computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

# TRIANGULATING A REGION BETWEEN ARBITRARY POLYGONS

**Vasyl Tereshchenko [1], Yaroslav Tereshchenko [2]**

Taras Shevchenko National University of Kyiv
4d Academician Glushkov avenue, Kyiv, Ukraine, 03680
[1] vtereshch@gmail.com, http://tvm.unicyb.kiev.ua/
[2] y_ter@ukr.net, http://mi.unicyb.kiev.ua/

**Abstract:** The paper presents an optimal algorithm for triangulating a region between arbitrary polygons on the plane with time complexity $\boldsymbol{O(N \log N)}$. An efficient algorithm is received by reducing the problem to the triangulation of simple polygons with holes. A simple polygon with holes is triangulated using the method of monotone chains and keeping overall design of the algorithm simple. The problem is solved in two stages. In the first stage a convex hull for *m* polygons is constructed by Graham's method. As a result, a simple polygon with holes is received. Thus, the problem of triangulating a region between arbitrary polygons is reduced to the triangulation of a simple polygon with holes. In the next stage the simple polygon with holes is triangulated using an approach based on procedure of splitting polygon onto monotone polygons using the method of chains [15]. An efficient triangulating algorithm is received. The proposed algorithm is characterized by a very simple implementation, and the elements (triangles) of the resulting triangulation can be presented in the form of simple and fast data structure: a tree of triangles [17].


**Keywords:** Triangulation, simple polygon, holes, region, reducible problem, monotone polygon.

## 1. INTRODUCTION

The problem of optimal triangulation of a region between arbitrary polygons on the plane is considered. There are a few efficient algorithms to the solution of this problem [1, 2]. However, searching for an optimal way of solution is still an actual task today. On the other hand, triangulation of simple polygons is widely used in many applications, in particular, engineering applications, and can refer to different disciplines such as microbiology, geodesy and others [3, 4].

By analyzing existing approaches to solving the discussed problem, we can note the following. Goodman showed the possibility of triangulating a region between $k$ polyhedra with $N$ vertices in $O(N + k^2)$ time [5]. As a result of further research, Chazelle pointed out that complexity of the solution depends on a shape of area between polyhedra [1]. Later, Joe has proposed a new approach, which is associated with a local polyhedra transformation that improves Delaunay triangulation [6]. However, the performance of this algorithm is $O(N^2 + k^2)$, that is far from desirable. In the work [2] authors proposed the method of the modified Delaunay triangulation with limitations for d-dimensional space in $O(N^2)$ time. Tarjan and Van Wyk in [7] had proposed an algorithm for triangulating a simple polygon with time complexity $O(N \log\log N)$ and later it was simplified by Kirkpatrick [8]. Clarkson [9], Devilers [10] and Seydel [11] proposed a randomized algorithm, which has execution time $O(N\log^* N)$.

All algorithms mentioned above give good results in the case of convex polygons (polyhedrons), however, it is desirable to have an optimal solution to the general case. Naturally, the next question arises: whether it is possible to develop an algorithm that would give high efficiency and have a simple enough implementation. It is especially important in terms of practical applications of the algorithm. For instance, it can be a triangulation of a simple polygon with holes of arbitrary shape using a model of a unified algorithmic environment, which can significantly increase efficiency of interaction between parts of the entire set of algorithms (sub-algorithms, procedures, functions), and therefore reduce the total algorithm execution time [12]. In this paper, we propose a new approach to solving the problem of triangulating a region between simple polygons using reduction of the problem of triangulating a simple polygon with holes and using the method of monotone chains.

## 2. PROBLEM AND METHOD OF SOLUTION

**Problem.** The given $m$ arbitrary $k$-vertex polygons on the plane are shown in Fig. 1. It is necessary to triangulate the region that is bounded by these polygons and their convex hull, Fig. 2.

