THE PARALLEL IMPLEMENTATION OF ALGORITHMS FOR FINDING THE REFLECTION SYMMETRY OF THE BINARY IMAGES

S. Fedotova [a *], O. Seredin [a], O. Kushnir [a]

[a]Tula State University, 300012, Tula, Lenin Ave., 92, Russia - fedotova.sonya@gmail.com

Commission II, WG II/10

KEY WORDS: Reflection Symmetry of Images; Binary Images, High-Performance Computing, Parallel Performance Evaluation

ABSTRACT:

In this paper, we investigate the exact method of searching an axis of binary image symmetry, based on brute-force search among all potential symmetry axes. As a measure of symmetry, we use the set-theoretic Jaccard similarity applied to two subsets of pixels of the image which is divided by some axis. Brute-force search algorithm definitely finds the axis of approximate symmetry which could be considered as ground-truth, but it requires quite a lot of time to process each image. As a first step of our contribution we develop the parallel version of the brute-force algorithm. It allows us to process large image databases and obtain the desired axis of approximate symmetry for each shape in database. Experimental studies implemented on "Butterflies" and "Flavia" datasets have shown that the proposed algorithm takes several minutes per image to find a symmetry axis. However, in case of real-world applications we need computational efficiency which allows solving the task of symmetry axis search in real or quasi-real time. So, for the task of fast shape symmetry calculation on the common multicore PC we elaborated another parallel program, which based on the procedure suggested before in (Fedotova, 2016). That method takes as an initial axis the axis obtained by superfast comparison of two skeleton primitive sub-chains. This process takes about 0.5 sec on the common PC, it is considerably faster than any of the optimized brute-force methods including ones implemented in supercomputer. In our experiments for 70 percent of cases the found axis coincides with the ground-truth one absolutely, and for the rest of cases it is very close to the ground-truth.

1. INTRODUCTION

By analyzing binary images it is easy to notice that shapes of many objects, both artificial and natural, have the intrinsic property of reflection (bilateral) symmetry. It is obvious that real-world images can rarely be absolute reflection-symmetric. So, it is valuable to detect approximate reflection symmetry and evaluate the symmetry measure of a shape (see Figure 1). Estimation of symmetry could be used in many computer vision applications like analysis of plants growing conditions or bilateral symmetry of insects.

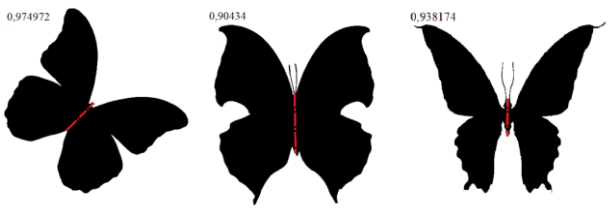

Figure 1. Examples of shapes with symmetric axis corresponding to the maximum symmetric measure according to Jaccard similarity

The problem of approximate symmetry detection and symmetry measure calculation applying to binary images is well known but there are a few efficient methods for solving it. The main ones are based on: 1) Fourier series expansion of parametric contour representation (van Otterloo, 1988), 2) contour representation by turning function (Sheynin, 1999), 3) contour representation by critical points and computation of similarity measure for two sub-contours via vectors of geodesic distances (Yang, 2008), 4) pair-wise comparison of sub-sequences of skeleton primitives (Kushnir, 2016). All mentioned methods exploit known algorithms of shapes dissimilarity (or similarity) evaluation.

In order to assess and compare results of above-mentioned efficient methods the exact algorithm of reflection symmetry detection was proposed (Kushnir, 2016). It allows evaluating ground-truth symmetry axis and the symmetry measure as well.

The exact algorithm is based on pair-wise complete enumeration of the shape outer contour points – lines are drawn through all possible pairs, each of them is considered as a potential symmetry axis as follows. A line divides a shape into two parts; each part is represented as a set of pixels. Similarity between two sets is evaluated using the Jaccard measure (for binary sets also known as Tanimoto):

$$\mu_T(B) = \frac{|S(B) \cap S(B_r)|}{|S(B) \cup S(B_r)|} \quad (1)$$

where B is the binary image, the brightness of the black pixels is coded by 1, and the white ones is coded by 0; B_r is reflection of binary image B relative to straight line, $S(B)$ is the set of pixels belonging to image B, the brightness code of which is equal to 1.

