Advances in Science and Technology Research Journal

Volume 14, Issue 3, September 2020, pages 49–56
https://doi.org/10.12913/22998624/122570

Received: 2020.05.15
Revised: 2020.06.30
Accepted: 2020.07.15
Available online: 2020.07.25

A Search Method for Reassembling the Elements of a Broken 2D Object

Jerzy Montusiewicz[1], Stanisław Skulimowski[1*]

[1] Department of Computer Science, Faculty of Electrical Engineering and Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

[*] Corresponding author's e-mail: s.skulimowski@pollub.pl

ABSTRACT

Searching for and reassembling the elements that used to form one whole is a very common issue faced by archaeologists. This is because preparing an interesting museum exhibition consists in the presentation of the objects that have been put together, not a pile of messily disassembled puzzle pieces. The article presents the concept of using the linguistic methods in the process of joining the elements of a 2D jigsaw puzzle. The method developed in the first stage creates the edge description of an object by defined unit vectors of the same length but different directions, and assigns them a designation in the form of letters, which leads to the creation of abstract words in the form of a sequence of signs. In the second stage, the words with a defined length of strings belonging to two different objects are compared. The authors have created a program that performs an exhaustive search until the pool of available elements is fully exhausted. The conducted numerical experiments indicate the correctness of the method and effectiveness in determining the places of joining elements. The developed method will be useful to automate the reassembly of 2D elements from archaeological excavations.

Keywords: exhaustive search, contour description, reassembling, linguistic methods, puzzle, apictorial

INTRODUCTION

The task of looking for the ways to match and assemble elements together, referred to as puzzle folding, is a problem known, among others, in archaeology, forensic museums, medicine and cartography. Regardless of the scenario (cracked bone, broken ceramics, cracked tombstones, torn maps or pictures), there are the same difficulties in trying to put the pieces together again. The process of reassembling is always accompanied by a degree of uncertainty:

- whether the available set of fragments is complete,
- whether the available set of fragments contains only fragments of one object (maybe there are "false" elements),
- whether it is possible to assemble the object precisely despite the existing defects in individual parts of the object (worn out edge or damaged, wiped out image on the surface).

The assembling of puzzles is very often done by manual testing and inspection. These tasks, carried out by field experts (e.g. museologists, doctors), consume a lot of time and energy, and their effectiveness is not very great. In the cases where the task of putting together the jigsaw puzzle is not critical (not a medical or forensic problem on which human health and life may depend), it is postponed and often remains unresolved for many years, and in extreme cases it is abandoned entirely. In such situation, any method of selection, classification or search for the possible combinations, and thus of generating alternatives, whether fully automatic or only partially, will be useful.

The purpose of the work is:

- to develop a method that will allow assembling a 2D puzzle using linguistic methods, and essentially to use one of them – the Levenshtein metric.
- confirmation of the correctness of the adopted method of describing the edges of the elements of objects by performing a numerical experiment with the use of a computer program built.

Background of the study

In the contemporary literature, the problem of combining the fragments belonging to one object is very often referred to as a "puzzle". This term is commonly associated with a form of entertainment consisting in assembling a 2D image of many elements with usually unique contours according to a printed pattern. The author of the first puzzles was the English cartographer John Spilsbury, who in the second half of the 18th century created such puzzles as a teaching aid to learn geography [1]. The issues dealt with by the scientists are related to assembling the jigsaw puzzle without having the information about the original object as whole.

The methods for finding the ways to combine the fragments of elements of destroyed objects can be divided into subgroups (Fig. 1). They have many variations, depending on: (i) the relevance and applicability of texture data, (ii) whether the purpose is to arrange the elements of an object or fragments of texture on the complete geometry [2] and (iii) the adopted projection of objects and the space in which they are checked (2D and 3D). The texture data can be used to rebuild the geometry or to complete the texture itself.

It is not possible to do an explicit division of the existing methods based on the area of use (archaeology, medicine, forensic science). Each field of knowledge may use any method depending on the specific task and available object data. Due to the complexity of the problem (in the case of 2D methods, the problem is NP-completeness [3]) and the ambiguity of the data available, a combination of different methods or a cascade of methods is also used to increase the chance of finding solutions, or at least to find classifiers in order to narrow down the group of potential solutions.

