A Simple yet Effective Image Repairing Algorithm

POST-PRINT

Article published on Lecture Notes in Computer Science, volume 13374
https://link.springer.com/chapter/10.1007/978-3-031-13324-4_7

Lidija Čomić
University of Novi Sad, Faculty of Technical Sciences,
Novi Sad, Serbia
comic@uns.ac.rs

Paola Magillo
University of Genova,
Department of Computer Science, Bioengineering, Robotics,
and Systems Engineering, Genova, Italy
magillo@dibris.inige.it

Abstract

A 2D binary image is well-composed if it does not contain 2×2 blocks of two diagonal black and two diagonal white pixels, called critical configurations. Some image processing algorithms are simpler on well-composed images. The process of transforming an image into a well-composed one is called repairing.

We propose a new topology-preserving approach, which produces two well-composed images starting from an image I depending on the chosen adjacency (vertex or edge adjacency), in the same original square grid space as I. The size of the repaired images depends on the number and distribution of the critical configurations. A well-composed image I is not changed, while in the worst case the size increases at most two times (or four times if we want to preserve the aspect ratio). The advantage of our approach is in the small size of the repaired images, with a positive impact on the execution time of processing tasks. We demonstrate this experimentally by considering two classical image processing tasks: contour extraction and shrinking.

keywords: Well-composed images, repairing 2D digital images, contour tracing, shrinking

1 Introduction

We consider two dimensional black-and-white (binary) images, where black is foreground and white is background. A critical configuration is a block of 2×2 pixels within an image, where two pixels are white and two are black, in a chessboard configuration. An image with no critical configurations is called well-composed.

The presence of critical configurations introduces ambiguity in the topology of the image, as the topological (homological) properties of the foreground and of the background of the image (the number of connected components and the number of holes) depend on the used adjacency type (edge- or vertex-adjacency, a.k.a. 4- or 8-adjacency). Opposite adjacency types must be used for foreground and background pixels in order to maintain some similarity between continuous and digital topology.

Furthermore, many topological image analysis and processing algorithms are simpler and easier to implement if their input image is known to be well-composed. For such reasons, the research community has been working for years on the topic of image repairing, i.e., the process of transforming an arbitrary image into a well-composed one. All the proposed approaches which preserve the image topology, increase the image size.

We propose a simple approach for topology-preserving image repairing, which is based on inserting new rows or new columns inside the image, just where critical configurations are present. In this way, the growing rate of image size depends on the number and distribution of critical configurations present in the image. We introduce two algorithms based on such an approach. Algorithm A guarantees less than 200% of size growth on all images, but modifies the aspect ratio. Algorithm B preserves the aspect ratio, but cannot guarantee the same bound. The theoretical worst case growing rate is 400%, while it is much less in practical cases.

2 Background Notions

A 2D (square) grid [1, 2, 5, 6, 7] is a tessellation of the plane into closed unit squares (pixels) centered at points in $\mathbb{Z}^2$, with edges parallel to the coordinate axes. Two types of adjacency relation are defined in the grid. Two pixels that share an edge or a vertex are called 4- or 8-adjacent, respectively.

A 2D digital object O is a finite set of pixels in the square grid. The pixels in O are called black (foreground). The pixels in the complement of O are called white (background). The carrier (or continuous analogue) of O is the union (as point sets) of the pixels in O. We will denote it also as O.

A vertex v is critical for a 2D digital object O if v is incident to two white and two black pixels, where black and white pixels alternate cyclically around v. The 2×2 pixels incident with a critical vertex are called a critical configuration, a.k.a. a gap.

3 Related Work

Several image repairing algorithms have been proposed. Here, we restrict our attention to the ones which preserve the topology, and whose output is still in the square grid.

The method by Rosenfeld et al. [10] scales the image by factor 3 in both x and y directions. In the rescaled grid, all black (white) pixels involved in a critical configuration are changed to white (black), for repairing the image according to 8-adjacency (4-adjacency). An example is shown in Figure 1 (b).

The algorithm by Stelldinger et al. [11] increases the grid resolution twice in both coordinate directions, by creating an additional square for each edge and each vertex in the grid. Therefore, the image size increases four times. If 4-adjacency (8-adjacency) is considered for the black pixels, the squares corresponding to the edges and vertices are black only if all the incident squares are black (at least one incident square is black). An example is shown in Figure 1 (c).