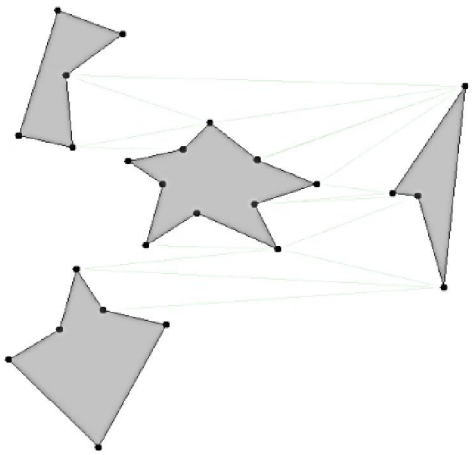


**Fig. 1 – The given set of polygons**

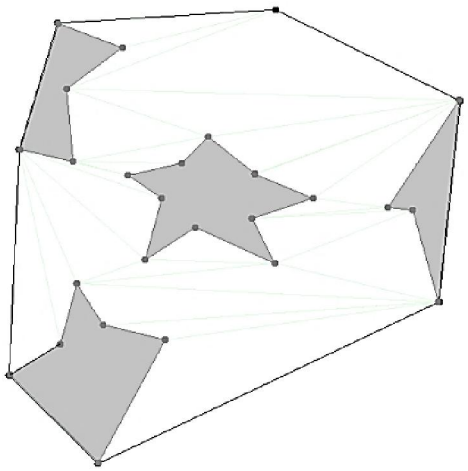


**Fig. 2 – The area for triangulation**

To solve the problem we apply the reduction method [3], reducing our problem to the problem of triangulating a simple polygons with holes. We recall some concepts [1].

**Definition 1.** A polygon is simple if there is no pair of nonconsecutive edges sharing a point or in other words: connected boundary without self-intersections.

**Definition 2.** A simple polygon $P$ is called monotonous relatively to a diagonal $l$ (if it exists), if $l$ divides polygon onto two monotone chains.

**Theorem 1.** Let there be given $m$ polygons, and $n_1, n_2, \ldots, n_m$ – number of vertices for each polygon respectively. The problem of triangulating a region between $m$ polygons (not necessarily convex) is reduced to the problem of triangulating a simple polygon with $m$-holes (where $N= n_1+ n_2+ ...+ n_m$) in linear time.

*Proof.* Let there be given $m$ polygons, and $n_1, n_2, \ldots, n_m$ – be a number of vertices of each polygon respectively, then $N= n_1+ n_2 +... + n_m$ – the total number of vertices for given polygons. We construct the convex hull for this set of polygons. As a result, we will get a simple polygon with $m$ holes. Therefore, triangulation of a region between $m$ arbitrary polygons in this case is reduced to triangulation of a simple polygon with holes. According to the theorem about existence of a simple polygon triangulation [13], such a triangulation always exists.

### 2.1 TRIANGULATION OF SIMPLE POLYGONS

Before turning to triangulating a simple polygon with holes we consider triangulation of a usual simple polygon without holes. Today, there are several methods for triangulating such simple polygons. The most famous method is offered in [13, 16] with complexity $O(N^2)$. In addition, one of the most promising approaches today is the method of selection of monotone polygons, which is considered in papers [1, 3, 9, 11, 14]. The simplest of these methods, which has complexity $O(N\log N)$ [1] offers some additional restrictions: the triangulated region is a rectangle that contains all vertices. In [11] Seidel proposes a randomized algorithm, in which the monotonicity and splitting on trapezes are used, resulting in average time complexity $O(N\log N)$. In [14] Chazelle suggests a triangulation algorithm for a monotone polygon in linear time, which is the best result to date. All of the algorithms are similar to a certain extent and differ only in complexity of implementation. Below is a comparison chart for existing methods of triangulation.

**Table 1. Comparison of triangulation methods**

| Method | Time complexity |
|---|---|
| Brute force method[14] | $O(N$^4) |
| Ear clipping method[17] | $O(N$^2) |
| Seidel's method [11] | $O(N \log N)$ |
| Tarjan's method [7] | $O(N \log \log N)$ |
| Chazelle's method [14] | $O(N)$ |

### 2.2 TRIANGULATION OF SIMPLE POLYGONS WITH HOLES

**Problem.** Can we triangulate a simple $N$ – vertex polygon that has $k$ holes inside without introducing additional points in $O(N\log N)$ optimal time?

To solve the problem we use a chain method [15]. Let us consider the basic algorithm.

**Algorithm**

1. **Preliminary preparation.** We represent a polygon with holes as a graph and orient it in order of ascending ordinate values for vertices (from smaller to larger *y*), Fig. 3. In the same way, we sort vertices, Fig. 4.