With regard to archaeological problems, specialist methods are being developed [4]. For the 3D elements of vessels representing a known culture and having the axis of symmetry, the vessel profile (described e.g. by the curvature signature) [5] is used. Methods have also been developed for 2D objects, using two features of the element: colour and edge [6], edge only [7], as well as methods operating on general 3D objects using the description of both the area of the alleged contact and the curvature signature [8].

The methods based on examining the edge characteristics most often use the description pertaining to the change of angles of short vectors placed one after another on the whole perimeter. This description is used to study angle variability using trigonometric functions [9]. Another way of describing the object's contour is to locate short vectors on the whole perimeter and assign them names resulting from the averaged absolute direction [10]. The compass rose (CR) figure, also called windrose or rose of the winds, is used for such a record.

In the scientific literature, one can find an application in describing change of the edge direction of linguistic objects [11], technical ones [12] or images [13] of chain codes. However, the solutions used there do not represent full information about the edge of objects, e.g. its total length. Eight-element Freeman's codes have different lengths for the orthogonal directions, and others for the diagonal ones. Thus, they are not invariant codes in their classical form. These codes are used to describe the edges at the level of a pixel image of objects.

The authors propose a new formula of string codes, which eliminate the disadvantages described above by creating the contour descriptions with information about both the shape and length of objects.

	pictorial		**apictorial**
	texture-based	*texture and shape based*	*shape-based*
2D	edge matching	jigsaw puzzles	packing puzzle polyform packing puzzle
3D	stretching and completing the texture on geometry	mesh matching + texture checking	mesh matching

Fig. 1. Schematic diagram of the classification of methods of searching for ways of arranging elements; in orange – a thematic area including the authors' method

METHODOLOGY OF ASSEMBLING A 2D PUZZLE

The authors propose a new method of matching a 2D puzzle, based on the information about the edges of the elements, omitting information about their texture. According to the presented classification of the assembling methods, the authors' idea belongs to the group of shape-based polyform puzzle (Fig. 1). This method can be used in archaeology, where large collections of fragments of objects with unique edges and uniform texture can be found (e.g. fragments of mosaics, ceramics, porcelain, bricks, etc.).

The developed method consists of three main stages (Fig. 2):

- Stage 1 – recording the edges of elements in the form of a character code,
- Stage 2 – sorting the existing elements according to the selected criterion,
- Stage 3 – recurrent comparison of the elements with the use of linguistic metrics.

Recording the edges of an element

The authors have developed their own recording system for edges, which can be classified as a recording family using the CR figure. The created notation was defined by four features:

- c1 – number of directions to which an object's circumference can be averaged (n),
- c2 – angle between successive directions (a=360/n),
- c3 – length of single vector (l),
- c4 – a written mark denoting the name of each direction.

Using three features: c1, c2, c3, it is possible to change the resolution of a discrete recording of an object's edges. The resolution is the higher the greater the c1 and the smaller the c3. The resolution is the smaller the c1 and the greater the c3.

After calibrating the c1, c2 and c3 features, an object edge outline is created (Fig. 3c) for a given object in a 2D projection (Fig. 3a), using the defined vectors (Fig. 3b). Next, a marker record is created on the basis of markings resulting from feature c4 (Fig. 3d).

In this way, a discrete record of the object's edge is obtained in the form of a character string. Thus, the process can be compared to writing Logo Drawing Commands [14], scout's Closed Course game [15], or Turn-by-turn navigation [16]. The contour description prepared in such a way is invariant to the rotation and selection of the starting point (Fig. 4).

Assuming that all elements of the set will be described in this way, comparing the edges of these elements can be done using not only mathematical methods but also linguistic metrics.

Levenshtein's metrics

Levenshtein's metrics (LM) belongs to the linguistic measures and is the edit distance that takes into consideration the insertion, deletion, substitution and transposition of two adjacent characters. It is used, among others, in medicine to check the similarity of DNA chains [17]. Two strings are identical when the measure LM=0. For each difference occurring in the compared strings a penalty of 1 is granted, e.g. the state of letter inconsistency in the same position (puck – pick), adding a letter (party – partly), removing a letter (shortly – shorty).

