Puzzle Form Images: How Does It Works?

Arkadiusz Drogoń
Faculty of Applied Mathematics
Silesian Univeristy of Technology
Gliwice, Poland
Kaszubska 23
arkadiusz.drogon@interia.pl

Karol Duda
Faculty of Applied Mathematics
Silesian Univeristy of Technology
Gliwice, Poland
Kaszubska 23
kmj.duda@gmail.com

Barbara Smoleń
Faculty of Applied Mathematics
Silesian Univeristy of Technology
Gliwice, Poland
Kaszubska 23
smolen94@gmail.com

***Abstract*—Image processing techniques are important algorithm both in applied mathematics and various aspects of computer science. The aim of this paper is to present the algorithm composing an image from the parts. We consider two cases. First, when the image is divided only vertically, and second, when the image is divided vertically and horizontally. We discuss our algorithm and also present appropriate statistics associated with the program's running time.**

***Index Terms*—image processing, Mathematica, images, composing algorithm**

I. Introduction

In today's world, computers play a very important role. They are used in every area of life and science. Today, artificial intelligence (AI) is one of the major directions of science. These methods combined with developing computer technology create enormous opportunities for science.

Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too. Digital image processing techniques help in manipulation of the digital images by using computers. These techniques have many advantages over analog image processing because computer allows a much wider range of algorithms to be applied to the input data. For example, pattern recognition uses algorithms such as neural networks, which are based on machine learning.

A very wide introduction to various image processing techniques was presented in [9]. Similarly in [10] was presented a discussion on positive aspects of algorithms, their complexity and implementations. In this book we can find many interesting algorithms for vision processing by the use of computer programming. In [7] were presented fundamental of image processing, while in [1] was given a discussion on possible applications. In [3] were presented geometric aspects of image processing, which are important for contour matching and therefore composition of images. A review on application of neural networks and their efficiency in image processing was presented in [5]. In [17] was discussed how to compose a detection algorithm which classifies shapes of bacterias from input images. In [16] was proposed a composition of techniques for detection of peel defects from images. While in [12] various models of computational intelligence were composed to detect obstacles on the way from camera images, and therefore advise users of the system about potential dangers on the road. In [14] was discussed an application of kernel tracking methodology for reconstruction of objects from input images. Recent advances in multi objective image processing were presented in [11]. Texture modeling for differences in pattern were discussed in [15], where a model of oscillations was used for matching objects details from input images. In [2] was presented how to use fuzzy systems to for recognition from images, and in [6] a discussion on nonlocal operators was presented for image processing.

Techniques of image processing use mathematical operations on matrices of images (because as we will see in section II, every digital image can be represented as a matrix). For example in application of filters in image processing, the values of points from its environment are taken into account in calculating the new value of the point. In this paper we present a proposal of the algorithm composing an image from the parts. The paper presents a simulation on composing puzzles from various images by the use of our method, where we compare significant pixels that match object shapes. We compare some features that describe complexity of this method and discuss its potential benefits.

II. Basics

The images we see on internet pages and the photos we take with our mobile phones are examples of digital images. It is possible to represent this kind of images using matrices. Mathematically the image can be seen as the matrix

$$F = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1M} \\ f_{21} & f_{22} & \dots & f_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & \dots & f_{NM} \end{bmatrix},$$

where $f_{ij},\ i = 1, \dots, N,\ j = 1, \dots, M$ represent colors of the pixels, N and M are sizes of image (respectively vertical and horizontal) given in pixels. Generally f_{ij} can be the vectors describing color in certain system (for example RGB). But in this paper we assume that for all images every f_{ij} is a number, which describe color in a greyscale, where set $\{0, 1, \dots, 255\}$ is transformed into $\{0, \dots, 1\}$. By this representation it is obvious, that we can operate on image using mathematical operations on its matrix.

III. Composition algorithm

Our algorithm is based on comparison of border pixels of components of image. As an input we take table of components of image in any order. In this paper M denotes table of transformations of those components into their matrix representations.

First we will present an algorithm for the case when the image is divided only vertically, so all components are strips with the same height as the image.

In the first step we create a matrix E of errors, where e_{ij} is an error between right side of M_i and left side of M_j, in the following way. For a diagonal of E we take a maximal value 1, because we do not want to link component of image with itself. For other elements mean value of difference between corresponding border pixels is computed.