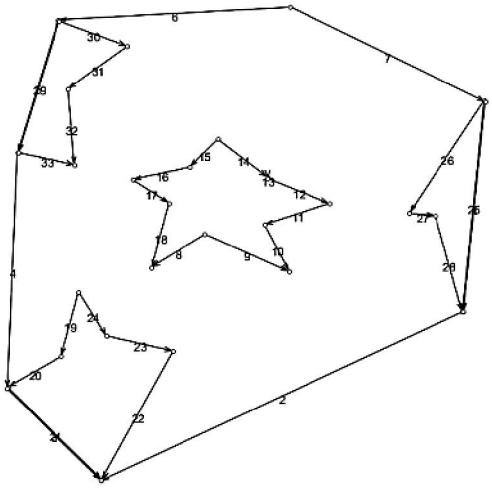


**Fig. 3 – Representation of a polygon with holes as an oriented graph; b) Graph regularization**

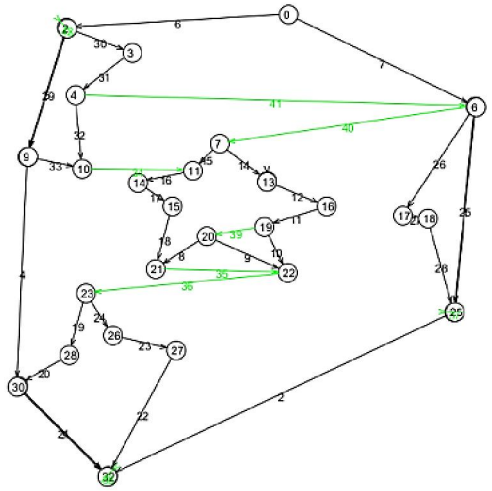


**Fig. 4 – Graph regularization**

2. **Regularization.** By sweeping plane in two passes we connect vertices that do not have input or output edges from the previous (following) vertices. So we get a graph, in which each vertex (except the first and last vertices) has an ordered set of input edges IN (*v*) (clockwise) and output edges OUT (*v*) (counterclockwise), Fig. 4.

3. **Pushing flow.** We push flow in two stages. In the first stage, we push total unbalanced flow of the upper edges through the lower left edge. In the second stage we push the flow from the bottom to up through the upper left edge. Thus, we get at each vertex *v* (except the first and last vertices) balanced by weights graph *G*: WIN (*v*) = WOUT (*v*) (Fig. 5). This allows to build a complete set of *m* monotone chains Z = Z($C_1$, $C_2$, ..., $C_m$) relatively *OY*.

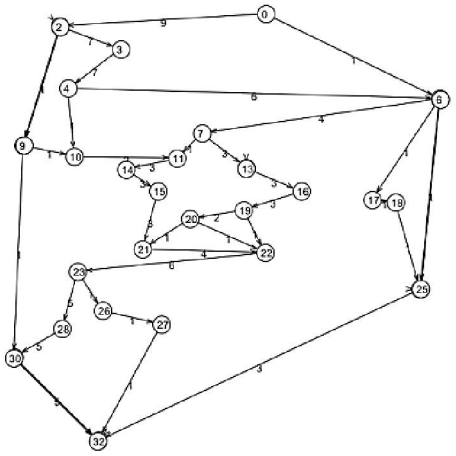


**Fig. 5 – Pushing flow stage**

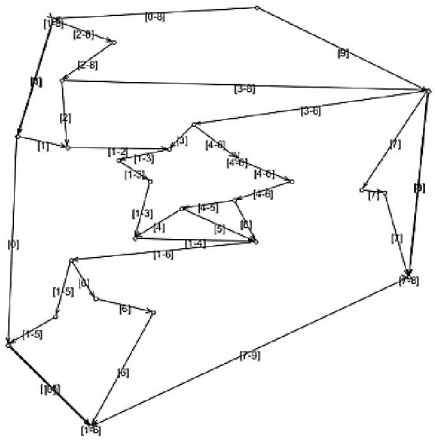


**Fig. 6 – Bind chain ranges**

4. **Determination of intervals for chains.** For each edge and vertex we define a number of chains to which they belong while passing from top to bottom (Fig. 6.).

5. **Splitting in monotone polygons.** Using sweeping plane method, based on information obtained at the previous stage, we divide our graph onto monotone polygons. We discard polygons that belong to holes (Fig. 7).

6. **Triangulation of monotone polygons.** We triangulate polygons by method described in [1], (Fig. 8).