The method presented by the authors assumes the use of the Levenshtein measure to check the similarity of character strings corresponding to fragments (e.g. sides) of objects. This is done by reversing the sequence of one of the objects and comparing it with the unreversed sequence of other objects (Fig. 5).

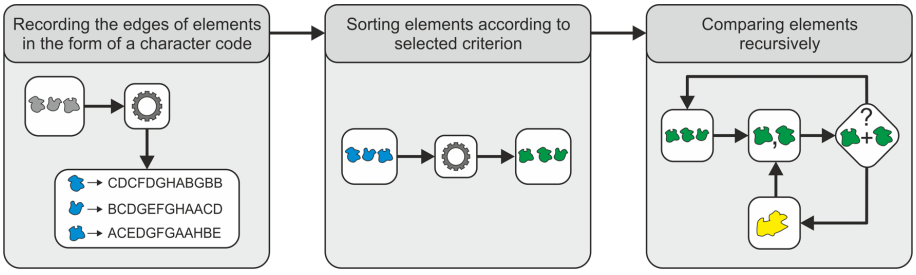

Fig. 2. Stages of the original method of matching 2D elements

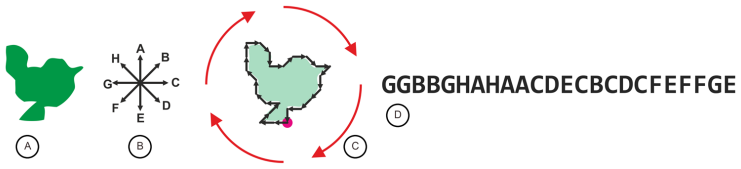

Fig. 3. An example of making a character notation: a) object, b) eight-point CR; c) object edge notation using CR vectors (red point – start position, red arrows – direction of contour drawing); d) character code after changing vectors into characters

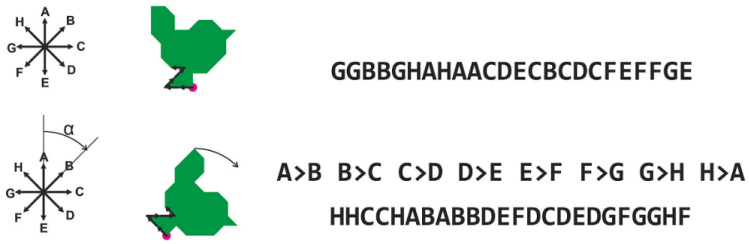

Fig. 4. Example of execution of an object rotation by a multiple of the angle defined in c2 – description properties will remain the same

Searching for possible matches

The procedure developed for creating matches is based on an exhaustive search and consists of always comparing two elements in pairs. The comparison of elements is based on a defined m-elemented character code, i.e. a fragment of a sequence which is always separated from the first element.

The matching of each two elements can take place in many ways when the elements have edges that can be joined in multiple combinations. In such a case, the procedure may include checking each of these ways. The number of possible matches is dependent on the length of the compared character strings and the minimum length of the compared parts of the character strings. The pessimistic number of comparisons of two elements can be represented by the formula:

$$\left|K_{p_1,p_2}\right| = (|p_1 - 2| - ms) * (|p_2 - 2| - ms) \quad (1)$$

where: K – a set of all possible ways to match the elements p_1 and p_2, p_2 – elements being compared,
$|p_i|$ – length of the character string of the i-th element,
ms – minimum length of the marker string of the fragment being compared,
2 – number of uncertain characters at the beginning and end of the string.

For each of the available initial elements (available in the set of fragments at the beginning of the procedure) it is checked if it matches the