The line which divides a shape into two most similar sets (with the least Jaccard measure among all possible sets) is considered as a sought-for symmetry axis. Unfortunately, values of Jaccard measure, being calculated for all pairs of shape contour points, form extremely sophisticated surface in a search space (see Figure 2), so we have to use the brute-force enumeration algorithm for symmetry axis search but it is extremely time-

consuming – it performs dozens and even hundreds of minutes in an ordinary PC.

It was the reason why two accelerated versions of the algorithm were proposed in (Kushnir, 2016; Fedotova, 2016); the first of them is called the optimization considering semi-perimeters of a shape and the second – optimization considering the center of mass of a shape. Although such optimizations significantly less computationally expensive, the speed of their execution isn't sufficient for processing large databases. They also need the correct set of input parameters to be chosen in order to achieve the most precise result, which makes the processing of large databases very complicated and laborious.

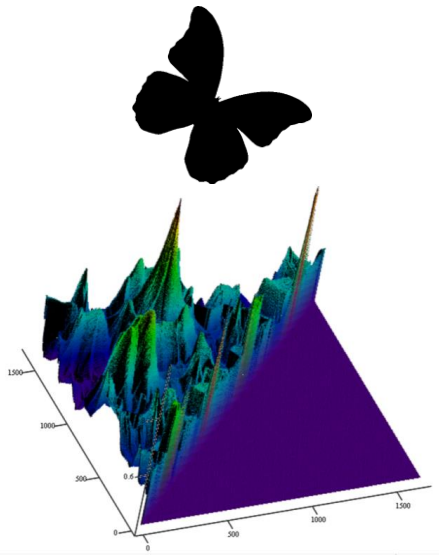

Figure 2. The example of a binary image and the surface formed by values of Jaccard measure calculated for all $N \cdot (N-1)/2$ pairs of contour points

In this paper two parallel versions of the algorithms proposed in (Fedotova, 2016) are described. The first one is the parallel version of brute-force enumeration for supercomputer system which allows evaluating accurate symmetry measures and symmetry axes for large image databases within reasonable time. Such automatically labeled image databases could be used as ground-truth in debugging and testing new fast procedures for searching symmetry axis. The second algorithm is designed as the parallel version of the one of such fast methods – the method of symmetry axis adjustment (Fedotova, 2016). This method takes as an initial axis the axis obtained by superfast comparison procedure of two skeleton primitive sub-chains (Kushnir, 2016). The parallel version processes an image within sub-second time on the common PC with multi-core processor. Such performance allows solving image analysis applications in real-time conditions and getting the best or very close to the best result.

2. PARALLEL IMPLEMENTATION OF ALGORITHMS FOR FINDING THE REFLECTION SYMMETRY OF THE BINARY IMAGES

2.1 Parallel version of the reflection-symmetry axis detection algorithm for a supercomputer based on complete enumeration

Brute-force search algorithm always finds the best axis of symmetry, but the computational complexity of its operation depends quadratically on the number of points in the shape contour. It takes on average about two hours to process a single image with a resolution of 800 by 600 pixels on a normal laptop. It was decided to apply parallelization technologies in order to use significant computing power and achieve the appropriate evaluation time on large image datasets. In particular, we used resources of "Lomonosov" (Sadovnichy, 2013; Voevodin, 2012) supercomputer complex at Moscow State University (http://hpc.msu.ru/).

The serial realization of brute-force algorithm based on complete enumeration of all lines - candidates to the desired axis of symmetry:
1. The contour of the shape is represented as a sequence of its boundary points.

2. Take a point from the sequence.

3. Iterate through the remaining points; each of them, together with the point selected in step 2 determines some line. Calculate and store the value of symmetry measure (1) relative to each resulting line. When all possible lines passing through the point chosen in step 2 are examined, this point should be excluded from further iterating.

4. Repeat steps 2 and 3 for a decremented sequence of contour points.

As a result, all the possible lines crossing the shape will be examined. The sought-for symmetry axis will be that of the lines relative to which the maximum symmetry measure is got.

It should be noted that calculation of symmetry measure with respect to each line is performed independently, that means the existence of data parallelism. It can be used in a natural way to develop a parallel version of this algorithm by distributing the handled lines between the application processes implementing the exact algorithm.