The algorithm by Čomić and Magillo [3] produces the output in a new square grid, rotated 45 degrees with respect to the original one and therefore called the (2D) diamond grid. The pixels of the diamond grid correspond to the pixels and vertices of the original square grid. Thus, the image size is increased two times.

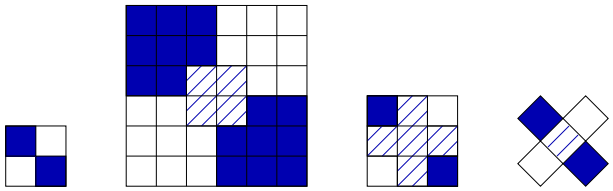


Figure 1: (A) A critical configuration and the way it is repaired by (b) Rosenfeld et al. [10], (c) Stelldinger et al. [11], and (d) Čomić and Magillo [3]. The dashed pixels are black (white) for preserving 8-adjacency (4-adjacency).

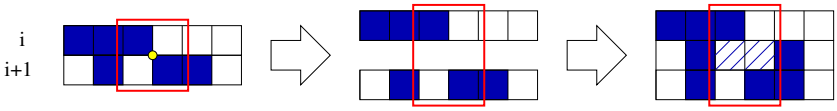


Figure 2: When a critical configuration (marked with the red box) exists across the rows i and $i+1$, a new row is inserted in between. The color of each pixel in the new row is copied from the row $i+1$, but the two pixels adjacent to those involved in the critical configuration (dashed) are both set to black or to white, for repairing the image according to 8-adjacency or 4-adjacency, respectively.

Depending on the choice of the color of the pixels associated with the vertices, the repaired image is homotopy equivalent with the original one with either 8- or 4-adjacency. An example is shown in Figure 1 (d).

4 Our Approach to Image Repairing

The idea is in a sense similar to [11], but, instead of adding a new row and a new column between each original row and column, we add a new row (or column) only where some critical configuration exists. For each critical configuration, one of the involved pixels changes color in the stretched image.

Our Algorithm A adds the minimum number of rows necessary to eliminate the critical configurations. Algorithm B adds both rows and columns, with the aim of preserving the aspect ratio of the image.

4.1 Algorithm A

Our basic idea is shown in Figure 2. For each pair of consecutive rows i and $i+1$ in the input image, such that some critical configuration exists across them, we add a new row between i and $i+1$. Such new row is a copy of the row $i+1$ in all pixels, with the exception of the pairs of consecutive pixels involved in a critical configuration. The color of such pairs is set to black (white) to obtain an equivalent image according to 8-adjacency (4-adjacency).

Equivalently, we can add new columns instead of new rows. We first compute the number of necessary new rows and the number of necessary new columns to be added, and then choose the option which gives the smaller image size. Algorithm A, obtained from this simple idea, has the following good properties:

1. It preserves the image topology with 4- or 8-adjacency.

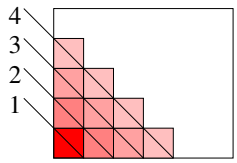


Figure 3: The image is examined diagonally. While scanning a diagonal of pixels, we check if the upper-right vertex of each pixel is critical.

2. The size of the repaired images is less than twice the size of the original one. In the worst case, for an image of size $N \times M$, a new row will be added between any two rows, and the size of the image will increase to $N \times (2M - 1)$.

3. The size increment depends on the number of critical configurations present in the input image. At most one row is added for each critical configuration. Intuitively, this is the first image repairing algorithm sensitive to the amount of repairing needed by the image.

The main drawback of Algorithm A is that the aspect ratio of the image is dramatically changed. So, the output of Algorithm A is feasible for processing the image to compute other information, but not for displaying it.

4.2 Algorithm B

In order to preserve the aspect ratio of the input image, Algorithms B adds both rows and columns in a balanced way across the image. Some critical configurations will be repaired by inserting a row, and other critical configurations by inserting a column.

We scan the image diagonally, as shown in Figure 3. Each time we find a critical configuration involving the four pixels across the rows $i, i+1$ and columns $j, j+1$, we compare the aspect ratio of the original image with that of the new image obtained with the already planned additions of rows and columns. We decide to add a new row or a new column, based on the choice that keeps the new aspect ratio more similar to the original one.