In the second step we find coordinates i, j in the matrix E with minimal value of E, it means the images with "the best fit". We join the matrices of these images, it is M_i with M_j, which corresponds to the merging of these images. The new component replace M_i and M_j is deleted from M. Additional change to matrix E occurs: column i is replaced by column j and column j and line j are deleted, then diagonal elements of this matrix are changed to 1 again. We could compute the matrix E after every merge of components of images but transformation described above give us the same result and decreases amount of necessary computations. In package *Mathematica* this step can be realized in the following way

```
E1=E;
Do[
   If[E1[[i, j]] == Min[E],
    joinl = M[[i]];
   joinr = M[[j]];
     M[[i]] = Join[joinl, joinr, 2];
     M = Delete[M, j];
       E[[i]] = E[[j]];
       E = Drop[E, {j}, {j}];
     Break];
   , {j, 1, Length[M]}]
  , {i, 1, Length[M]}];
Do[E[[i, i]] = 1, {i, 1, Length[E]}];
```

Next second step is repeated with table M and matrix E adjusted in previous iteration. The implementation of the described technique is presented in following algorithm

Input: Table M of images
1: $r :=$ lenght of M
2: $mv :=$ vertical dimension of all M_i
3: Creating the matrix E
4: **for** $i = 0,\ i \leq r$ **do**
5: **for** $j = 0,\ j \leq r$ **do**
6: **if** $i \neq j$ **then**
7: $e_{ij} := \frac{1}{r} \sum_{k=1}^{mv} |M_i(k,-1) - M_j(k,1)|$
8: **else**
9: $e_{ij} := 1$
10: **end if**
11: **end for**
12: **end for**
13: **for** $k = 1,\ k < r$ **do**
14: $n :=$ length of M
15: $min :=$ Minimum value of E
16: **for** $i = 0,\ i \leq n$ **do**
17: **for** $j = 0,\ j \leq n$ **do**
18: **if** $e_{ij} = min$ **then**
19: replace M_i by joined M_i with M_j
20: delete M_j from M
21: replace in E column i by column j
22: delete column j and line j from E
23: break the loop
24: **else**
25: do nothing
26: **end if**
27: **end for**
28: **end for**
29: **for** $i = 1,\ i \leq$ lenght of E **do**
30: $e_{ii} := 1$
31: **end for**
32: **end for**
Output: M

In the second case, when the image is divided vertically and horizontally into rectangles of the same dimensions, we use modified algorithm.

As an input we take table M of components of image in any order and number of rows (the number of parts into which the image was divided horizontally). First we compute width of image. To obtain this we sum width of all components and divide it by number of rows. Now we create the matrix E of errors in the same way as in the algorithm for strips. Next we execute Algorithm 1, with one modification, until the same number of components as number of rows remains. At the end of the second step of the Algorithm 1 we check width of component M_i which is result of this step. If it is less than width of image then we proceed the next step. Otherwise all values in both the column i and the line i in matrix E are replaced by 1. Those modification ensures that when algorithm stops, remaining components have the same dimensions.

Finally, we transpose matrices of all remaining components and use Algorithm 1 again. As a result we obtain image reflected with respect to the diagonal.

The implementation of the described technique is presented in following algorithm

Input: Table M of images, number m of rows
1: $m_h :=$ horizontal dimension of all M_i
2: $r :=$ lenght of M
3: $mv :=$ vertical dimension of all M_i
4: $width := \frac{rm_h}{m}$
5: Creating the matrix E by executing of steps from 4 to 12 of the Algorithm 1.
6: **while** Length of $M > m$ **do**
7: Executing of steps from 14 to 22 of the Algorithm

1.
if (horizontal dimension of joined M_i with M_j) = $width$ **then**
for $t = 1$, $t \leq$ length of E **do**
$e_{it} = 1$
$e_{ti} = 1$
end for
else
do nothing
end if
end while
$\overline{M}$:= Table of transposed matrices from M.
Executing of Algorithm 1 with $\overline{M}$ as an input.

Output: $\overline{M}$

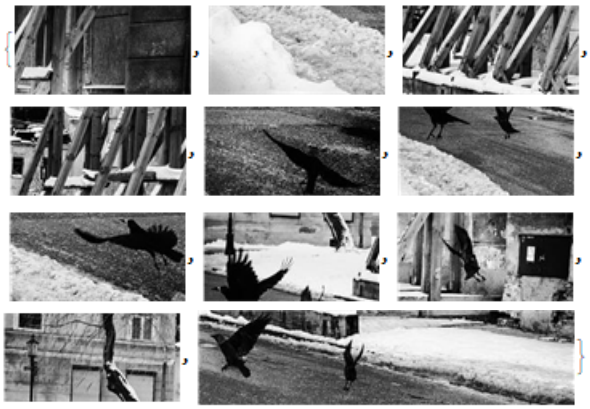
IV. Example

First example[1] is basic. We have three strips. In first iteration the minimal error was found for M_1 and M_3, so we join first and third part. Now we have only 2 parts and we need to check in which way we should join them, so we need to check which erros is smaller, e_{21} or e_{12}. e_{12} is smaller so we join the component M_1 with M_2.