To distribute $L = C_N^2 = \frac{N!}{(N-2)! \cdot 2!} = \frac{(N-1)N}{2}$ lines between P processes, the following scheme is used in this article: first $L \bmod P$ processes get $l = (L - L \bmod P)/P + 1$ lines, other $L - L \bmod P$ processes get $l = (L - L \bmod P)/P$ lines, where $\bmod$ is the modulo operation which gives the remainder after division of one integer by another.

The parallel realization of brute-force algorithm based on complete enumeration for P processes:

1. Each process defines a set of lines to handle according to the scheme described above.

2. Each process iterates through the lines from its set, calculates a measure of symmetry for each line and selects the line which gives the maximum value of symmetry measure.

3. Each process transfers information about the selected line to the zero process.
4. The zero process receives information about detected lines from all other $P-1$ processes and selects among P lines (including one found by itself) the one which gives the maximum value of symmetry measure.

It is obvious that such algorithm construction guarantees that a line which gives the maximum value of symmetry measure among the complete set of L lines will be found, and which, therefore, will be a sought-for axis of symmetry.

This algorithm was implemented in C++ programming language by using MPI parallel programming technology (Michael, 2004) in order to organize data exchange between parallel processes. Selection of MPI programming technology can be justified by the following factors: 1) peculiarities of the algorithm, which contains many independent fragments requiring long calculations, 2) peculiarities of architecture of used computer system – the "Lomonosov" MSU supercomputer complex has several thousands of eight-processor nodes. The developed implementation of the proposed parallel algorithm allows using resources of the computer system in the most efficient way.

Section 3 shows the results of performance tests for the parallel implementation using the resources of the "Lomonosov" Moscow State University supercomputer complex. It is able to reduce time of a symmetry axis searching to 1-5 minutes per image for images from the "Butterflies" database – compared with about 2 hours per image by utilizing serial implementation. So, the obtained results show the efficiency of the proposed parallel method. It became possible to find symmetry axes for large image databases. For example, the entire Flavia database, which consists of 1907 binary images of plant leaves (Wu, 2007) with a resolution of 800 by 600 pixels, was processed in less than 9 hours using 1024 processes.

However, the work in production conditions requires speeds comparable to real time (tens of milliseconds in streaming video processing), or close to real (hundreds of milliseconds for performing operations by industrial robots), and these speeds could be achieved on personal computers. A parallel version of the fast symmetry axis searching algorithm implemented for personal computer is described below.

2.2 Parallel version of the algorithm based on adjustment of reflection symmetry axis found with fast numeric method implemented for a personal computer

A parallel implementation of the exact complete enumeration algorithm for symmetry measuring is not applicable to problems of real-life decision-making, because it is impossible to involve a supercomputer for everyday tasks solving. So, another parallel realization for a conventional personal computer with multi-core processor was developed. It is based on the numerical method which adjusts a reflection symmetry axis found by the fast skeleton primitive sub-chains comparison algorithm (Kushnir, 2016). This method, in general, gives an approximate solution, but has an extremely high operation speed.

Preliminary studies have shown that axis found by skeleton primitive sub-chains comparison algorithm usually gives a smaller value of symmetry measure in comparison with the axis, obtained by exact complete enumeration algorithm, which always gives the maximum value of the symmetry measure for the same image. Nevertheless, the skeleton axis is located so that it crosses the contour of the shape in an ε-neighborhood of each of the points of intersection of the exact symmetry axis with the contour of this shape. Thus, the proposed approach is that we refine axis found by skeleton method, i.e. find a line in its neighborhood, which gives a value of symmetry measure larger than the value of symmetry measure corresponding to adjusted axis, that is, we improve the accuracy of skeleton method. The refined skeleton axis is called the seed axis, and any candidate axis in searching process is the probe one. The symmetry axis necessarily crosses the contour of the object, so we consider only the boundary points of the shape to get probe axes. The boundary of the image is represented by a sequence of points numbered as $0...N-1$.

The results presented in (Fedotova, 2016) show that the second variant of the symmetry adjustment algorithm, based on the location of the center of mass of the figure, has the best timing and accuracy characteristics. The algorithm implies the fact that the symmetry axis must pass through the center of mass of the absolute symmetric shape.