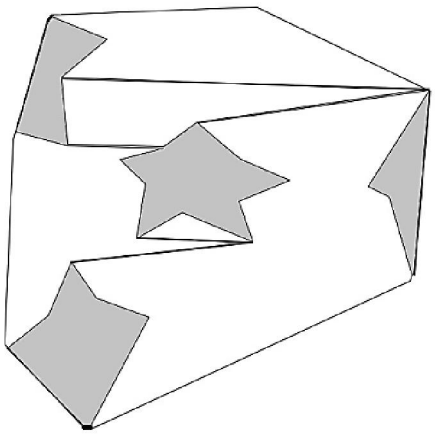


**Fig. 7 – Splitting in monotone polygons. Triangulation of monotone polygons**

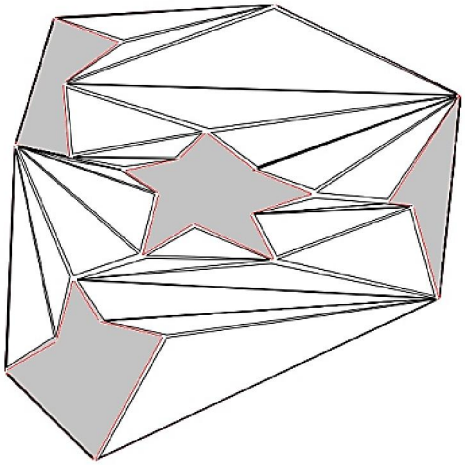


**Fig. 8 – Triangulation of monotone polygons**

## 3. SUBSTANTIATION OF COMPLEXITY ESTIMATION

**Theorem 2.** Triangulation of simple N- vertex polygon with holes on a plane can be performed in $O(N\log N)$ time for the worst case using $O(N)$ memory.

*Proof.* Let us consider time complexity for each step of the proposed algorithm:

*Preliminary preparation.* Analysis of time complexity: sorting gives $O(N\log N)$ and one pass for creating and orientating the mentioned graph gives $O(N)$ in the worst case.

*Regularization.* In regularization stage we pass through all vertices in $O(N\log N)$ time in the worst case.

*Pushing flow.* The pushing flow in two directions (from the bottom to up and from the top to down) gives $O(N)$ in the worst case.

*Computing intervals for chains.* In this stage all edges and vertices are selected only once: $O(N)$ time.

*Splitting on monotone polygons.* One pass through all vertices and incidental edges takes $O(N)$ time in the worst case.

*Triangulation of monotone polygons.* According to [1, 13] triangulating of each of $M$ monotone polygons can be performed in $O(K)$ time, where $K$ is the number of vertices for the monotone polygon. In general, passing through all monotone polygons gives $O(N)$ in the worst case.

**Theorem 3.** The problem of triangulating a region between m polygons with total of $N$ vertices can be solved in $O(N \log N)$ time.

*Proof.* Summing up complexity of all stages described above and taking into account the result of the Theorem 2, we get the total time $O(g(N))$ for triangulation:

$$\begin{aligned} \boldsymbol{O(g(N)) = O(N\log N)} \\ \boldsymbol{+ O(N\log N) + O(N)} \\ \boldsymbol{= O(N\log N)} \end{aligned}$$

## 4. IMPLEMENTATION

Program implementation is executed on the Java platform and includes two components: interactive input data tasks and a conversion triangulation module. Conversion is performed in parallel with data entry and immediately displayed after changing data. The program contains controls that allow to upload and store configuration polygons on the screen.

The main modules of the program: **Main, PointsPanel, Point, ConvexHull, TRIANGTree, TRIANGNode, ComparableComparator, PolarAngleComparator, Generate random.**

**Main** starts the main window. It contains polygon generation and processing algorithms results.

**PointsPanel** – drawing points, convex hull and triangulation.

**Point** – stores the point coordinates and color.

**ConvexHull** – builds convex hull and supports the corresponding data structure.

**TRIANGTree** – implements and supports data structure: a tree of triangles.

**TRIANG**Node – stores information about the class point **TRIANGTree**.

**ComparableComparator –** standard comparator used in **TRIANGTree**.

**PolarAngleComparator** – comparator that sorts points by polar angle **TRIANGTree**.

**Generate random** generates *m* of *k*- vertices.

One feature of the proposed algorithm is a very simple implementation, and the elements (triangles) of the resulting triangulation can be presented in the form of simple and fast data structure: a tree of triangles [17]. This makes the algorithm convenient for solving a wide range of applied and research problems, in particular, those referring to computational geometry and computer graphics, meshing and rendering [18], pattern recognition and image segmentation [19, 20], approximation and 2D (3D) simulation. In particular, the example of application of the algorithm is geo-information systems (GIS) for the surface reconstruction from input data captured by satellites.

