 1459–1463. https://doi.org/DOI 10.1109/ICDAR.2013.294

[4] T. Rowney, A. Iliev. Recognition of Handwritten Mathematical Expressions Using Systems of Convolutional Neural Networks. Serdica Journal of Computing. – 2023, vol. 17. pp.107-116. https://doi.org/10.55630/sjc.2023.17.107-116

[5] R. Zanibbi, D. Blostein, J. Cordy. Recognizing mathematical expressions using tree transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2002, vol.24. – pp. 1455–1467. https://doi.org/10.1109/TPAMI.2002.1046157

[6] N. Iskandar, W. Chew, S. Phang. The Application of Image Processing for Conversion of Handwritten Mathematical Expression. Journal of Physics: Conference Series. – 2023, vol.25. pp.401-409. 012014. https://doi.org/10.1088/1742-6596/2523/1/012014

[7] Y. Cao, Z. Xie, L. Li. Research on Identification of Handwritten Mathematical Formulas. – 2020. pp.1494-1500. https:/doi.org/10.1007/978-3-030-25128-4_183.

[8] R. Aggarwal, G. Harit, A. Tiwari. Symbol Spotting in Offline Handwritten Mathematical Expressions. – 2019. pp. 52-64. https:/doi.org/10.1007/978-981-13-9361-7_5.

[9] L. Zhang, D. Blostein, R. Zanibbi. Using Fuzzy Logic to Analyze Superscript and Subscript Relations in Handwritten Mathematical Expressions. – 2005. pp. 972-976. https:/doi.org/10.1109/ICDAR.2005.250.

[10] K. Toyozumi, T. Suzuki, K. Mori, Y. Suenaga. A system for real-time recognition of handwritten mathematical formulas. – 2001. pp.1059-1063. https:/doi.org/10.1109/ICDAR.2001.953948.

[11] Q. Xiangwei, P. Weimin, W. Yusup. The Study of Structure Analysis Strategy in Handwritten Recognition of General Mathematical Expression. – 2009. Vol. 2. pp.101 - 107. https:/doi.org/10.1109/IFITA.2009.169.

[12] L. Dong, H. Liu, X. Zhang. Synthetic Data Generation and Shuffled Multi-Round Training Based Offline Handwritten Mathematical Expression Recognition. Journal of Computer Science and Technology. -2022. Vol. 37. pp. 1427-1443. https:/doi.org/10.1007/s11390-021-0722-4.

[13] W. Aly, S. Uchida, M. Suzuki. Identifying Subscripts and Superscripts in Mathematical Documents. Mathematics in Computer Science. - 2008. Vol. 2. pp.195-209. https:/doi.org/10.1007/s11786-008-0051-9.

[14] W. Aly, S. Uchida, A. Fujiyoshi, M. Suzuki. Statistical classification of spatial relationships among mathematical symbols. Document Analysis and Recognition. – 2009. pp. 1350–1354. https://doi.org/ 0.1109/ICDAR.2009.90

[15] T. Basyuk, A. Vasyliuk, V. Lytvyn, O. Vlasenko. Features of designing and implementing an information system for studying and determining the level of foreign language proficiency// CEUR Workshop Proceedings. – 2023. – Vol. 3312: Modern Machine Learning Technologies and Data Science Workshop (MoMLeT&DS 2022): Proceedings of the Modern Machine Learning Technologies and Data Science Workshop, Leiden, The Netherlands, November 25-26, 2022. pp. 212-225.

[16] Z. Li, L. Jin, S. Lai, Y. Zhu. Improving Attention-Based Handwritten Mathematical Expression Recognition with Scale Augmentation and Drop Attention. – 2020. pp. 175-180. 10.1109/ICFHR2020.2020.00041.

[17] T. Basyuk, A. Vasyliuk. Peculiarities of an Information System Development for Studying Ukrainian Language and Carrying out an Emotional and Content Analysis // CEUR Workshop Proceedings. – 2023. – Vol. 3396: Computational Linguistics and Intelligent Systems 2023: Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems. Volume II: Computational Linguistics Workshop, Kharkiv, Ukraine, April 20-21, 2023.pp. 279–294.

[18] A. Vasyliuk, T. Basyuk, V. Lytvyn. Design and Implementation of a Ukrainian-Language Educational Platform for Learning Programming Languages// CEUR Workshop Proceedings. – 2023. – Vol. 3426: Modern Machine Learning Technologies and Data Science Workshop (MoMLeT&DS 2023): Proceedings of the Modern Machine Learning Technologies and Data Science Workshop, Lviv, Ukraine, June 3, 2023. pp. 406–420.

[19] G. Wadaskar, V. Bopanwar, P.a Urade, S.i Upganlawar, P. Shende. Handwritten Character Recognition. International Journal for Research in Applied Science and Engineering Technology. -2023. Vol.11. pp. 508-511. https:/doi.org/10.22214/ijraset.2023.57366.
[20] P. Mrzyglocki, UML Summarized: Key Concepts and Diagrams for Software Engineers, Architects and Designers, Independently published, 2023. P. 59
[21] B. Shamile. Software Development with UML Diagrams, Independently published. 2022. P.81.
[22] S. Sundaramoorthy. UML Diagramming: A Case Study Approach, Auerbach Publications. 2022. P.430.
[23] A. Rababah. Assessing the Effectiveness of UML Models in Software System Development. International Journal of Applied Science and Research. 2024. Pp.13-24. https:/doi.org/10.56293/IJASR.2024.5703.
[24] V. Ovsyak, ALGORITHMS: methods of construction, optimization, probability research. Lviv: Svit, 2001. P. 268. (In Ukrainian).
[25] K. Sierra. Head First Java, 3rd Edition: A Brain-Friendly Guide, O'Reilly Media; 3rd edition 2022. P. 752.
[26] C. Horstmann. ore Java: Advanced Features, Volume 2 (Oracle Press Java, Addison Wesley; 12th edition. 2022. P.1040.
[27] B. Evans. Optimizing Java: Practical techniques for improving JVM application performance, O′Reilly. 2018. P.437.
[28] M. Putten, S. Kennedy. Java Memory Management: A comprehensive guide to garbage collection and JVM tuning, Packt Publishing; 1st edition, 2022.P.146.
[29] B. Evans. Java in a Nutshell: A Desktop Quick Reference, O'Reilly Media; 8th edition. 2023. P.455.
[30] M.T. Somashekara, D.S. Guru, K.S. Manjunatha. Object Oriented Programming with Java, PHI Learning. 2017. P.754
[31] S. Kottwitz. LaTeX Beginner's Guide - Second Edition: Create visually appealing texts, articles, and books for business and science using LaTeX, Packt Publishing; Second edition. 2021.P.354.
[32] F. Mittelbach, U. Fischer. The LaTeX Companion: Parts I & II, 3rd Edition (Tools and Techniques for Computer Typesetting), Addison-Wesley Professional; 3rd edition. 2023. P.977.