As a rule, the axis of the approximate symmetric shape does not pass exactly through the center of mass, but in its neighborhood, which we will consider as a circle with a center coinciding with the center of mass of the shape. The radius R of such a circle is calculated as $k_R \cdot D$, where D is the distance from the center of mass to the outermost point of the contour, k_R is the coefficient of proximity to the center of mass. If there are a priori knowledge about the quality of the seed axis, then it will be enough to go through only those test axes that are not further than the distance R from the center of mass specified through the parameter k_R. Only they cross the circle with a radius centered at the center of mass.

This algorithm also has some resource of internal parallelism, connected with search of all probe lines in neighborhood and calculation of the corresponding values of the symmetry measure.

These operations, similar to the previous case, are independent and can be performed in parallel for different test lines. The use of parallel computations in this case leads to additional increase in productivity, which is the goal of this work.

The parallel version of the algorithm has to be developed for a conventional personal multi-core computer that is why OpenMP (Michael, 2004) was chosen as the parallel programming technology. It is designed for shared memory systems and provides convenient tools for manipulating threads within a single application.

The algorithm of adjustment of a symmetry axis found by the skeleton method, taking into account the center of mass of the figure, for T threads is follow:

1. The seed axis is defined by two points p_1 and p_2 of intersection of shape contour with the symmetry axis found by the skeleton method.
2. Calculate the points that are in a predetermined ε-neighborhood of the first point: $a = p_1 - \varepsilon$, $b = p_1 + \varepsilon$. These two points (a and b) limit the finite set of certain points of the contour $[a;b]$. Determine the points that are in a predetermined ε-neighborhood of the second point: $c = p_2 - \varepsilon$, $d = p_2 + \varepsilon$. These two points (c and d) limit the finite set of certain points of the contour $[c;d]$.
3. On the segments $[a;b]$ and $[c;d]$ select the sets of equidistant points $Q = \{q_i = a + h \cdot i\}$, $S = \{s_i = c + h \cdot i\}$,

$i = 0,..,n$, where n – the number of segmentation parts, h – the stride of partition, which is calculated as $2\varepsilon / n$.

4. Iterate through all pairs of the points belonging to sets Q and S obtained in step 3. This enumeration defines $l = |Q| \cdot |S|$ probe lines; processing of them is distributed statically among T threads. Calculate the symmetry measure relative to each obtained probe axis, located in R-neighborhood of the center of mass, store one of them for which the maximum value of symmetry measure is obtained.

5. Choose the one line which gives the maximum value of symmetry measure among T lines represented by each thread.

6. If the stride h is bigger than 1, two contour points p_1' and p_2' belonging to the line which gives the maximum value of symmetry measure, are passed to step 2: $p_1 := p_1'$, $p_2 := p_2'$, $\varepsilon := h$, otherwise pass to step 7.

7. If any of the points a, b, c, d belongs to obtained line, this line is declared as the seed one and its points p_1' and p_2' are passed to step 1, otherwise this line is a sought-for symmetry axis.

The proposed algorithm finds the axis, which gives the symmetry measure that is not less than the measure obtained by the skeleton method.

The proposed parallel version of the adjustment algorithm is implemented in C++ using OpenMP parallel programming technology.

3. EXPERIMENTAL STUDY

We have implemented experimental study of proposed methods on “Butterflies” database, consists of 30 images with a resolution of 400 by 600 pixels (see Figure 3).