We have tested the algorithm for different input data. In particular, Fig. 9 shows an example of triangulating a region between 100 polygons, and Fig. 10, Fig. 11 (a, b) demonstrate time complexity for different number of unique polygons and points.

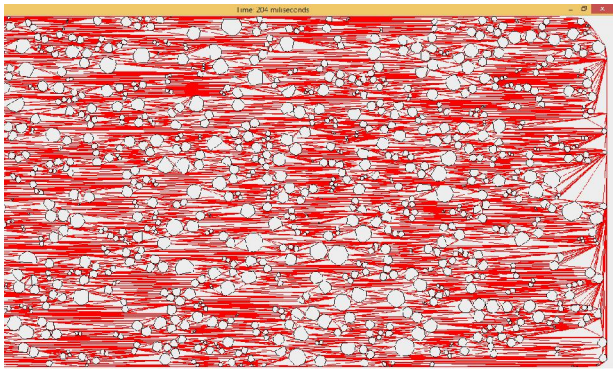


**Fig. 9 – An example of triangulation algorithm is a region between 100 polygons**

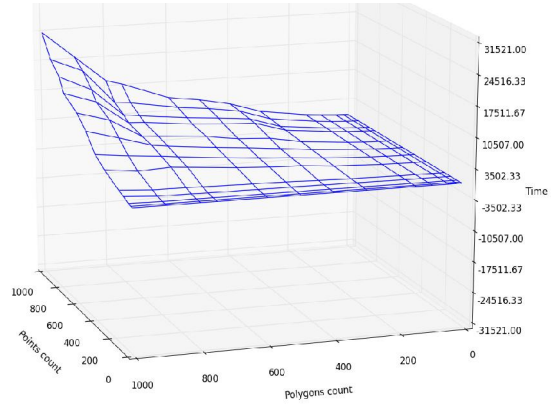
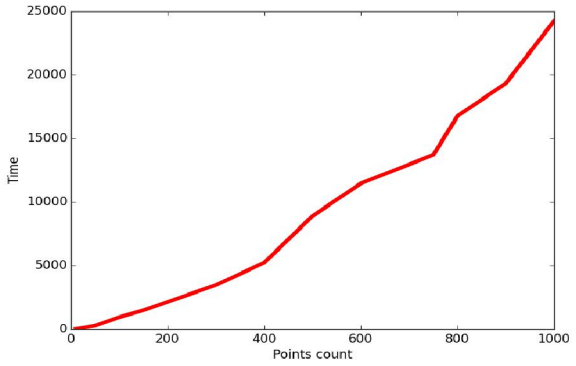
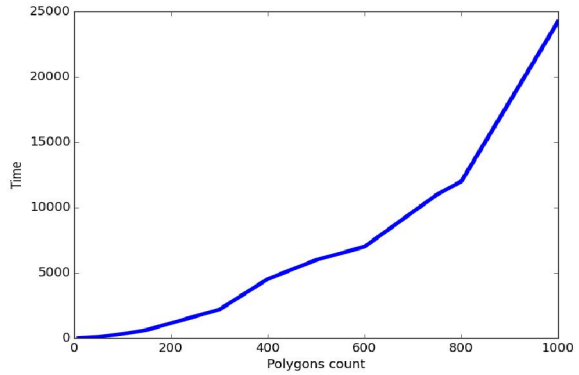



**Fig. 10 – Time comlexity of the algorithm both for points and polygons**




**a)**




**b)**

**Fig. 11 – Time comlexity for a) points with 1000 polygons; b) polygons with 1000 points**

## 5. CONCLUSION

In this paper, we propose the method of triangulating a region between $m$ polygons (star or simple polygons) having $N$ vertices in total in $O(N\log N)$ time. The main idea of the described method is in the fact that it consists of two stages: reducing the problem to triangulating a simple polygon with holes and triangulating of the polygon by splitting it onto monotone polygons using the method of chains [15]. The suggested method allows successful balancing between performance and complexity of implementation. Another advantage of the proposed algorithm is that it can solve an extended problem: the case of "nested holes".

## 6. REFERENCES

[1] B. Chazelle, N. Shouraboura, "Bounds on the size of tetrahedralizations," *J. Discrete & Computational Geometry*, vol. 14, no. 1, pp. 429-444, 1995.

[2] V. Tereshchenko, S. Pilipenko, A. Fisunenko, "Domain triangulation between convex polytopes," *J. Procedia Computer Science*, vol. 18, pp. 2500–2503, 2013.

[3] F. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, Berlin, 1985, 398 p.