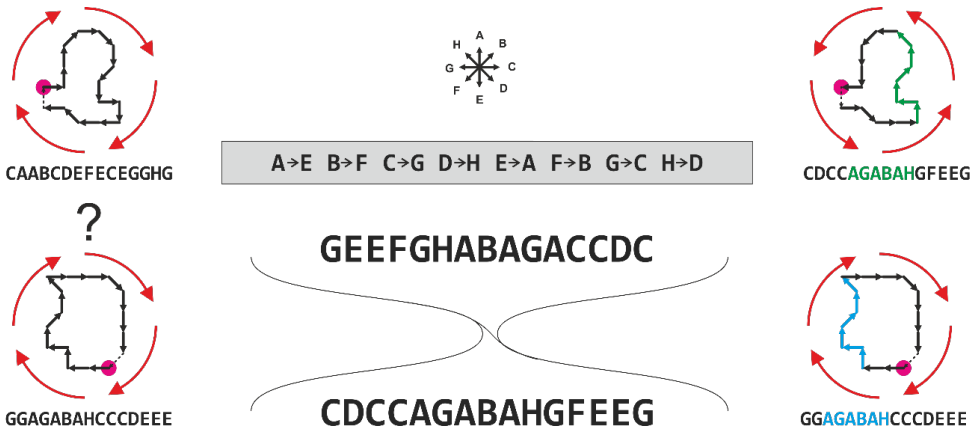

Fig. 5. Example of the reversal of an object's character string

next ones in the list of elements. If so, a new character code describing the new element is created on their basis. Then for this new element, the possibility of matching with other (existing elements in the set of initial fragments) subsequent ones in the list of available elements is checked. Each check always applies only to two elements (two original ones with each other or the assembled one with the original one). The number of possible results in terms of the elements used from the set of available elements (E), assuming that the available elements certainly come from a single puzzle, is equal:

$$|S| = \frac{(n-1)n}{2}\left(1 + \sum_{i=1}^{n-2}\left(\prod_{j=i}^{n-2} j\right)\right), \qquad (2)$$

$$S = \{s_1, s_2, \ldots, s_m\}, 2 \leq |s| \leq n, |E| = n$$

where: E – a set of basic elements,
n – number of basic elements,
S – an orderly set of possible assemblies, taking into account the fact that the basic elements belong to the assembly, without taking into account the way of matching the elements to each other, and each assembly can consist of any number of elements from 2 to n.

In order to reduce the number of comparisons and potential solutions, a number of limitations have been introduced into the procedure, allowing for precise clarification of the way of conducting the search and an automatic decision on the significance of the comparisons performed.

Programme description

The prepared program LiMePuRe2D (Linguistic Methods of Reassembling 2D Puzzle) was written in C#, in the Visual Studio environment. The program uses the prepared list of strings that mean a containing the ID and discrete record of the edges of the objects to be examined. It also includes generating the instructions of assembly solutions in the form of step-by-step instructions in the graphic and text forms. In this way it was possible to check the list of potential solutions found. In the pilot version of the program to compare pairs, elements are selected by drawing from a pool of string lists. The possible match is remembered and the process of searching for the next matching element is continued.

RESULTS AND DISCUSSION

In order to check the effectiveness of the method and the operation of the program, an example was prepared that included a set of several elements that fit together to form a compact puzzle. Thus, the set was prepared in a pre-assembled form. The characteristic feature of this example is that the fragments of the sign strings separated from individual elements are repeated in at least one other fragment. This causes the program to generate alternative assemblies of individual elements. In this way the authors wanted to check whether the program: would be able to recreate the shape originally assumed by the authors and how it would deal with alternative combinations.

Character representation of object edges and number of characters p_i: 1. BBDBACABACDCEDCDCEDECECEDEGFGGFGEGGGHGHGHGHAAGA / p_1=47;

2. CDDEEFECEDECECEDEFF E G H F F G G G H H A G A H A G A G A HABBCBBCAACB / p_2=51;

3. FFGEEGFFGFFGHGHAGHABABABACCACB CCBDDEDDEC / p_3=41;

4. CEDDFFEEDFGGGFGHAGAGAHACCACBCCB / p_4=32;

12. GFFEGHFGHAGFGAHAGAGAHAGHGHABBAHAABCBCBCBDDBDCDDCCDDC DFEDEEEDEEFFE / p_{12}=65.

Figure 6 presents a sign description of the elements selected for the experiment. A graphic representation of the elements tested is made up of unclosed contours. It results from the limitations of the compass rose resolution (c1, c2, c3). The total length of vectors does not always coincide with the actual circumference length of the element. In order to limit the possible errors when comparing elements, the program does not take into account two extreme vectors (first and last).