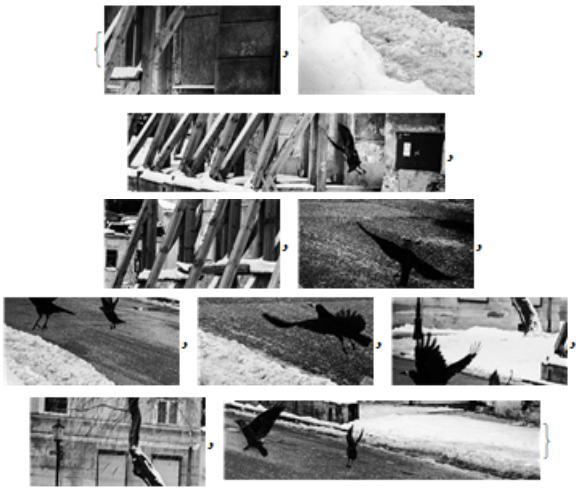
The next example shows how the algorithm works to divide an image into rectangles. We have 12 rectangles: 200×100 px. Below we have the program start.

In first iteration the minimal error was found for M_{12} and M_{10}, so we join 12th and 10th. As a result we have 11 parts. Next steps are similar. In each steps (I-VIII) we find the minimal error between all parts of the image and then we check the minimal error and join the corresponding parts of the the image.

II step (joining part 12 and 10)



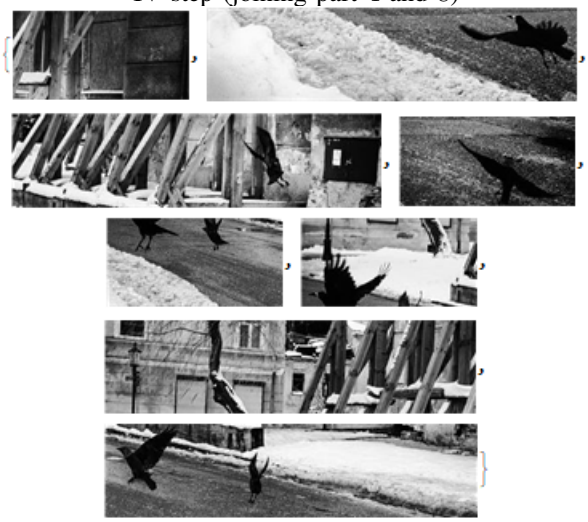
II step (joining part 3 and 8)



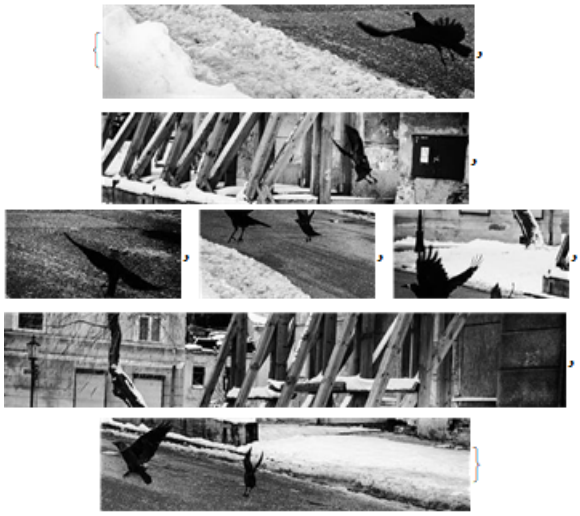
III step (joining part 2 and 6)

[1]To test the algorithm were used random images from Google Images.