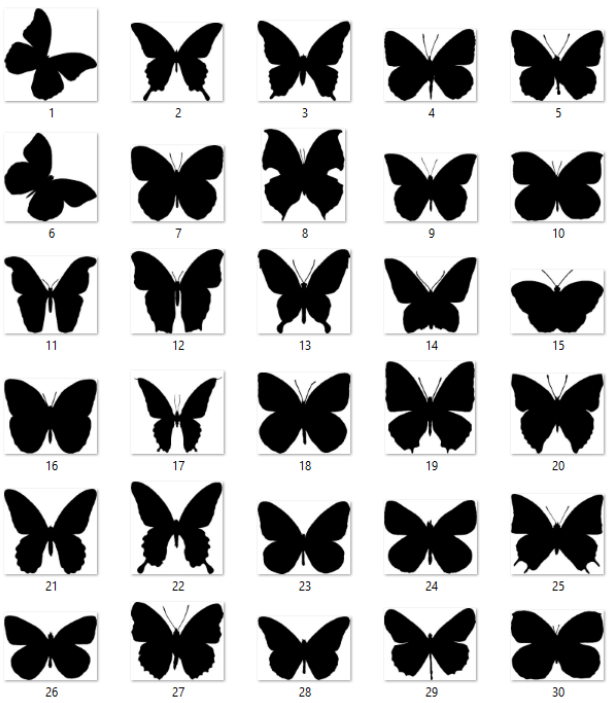

Figure 3. “Butterflies” database

The experiments with exact brute-force algorithm were performed in “Lomonosov” supercomputer at Moscow State University. Table 1 demonstrates results for all 30 images from “Butterflies” dataset, namely, values of symmetry measure, processing time of each image for sequential and parallel versions, and achieved speedup. Parallel version examined on 64 processes. The implementation of complete enumeration algorithm allows parallelizing $1-\alpha = 99,93\%$ of computations, where $\alpha = 0,07\%$ is a part of sequential computations in whole process. So, the theoretical acceleration (Amdahl's law) for the system of $p = 64$ processors does not exceed $S_p = \left(\alpha + \frac{1-\alpha}{p}\right)^{-1}$, i.e. $S_{64} = \left(0,0007 + \frac{0,9993}{64}\right)^{-1} \approx 61,3$.

The real speedup got in our experiments is as much as 58 times, which is close to theoretical estimation.

Images from “Butterflies” database				
Image number	Symmetry measure	Number of threads		Acceleration for increasing number of threads, times
		1	64	
		Time in seconds		
1	0,974	3469,73	59,68	58,14
2	0,940	8640,51	147,61	58,53
3	0,925	9121,3	156,62	58,24
4	0,873	11743,7	208,46	56,34
5	0,852	14633,6	240,91	60,74
6	0,917	4696,73	81,48	57,64
7	0,903	10410,6	182,03	57,19
8	0,904	5456,33	92,16	59,21
9	0,9213	11460,4	189,07	60,61
10	0,847	11529,1	204,06	56,50
11	0,935	6504,34	110,16	59,04
12	0,931	5741	104,44	54,97
13	0,938	5922,68	99,52	59,51
14	0,956	5745,89	102,85	55,87
15	0,962	4847,06	84,80	57,16
16	0,936	5781,57	96,05	60,20
17	0,938	6590,04	116,24	56,69
18	0,856	8570,97	141,06	60,76
19	0,903	7802,34	129,74	60,14
20	0,928	7021,68	117,79	59,61
21	0,945	5541,84	97,51	56,83
22	0,935	5181,28	89,52	57,88
23	0,917	6739,13	118,61	56,82
24	0,895	9412,85	166,89	56,40
25	0,925	14363,6	235,49	61,00
26	0,901	9320,4	158,42	58,83
27	0,881	13511,5	225,11	60,02
28	0,941	3217,98	56,82	56,63
29	0,922	3554,41	62,60	56,78
30	0,795	8561,1	146,89	58,28

Table 1. Results of sequential and parallel versions of exact symmetry detection algorithm implemented on supercomputer

The realization of parallel adjustment algorithm seeded by skeleton procedure was tested on PC (Intel® Core™ i7-2670QM CPU @ 2.2GHz, 16 Gb). Table 2 demonstrates results for all 30 images from "Butterflies" database, namely, etalon values of symmetry measures obtained by exact brute-force algorithm, values of symmetry measures obtained by adjustment algorithm, processing time of each image for sequential and parallel versions, and achieved acceleration. Adjustment algorithm with parameters $k_R = 0,03$, $\varepsilon = 0,25$ was tested on 2 and 4 threads. Code analysis of this algorithm realization shows that $\alpha = 15,3\%$; and according to principles described above the theoretical speedup is $S_2 = \left(0,153 + \frac{0,847}{2}\right)^{-1} \approx 1,73$ and $S_4 = \left(0,153 + \frac{0,847}{4}\right)^{-1} \approx 2,74$. The average time for test study shows the acceleration of 1,158 times for two threads.