The experiment was conducted in several variants: with a change in the number of the possible combinations of each of the two elements (PCC), and with a change in the threshold value for the Levenshtein method (value 0 – full match and 1 – match with one error). For all the attempts, it was assumed that the minimum length of the strings to be compared was ms=5. In the process of comparing the two elements, the "slide" method was used for both elements. For each of the first element's substrings, a comparison was made with all the substrings of the second element. The

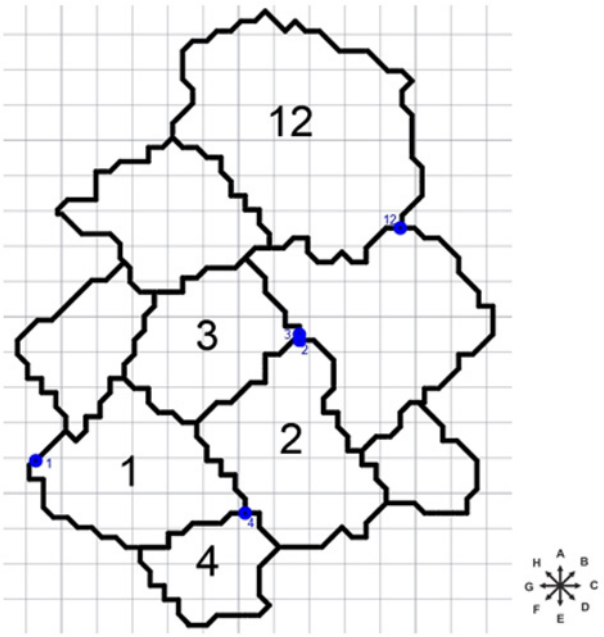

Fig. 6. The assumed way of matching the prepared elements with the compass rose figure used, blue point – starting point

joists in this comparison were constructed from every five consecutive characters in the element (0 to 4, 1 to 5, 2 to 6, etc.) except the first and last character (Ex. 1). The substrings were checked for the best match of the sign string in terms of the Levenshtein method.

Each of the variants was repeated many times for a precise measurement of the actual time of making comparisons and connections (excluding the markup of the programming environment). The test program was run on a machine with an i7–4770, 3.90 GHz processor and 16GB RAM.

In the experiment, for pair comparison, the elements were taken at random. It should be noted that the puzzles for testing the program were prepared in such a way that there were many alternative ways of combining the components that did not comply with the patterns (Fig. 8).

In the prepared test pattern (Fig. 6), element 12 is combined with an element that was not available in the numerical experiment. The analysed sequence of five characters allows connecting two different elements to the same side: element 1 and element 2. In the case shown in Figure 8d it can be seen that between elements 2 and 12 there is a collision in the further part of the connected sides. This means that the selection of the length of the string to compare both elements is an important value and determines the number of possible comparisons (Ex. 2) as well as the correctness of the searched connections. The case presented in Fig. 8a shows the situation that the pattern connection between elements 4 and 2 is realised on a 4 character long string. Of course, when the elements 1 and 2 have already been connected, analysing the substring of length 5 will bring the right solution.

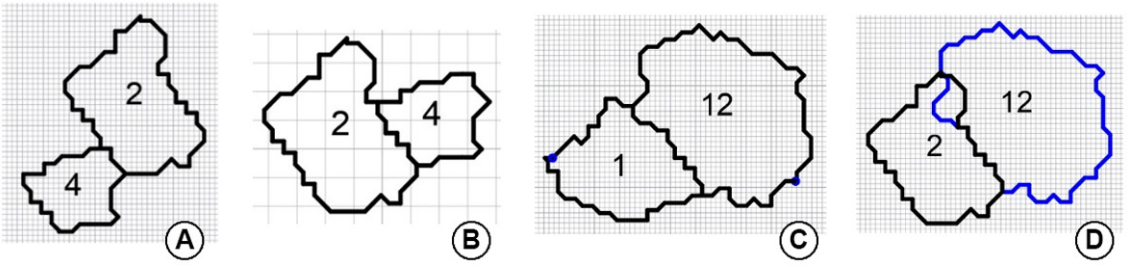

Fig. 7. Examples of connection of elements: (a) connection from pattern, (b) correct alternative connection, (c) correct alternative connection, (d) incorrect alternative connection