[4] V. S. Bileckiy, *Small Mine Encyclopedia*, vol. 1-3, Shidniy Vidavnichiy Dim, Donetsk, 2013, 1936 p. (in Ukrainian)

[5] J. E. Goodman, J. O'Rourke, *Handbook of Discrete and Computational Geometry*, 2nd ed., CRC, A CRC Press Company, Boca Raton, London, 2004, 1453 p.

[6] B. Joe, "Construction of three-dimensional Delaunay triangulations using local transformations," *J. Computer Aided Geometric Design*, no. 8, pp. 123-142, 1991.

[7] R. E. Tarjan, C. J. Van Wyk, "An O(nlog logn)-time algorithm for triangulating a simple polygon," *SIAM J. Comput*, vol. 17, pp. 143–178, 1988.

[8] D. G. Kirkpatrick, M. M. Klawe, R. E. Tarjan, "Polygon triangulation in O(nlog logn) time with simple data structures," *J. Discrete Comput. Geom.*, vol. 7, pp. 329–346, 1992.

[9] K. L. Clarkson, R. E. Tarjan, C. J. Van Wyk, "A fast Las Vegas algorithm for triangulating a simple polygon," *J. Discrete Comput. Geom.*, vol. 4, pp. 423–432, 1989.

[10] O. Devillers, "Randomization yields simple O(nlogn) algorithms for difficult Ω(n) problems," *Internat. J. Comput. Geom. Appl.*, vol. 2, pp. 97–111, 1992.

[11] R. Seidel, "A simple and fast incremental randomized algorithm for computing

trapezoidal decompositions and for triangulating polygons," *J. Comput. Geom. Theory Appl.*, vol. 1, pp. 51–64, 1991.

[12] V. Tereshchenko, I. Budjak, A. Fisunenko, "The unified algorithmic platform for solving complex problems of computational geometry," *J. Parallel Computing Technologies*, vol. 7979, pp. 424-428, 2013.

[13] M. de Berg, M. van Kreveld, M. Overmars, O. Cheong, *Computational Geometry*, 3rd ed., Springer-Verlag, Berlin, 2008, 398 p.

[14] B. Chazelle, "Triangulating a simple polygon in linear time," *J. Discrete Comput. Geom.*, vol. 6, pp. 485-524, 1991.

[15] E. Edelsbrunner, L. J. Guibas, J. Stolfi, "Optimal point location a monotone subdivision," *SIAM J. Comput*., vol. 15, no. 2, pp. 317–340, 1986.

[16] G. H. Meisters, "Polygons have ears," *J. American Mathematical Monthly*, vol. 82, pp. 648–651, 1975.

[17] V. Tereshchenko, Y. Tereshchenko, D. Kotsur, "Point triangulation using Graham's scan," in *Proceedings of the 5-th IEEE International Conference on Innovative Computing*, Galicia, Spain, May 20-22, 2015, pp. 148-151.

[18] U. Grossmann, M. Schauch, S. Hakobyan, "The accuracy of algorithms for WLAN indoor positioning and the standardization of signal reception for diferent mobile devices", *International Journal of Computing*, vol. 6, issue 1, pp. 103-109, 2007.

[19] D. Zahorodnia, Y. Pigovsky, P. Bykovyy, "Canny-based method of image contour segmentation," *International Journal of Computing*, vol. 15, issue 3, pp. 200-205, 2016.

[20] N. I. Korsunov, D. A. Toropchin, "The method of finding the spam images based on the hash of the key points of the image," *International Journal of Computing*, vol. 16, issue 4, pp. 259-264, 2016.

***Prof. Vasyl Tereshchenko,*** *Graduated in 1986 Mathematics and Mechanics Faculty at Taras Shevchenko National University of Kyiv, speciality – applied mathematics and mechanics. Now he works as Head of the Department of Mathematical Informatics.*

*Areas of scientific interests: simulation and visualization, computational geometry, computer graphics, computer vision, pattern recognition, theory of algorithms, parallel algorithms and programming, information systems, database, nonlinear integral and differential equations, thermo mechanics inhomogeneous solids.*

***Yaroslav Tereshchenko*** *is a student at the student of second year Master's Program, at the Department of Mathematical Informatics, Faculty of Computer Science and Cybernetics (computer sciences specialty), Taras Shevchenko National University of Kyiv.*

*Areas of scientific interests: simulation and visualization, computational geometry, computer graphics, computer vision, pattern recognition, theory of algorithms.*