Images from "Butterflies" database								
Image number	Exhaustive search on the supercomputer with 64 processors		Symmetry axis adjustment algorithm run on the PC Intel® Core™ i7-2670QM CPU @ 2.2GHz, 16 Gb					
	Symmetry measure	Time in seconds	Symmetry measure	Number of threads: Time in seconds, 1	2	4	Speedup with 2 threads	Speedup with 4 threads
1	0,974972	3469,73	0,974972	0,553	0,487	0,429	1,136	1,289
2	0,940896	8640,51	0,940896	0,652	0,552	0,553	1,180	1,178
3	0,925644	9121,3	0,925644	0,566	0,489	0,488	1,157	1,161
4	0,873039	11743,7	0,79014	0,522	0,472	0,448	1,107	1,166
5	0,852842	14633,6	0,852842	0,637	0,543	0,541	1,173	1,179
6	0,91712	4696,73	0,91712	0,674	0,550	0,533	1,225	1,263
7	0,90349	10410,6	0,90349	0,656	0,561	0,556	1,170	1,181
8	0,90434	5456,33	0,89266	0,584	0,498	0,437	1,173	1,337
9	0,921313	11460,4	0,921313	0,684	0,681	0,645	1,004	1,061
10	0,847191	11529,1	0,847191	0,685	0,644	0,632	1,065	1,083
11	0,935819	6504,34	0,935819	0,523	0,476	0,442	1,097	1,182
12	0,931439	5741	0,931439	0,538	0,493	0,448	1,093	1,201
13	0,93892	5922,68	0,850256	0,257	0,236	0,216	1,092	1,189
14	0,956936	5745,89	0,819866	0,383	0,342	0,316	1,121	1,215
15	0,962781	4847,06	0,962781	0,525	0,522	0,489	1,005	1,073
16	0,936332	5781,57	0,936332	0,507	0,491	0,450	1,032	1,128
17	0,938174	6590,04	0,938174	0,678	0,618	0,582	1,098	1,166
18	0,85629	8570,97	0,703983	0,459	0,411	0,399	1,118	1,150
19	0,903032	7802,34	0,903032	0,440	0,395	0,372	1,114	1,181
20	0,928616	7021,68	0,737619	0,378	0,340	0,334	1,110	1,131
21	0,94528	5541,84	0,94528	0,460	0,420	0,408	1,096	1,127
22	0,935842	5181,28	0,935842	0,437	0,391	0,367	1,120	1,193
23	0,917868	6739,13	0,749053	0,595	0,570	0,539	1,045	1,105
24	0,895583	9412,85	0,895583	0,643	0,679	0,637	0,948	1,009
25	0,925221	14363,6	0,925221	0,669	0,596	0,617	1,122	1,083
26	0,900524	9320,4	0,900524	0,703	0,651	0,653	1,080	1,076
27	0,881277	13511,5	0,880764	0,595	0,583	0,562	1,020	1,058
28	0,940923	3217,98	0,940923	0,491	0,437	0,401	1,124	1,224
29	0,921864	3554,41	0,921864	0,478	0,376	0,346	1,272	1,384
30	0,794933	8561,1	0,794714	0,665	0,610	0,621	1,089	1,070

Table 2. Results of sequential and parallel versions of symmetry detection algorithm implemented on PC

4. CONCLUSION

Proposed parallel algorithm of exact search allows finding the etalon symmetry axis within reasonable time using massive processing power (with the aid of supercomputer). Experimental studies implemented on "Butterflies" and "Flavia" image datasets have shown that the proposed algorithm takes several minutes per image to find the symmetry axis. So, it is acceptable for automatic labeling of databases consisting of hundreds and thousands of images in order to debug and test new fast procedures for searching symmetry axis.

However, in case of real-world applications we need computational efficiency which allows solving the task of symmetry axis search in real time (dozens of milliseconds in video processing tasks) or quasi-real time (hundreds of milliseconds for industrial robots). Moreover, it is not possible to use supercomputers for solving everyday computer vision tasks. So, for the task of fast symmetry calculation on the common multicore PC we elaborated another parallel program, which based on the procedure suggested before in (Fedotova, 2016). This method takes as an initial axis the axis obtained by superfast comparison procedure of two skeleton primitive sub-chains. In this case we use the OpenMP technique which makes possible to find an axis that close to the ground-truth axis of symmetry. This process takes about 0.5 sec on the common PC that is considerably faster than any of the optimized exact brute-force methods including ones implemented on supercomputer.

According to Table 2, the adjustment algorithm finds correct symmetry axis for the 21 images of 30 (green filler in the table). For other nine images the adjusted axes are very close to the etalon (see Figure 4). An inexact final decision depends on bad seed axis or too complicated contour configuration in the search area (feelers of a butterfly).