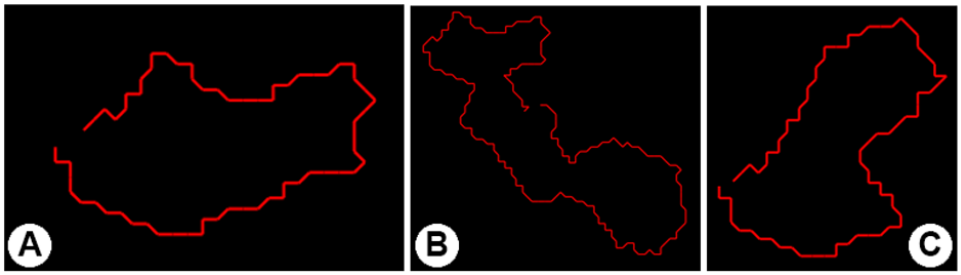

Fig. 8. Examples of the obtained assemblies, elements with numbers: a) 1 and 4, b) 1,4,3,2,12 – looking at the object from above, c) 1 and 3

The results are presented in Table 1 with: number of connection options tested – NCOT, number of connections made – NCM, number of final results – NR and average program running time.

The NCOT determines the total number of comparisons made for each of the two 5-character-long substrings, regardless of the Levenshstein metric result. This value is the resultant of checking all possible ways of connecting elements (Ex. 1) and the number of ways of selecting the elements for checking connections (Ex. 2). The NCOT depends strictly on PCC, LM and NCM. NCM is a number that indicates how many matches there are for each bonded part of all the other parts that do not form a part of the bond. This number is difficult to predict, but from the experiment it can be concluded that even a simple example is able to generate a large number of tests, so it is necessary to introduce the optimisation mechanisms and screening algorithms that will reject certain paths before the calculation is continued. At present, in the prototype version of the program, no mechanism has been implemented yet. NCM determines the number of the successfully completed assemblies, i.e. the ones that have been successfully implemented: are unique, meet the accepted criteria of the Levenshtein metric and their execution does not cause the overlapping of elements (Fig. 8d). Determination of the ratio of the NCM to the NCOT numbers (which is from 0.02% to 0.08%) shows that there are great possibilities and at the same time the necessity to develop a mechanism of prediction assembly places for elements.

On the basis of the results obtained, it can be concluded that the time of operation execution increases exponentially along with the number of tested combinations and with the increase in the margin of error in determining the matching of strings using the Levenshtein metric. A minimal increase in LM results in a significant increase in the number of possible assemblies. Each assembly means that a new search for subsequent matches has to be started, which significantly extends the whole process. Assuming an even probability distribution of the possible assemblies for each substring of the tested element, it is necessary to check all the possible connections and wait until the program completes all actions. With a large number and high complexity of objects to be checked, without optimisation mechanisms, the process may prove too long to be carried out in a reasonable time.

The prototype version of the program was able to correctly determine the assembly of four adjacent elements according to the prepared pattern, and that with the first, the simplest of the adopted search variants – LM=0 and PCC=1. The program also suggested the possibility of matching the fifth element (No. 12) according to the adopted assumptions of the method. The example connections are presented in Figure 8. While analysing the results, it was found that in each of the variants nearly half of the assembly possibilities found were very similar to each other (e.g. they differed in the shift of the last connected element). The exception is the most extensive variant (LM=1, PCC=5) where the number of similar elements was about 80%.

Table 1. Results of the experiment to determine the possible assemblies for the assumed PCC and LM parameters

PCC	LM=0				LM=1			
	NCOT	NCM	NR	Time [s]	NCOT	NCM	NR	Time [s]
1	318919	60	24	3.735	701931	159	60	8.377
2	511441	95	49	6.227	4045269	1564	949	68.418
5	511441	95	49	6.206	34510887	26904	21404	619.444

The number of assembly results (NR) to which it is no longer possible to assembly other elements, or there is a lack of elements that can be assemble, increases along with the PCC and LM values. However, this does not mean that the solutions found are correct in terms of the best fit or packaging of the elements. The connection shown in Figure 8a shows the correct alternative gluing of elements 1 and 4 – different from the pattern in Figure 6 (elements 1 and 4 are connected with other sides). The assembly of elements 1 and 3 shown in Figure 8c was made in accordance with the pattern in Figure 6. The alternative assembly of the same elements shown in Figure 8b did not match the pattern (Fig. 6.). In a situation when one does not have the remaining elements from the pattern, it is impossible to say whether this assembly is correct or just a solution without errors. Testing the program on the supplied elements shows that few of the generated solutions were constructed on the basis of 4 or 5 elements. The most common solutions were those consisting of 2 elements, which from the perspective of searching for the best matches is unexpected. Such solutions had no further continuation and were eliminated in further search.