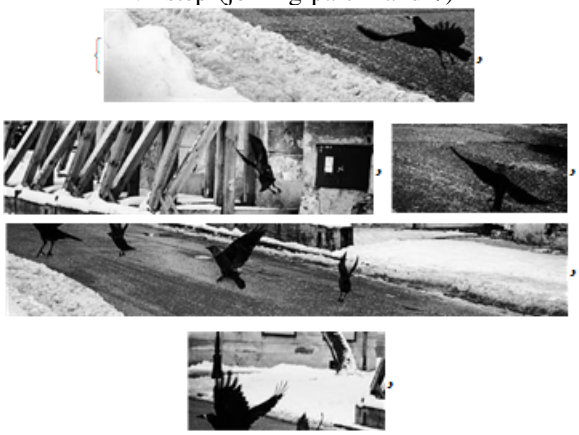
IV step (joining part 4 and 8)



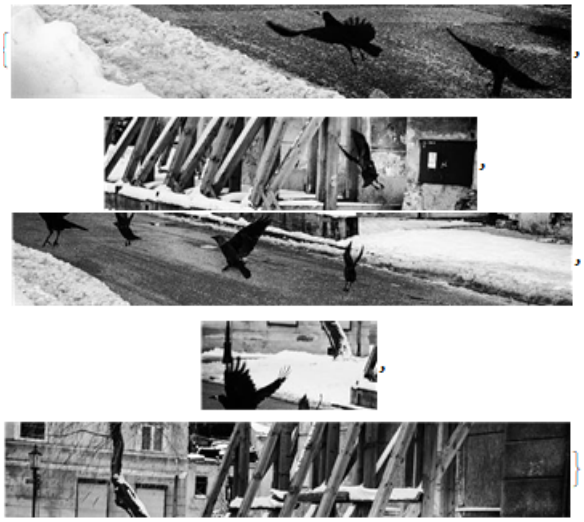
V step (joining part 1 and 7)



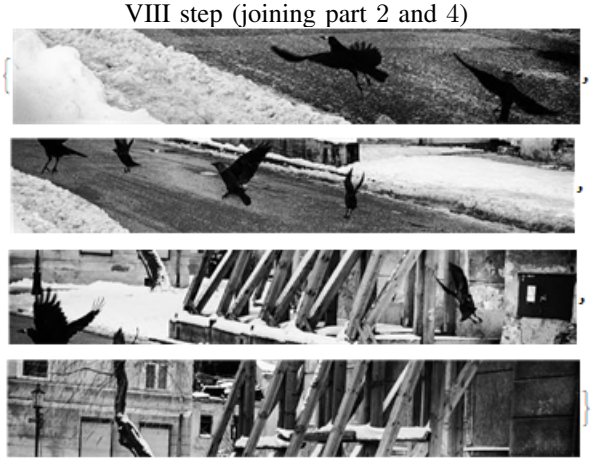
VI step (joining part 4 and 7)



VII step (joining part 1 and 3)



VIII step (joining part 2 and 4)

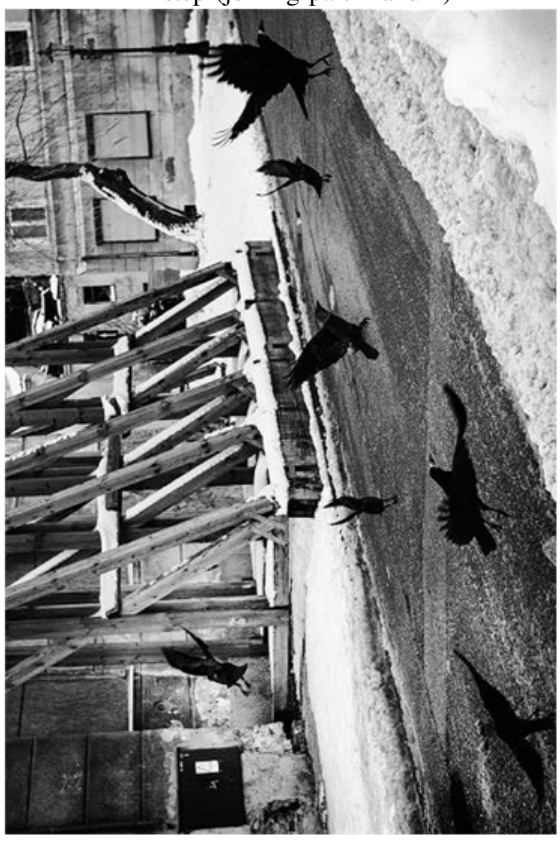


The IXth step is another than earlier steps. We have to transpose the matrix to continue the procedure. Next, we will continue the previously described procedure, i.e. we will compare all parts and join those for which the error e_{ij} will be the smallest.