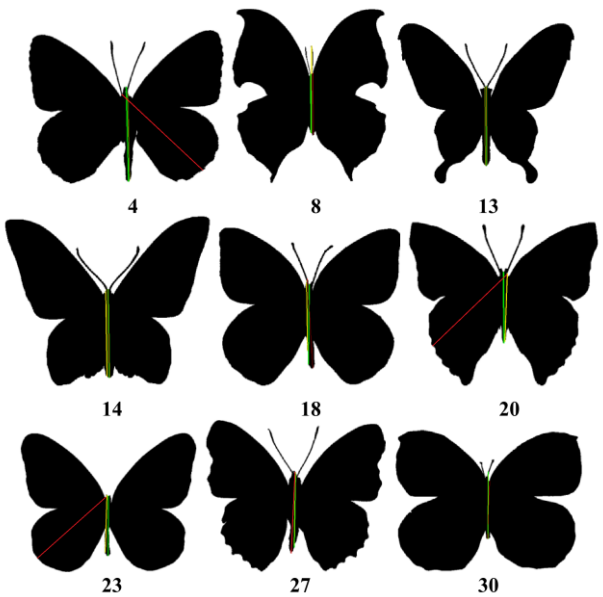

Figure 4. Cases from "Butterflies" database where adjusted axis does not coincide with etalon.
Green - etalon axis, red – seed axis, adjusted – yellow axis

ACKNOWLEDGEMENTS

This work is supported by the Russian Fund for Basic Research, grant 16-57-52042. The authors would like to thank rector of Lomonosov Moscow State University Viktor Sadovnichiy and Moscow State University Supercomputing Center for providing "Lomonosov" supercomputer complex to perform experimental study; and Dr. Valentina Sulimova for fruitful discussion.

REFERENCES

van Otterloo, P. J. A, 1988. Contour-Oriented Approach to Digital Shape Analysis // PhD thesis, Delft University of Technology, Delft, The Netherlands.

Sheynin, S., 1999. Computation of Symmetry Measures for Polygonal Shapes / S. Sheynin, A. Tuzikov, D. Volgin // Computer Analysis of Images and Patterns. Springer Berlin Heidelberg. pp. 183–190.

Yang, X., 2008. Symmetry of shapes via self-similarity / X. Yang, N. Adluru, L. J. Latecki, X. Bai, Z. Pizlo // Advances in Visual Computing, Springer Berlin Heidelberg. pp. 561–570.

Kushnir, O., 2016. Reflection Symmetry of Shapes Based on Skeleton Primitive Chains / O. Kushnir, S. Fedotova, O. Seredin, A. Karkishchenko // Fifth International Conference, AIST 2016, Yekaterinburg, Russia, April 7–9, 2016, Revised Selected Papers, CCIS, Springer International Publishing Switzerland. Vol. 661. – pp. 281–292.

Vl. V. Voevodin, 2012. Practice of 'Lomonosov' Supercomputer / Vl. V. Voevodin, S. A. Zhumatiy, S. I. Sobolev, A. S. Antonov, P. A. Bryzgalov, D. A. Nikitenko, K. S. Stefanov, and Vad. V. Voevodin (in Russian), Otkrytye Sistemy (Open Systems), No. 7, pp. 36-39.

Sadovnichy, V., 2013. "Lomonosov": Supercomputing at moscow state university. / Sadovnichy, V., Tikhonravov, A., Voevodin, V., and Opanasenko, In Contemporary High Performance Computing: From Petascale toward Exascale, Chapman & Hall/CRC Computational Science, pp. 283–307, Boca Raton, United States.

Fedotova S., 2016. The Algorithms of Adjustment of Reflection Symmetry Axis Found by the Skeleton Primitive Sub-Chains Comparison Method / Fedotova S., Seredin O., Kushnir O. // Intelligent Data Processing: Theory and Applications: Book of abstracts of the 11th International Conference (Moscow, Russia – Barcelona, Spain). 2016. – Moscow: TORUS PRESS. – pp. 110–111.

Wu, S. G., 2007. A leaf recognition algorithm for plant classification using probabilistic neural network // 2007 IEEE international symposium on signal processing and information technology. – IEEE, pp. 11–16.

Michael, J. Q., 2004. Parallel Programming in C with MPI and OpenMP // Dubuque, IA: McGraw-Hill.