CONCLUSIONS

The results obtained in the numerical experiment allow drawing the following conclusions:

- the description of the element contours using an eight-element unit rose of vectors contains enough information to perform the correct assembly of elements,
- the developed method based on the Levenshtein metric, belonging to the linguistic measures, allows finding the possible matching solutions for a 2D puzzle,
- the constructed computer program correctly executes the described method and procedure, and the correct definition of the length of the character string to compare the matching of elements allows for the calculations to be performed in a dozen or so seconds,
- in the tested version of the program, the final verification of the assembly of individual elements was graphically verified.

REFERENCES

1. Lau C., Schwartzburg Y., Shaji A., Sadeghipoor Z., Süsstrunk S., Creating Personalized Jigsaw Puzzles, In Proceedings of the Workshop on Non-Photorealistic Animation and Rendering (NPAR '14). Association for Computing Machinery, New York, NY, USA, 2014, 31–39.
2. Guo K., Chen X., Liu Y., Zhou B., Guo Y., Geometric Based Structure Propagation and Texture Matching for 3D Texture Completion, International Conference on Virtual Reality and Visualization, Xi'an, 2013, 87–93.
3. Demaine E.D., Demaine M.L., Jigsaw Puzzles, Edge Matching, and Polyomino Packing: Connections and Complexity, Graphs and Combinatorics 23, 2007, 195–208.
4. Rasheed N.A., Nordin M.J., A Survey of Computer Methods in Reconstruction of 3D Archaeological Pottery Objects, International Journal of Advanced Research, 2015, 712–724.
5. Willis A. R., Computational Analysis of Archaeological Ceramic Vessels and their Fragments, Digital Imaging for Cultural Heritage Preservation, 2011, 323–352.
6. Zhou M., Geng G., Wu Z., Zheng X., Shui W., Lu K., and Gao Y., A System for Re-assembly of fragment Objects and Computer Aided Restoration of Cultural Relics, Virtual Retrospect 2007, 21–27.
7. Kong W., and Kimia B., On solving 2D and 3D puzzles using curve matching, Computer Vision and Pattern Recognition, Proceedings of the 2001 IEEE Computer Society Conference on, 2001, 583–590.
8. Andreadis A., Papaioannou G. and Mavridis P., Generalized Digital Reassembly using Geometric Registration, 2015 Digital Heritage, 2015, 549–556.
9. Freeman H., Davis L., A Corner-Finding Algorithm for Chain-Coded Curves, IEEE Transactions on Computers, 1977, 297–303.
10. H. Freeman, L. Garder, Apictorial Jigsaw Puzzles: The Computer Solution of a Problem in Pattern Recognition, IEEE Transactions on Electronic Computers, 1964, 118–127.
11. Aini N.A., Dewi N., Azurah A.S. Freeman chain code as representation in offline signature verification system, Jurnal Teknologi, 2016, 89–94.
12. Grabowik C., Kalinowski K., Ćwikła G., Gwiazda A., Monica Z., Zastosowanie kodów łańcuchowych do opisu konstrukcji oraz identyfikacji konstrukcyjnych obiektów elementarnych, Innowacje w zarządzaniu i inżynierii produkcji, 2017, 180–190.
13. Karczmarek P., Kiersztyn A., Pedrycz W., Dolecki M., An application of chain code-based local descriptor and its extension to face recognition, Pattern Recognition, 2017, 26–34.
14. Ernest P., What's the Use of LOGO?, Mathematics in School, JSTOR, 1988, 16–20.
15. Eureka!, 3 compass games that teach kids to use a compass, https://www.eurekacamping.com/blog/article/3-compass-games-teach-kids-use-compass, access: 07.04.2020
16. Kenneth J.B., Hensher D.A., Handbook of transport systems and traffic control. Emerald Group Publishing, 2001, 497.
17. Buschmann, T., Bystrykh, L.V., Levenshtein error-correcting barcodes for multiplexed DNA sequencing, BMC Bioinformatics 14, 2013, 272.