XI step (joining part 3 and 2)

X step (joining part 4 and 3)

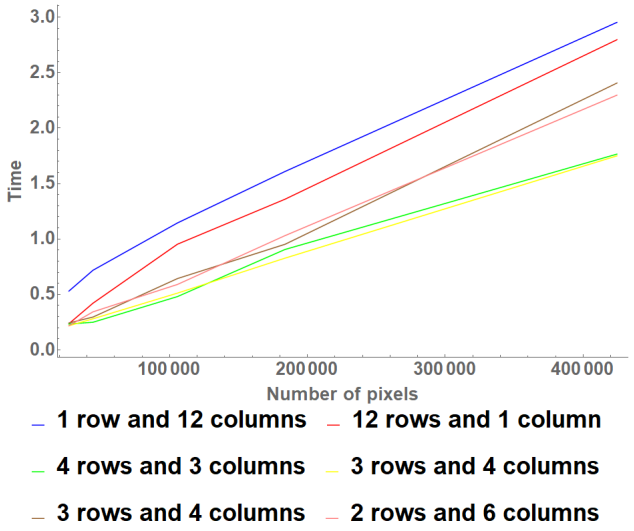
XII step (joining part 2 and 1)



XIII step (joining part 2 and 1)

V. Time of working

The basic problem of algorithms is their working time. We can compare the operation of the program for division the image only veritcally or veritcally and horizontally. The time of the program does not depend on the degree of differentiation of the image, because each time we compare the same number of pixels (for a fixed image size and its division). The duration of the program depends on the number of compared pixels and kind of divising. Of course if the size of the image (number of pixels) is greater, then the working time is greater. The graph below shows the approximate running time of the program for diffrent divisions.



We see that the relationship between image size and time is linear for each division: for only vertically and for vertically and horizontally division.

VI. Conclusion

The "Puzzles" algorithm can be extended to cases when the divisions are irregular, i.e. the sizes of vertical and horizontal divisions are different. Currently, such a situation is impossible, because the algorithm requires equal divisions. In addition, it can be seen that combining images may be incorrect for situations where the images are very similar or specific. Since our algorithm finds first components with minimal error, join them and proceed to the next step, we might obtain wrong result if there were more components with the same error. For example when the majority of the image is one color.

Another problem is time of working. For large images divided in many components, it might take relativily a lot of time for algorithm to stop. It is caused by amount pixels needed for algorithm to compare.

Algorithms described above may have applications in decision support systems that search for objects with characteristic attributes. Our algorithms may be used for finding elements of images that fits the fixed pattern.

References

[1] G. A. Baxes. *Digital image processing: principles and applications*. Wiley New York, 1994.

[2] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. Pal. *Fuzzy models and algorithms for pattern recognition and image processing*, volume 4. Springer Science & Business Media, 2006.

[3] V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische mathematik*, 66(1):1–31, 1993.

[4] R. Damaševičius, C. Napoli, T. Sidekerskienė, and M. Woźniak. Imf mode demixing in emd for jitter analysis. *Journal of Computational Science*, 22:240–252, 2017.

[5] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networksâĂŤa review. *Pattern recognition*, 35(10):2279–2301, 2002.

[6] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2008.

[7] A. K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[8] T. Kapuściński, R. K. Nowicki, and C. Napoli. Comparison of effectiveness of multi-objective genetic algorithms in optimization of invertible s-boxes. In *International Conference on Artificial Intelligence and Soft Computing*, pages 466–476. Springer, 2017.

[9] J. R. Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[10] T. Pavlidis. *Algorithms for graphics and image processing*. Springer Science & Business Media, 2012.

[11] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. *Remote sensing of environment*, 113:S110–S122, 2009.

[12] D. Połap, K. Kęsik, K. Książek, and M. Woźniak. Obstacle detection as a safety alert in augmented reality models by the use of deep learning techniques. *Sensors*, 17(12):2803, 2017.

[13] D. Połap, M. Wozniak, C. Napoli, and E. Tramontana. Is swarm intelligence able to create mazes? *International Journal of Electronics and Telecommunications*, 61(4):305–310, 2015.

[14] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on image processing*, 16(2):349–366, 2007.

[15] L. A. Vese and S. J. Osher. Modeling textures with total variation minimization and oscillating patterns in image processing. *Journal of scientific computing*, 19(1-3):553–572, 2003.

[16] M. Woźniak and D. Połap. Adaptive neuro-heuristic hybrid model for fruit peel defects detection. *Neural Networks*, 98:16–33, 2018.

[17] M. Woźniak, D. Połap, L. Kośmider, and T. Cłapa. Automated fluorescence microscopy image analysis of pseudomonas aeruginosa bacteria in alive and dead stadium. *Engineering Applications of Artificial Intelligence*, 67:100–110, 2018.

[18] M. Woźniak, D. Połap, C. Napoli, and E. Tramontana. Application of bio-inspired methods in distributed gaming systems. *Information Technology And Control*, 46(1):150–164.