Figure 4 shows the same image repaired by adding just rows or just columns, or by adding both rows and columns.

Compared with Algorithm A, Algorithm B gives an aspect ratio which is very similar to the original one, but it does not guarantee the same bound on size growth. In the worst case, a chessboard pattern, Algorithm B would add a new row between any two rows, and a new column between any two columns. The image size would increase from $N \times M$ to $(2N - 1) \times (2M - 1)$, i.e., four times, as in [11].

5 Proof of Correctness

We show that the output of our repairing algorithms with respect to 8-adjacency is well-composed and homotopy equivalent to O. The claim for 4-adjacency follows by duality [7]. We focus on the insertion of a new row. The insertion of a column is symmetric. For the insertion of more rows (columns), it is sufficient to repeat the reasoning. For simplicity, we assume the critical configuration is as in Figure 5.

Figure 4: Test image *flamingo* at low resolution, and its repaired versions with Algorithms A and B. Critical configurations in the input image, and added rows and columns in the repaired ones, are rendered in red. In order to fit in the page, the images have been scaled 20 percent of their actual size.

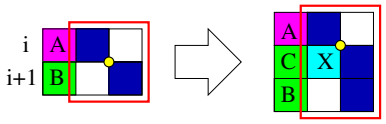


Figure 5: The configuration at a critical vertex (yellow). The color of the pixels of the new row is copied from the bottom row. Only the cyan pixel is changed to black in order to repair the critical configuration.

5.1 Well-Composedness

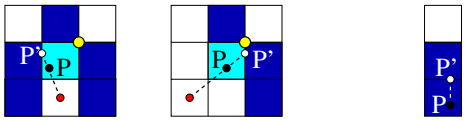
We show that after processing the rows i and $i+1$, all critical configurations between them are removed and no new critical configurations are created. Let X be the pixel in the duplicated row whose color is changed from white to black (the cyan pixel in Figure 5), and let us consider the 3×3 neighborhood of X. This color change will remove a critical configuration at the upper-right vertex of X (marked with the yellow dot), and will not create a new critical configuration at any of the other three vertices of X, whatever is the color of A and of B (the latter copied to C), as shown in Figure 5. The lower-right vertex of X is incident with exactly three black pixels (X and the two black pixels involved in the critical configuration), its lower-left vertex is incident with two 4-adjacent pixels of the same color (the pixels B and C) and its upper-left vertex is incident with two 4-adjacent black pixels (X and the one above it).

5.2 Homotopy Equivalence

Given a topological space X and its subspace A, a continuous function $F : X \times [0,1] \to X$ is a (strong) deformation retraction of X onto A if $F(x,0) = x$, $F(x,1) \in A$, $F(a,t) = a$ for all $x \in X$, $a \in A$ and $t \in [0,1]$. If a deformation retraction exists, then X and A are homotopy equivalent.

We show the homotopy equivalence by constructing a deformation retraction from the three rows processed at each step of the algorithms to the two rows of the original image.

Let $f(P,t) = (1-t)P + tP'$ for each point P in each added black pixel X, where P' is its radial projection



(a) (b)

Figure 6: The cyan pixel has been set to black in order to repair a critical configuration. The black dot is a point P inside a black pixel and the white dot is its image P' through f in (a) and through h in (b). The red dot is the center of the white pixel.

on the border of that pixel from the center of the pixel below X if the pixel to the left of X is black, or from the center of the lower left neighbor of X if that pixel is white (see Figure 6 (a)). Let $f(P,t) = P$ for all other points in the black pixels. Let $h(P,t) = (1-t)P + tP'$ for each point P in each black pixel in the third row, where P' is its vertical projection on the upper edge of that pixel, and let $h(P,t) = P$ for all other points in the black pixels in the first two rows (see Figure 6 (b)). The required deformation retraction is the composition of the maps f and h.

6 Experimental results and discussion

We used ten images from the Pixabay repository [9], with two different resolutions for each image. The images were gray-scale with gray values ranging from 0 to 255, and they have been converted to binary images by applying a threshold equal to 128 or 64 (depending on the darkness of the image). The low resolution versions of the images are shown in Figure 7.

In Section 6.1 we show and comment the results of Algorithms A and B. In Section 6.2 we consider the impact of the size of the repaired images in further processing algorithms, including a comparison with images repaired by other algorithms at the state of the art. All presented results refer to image repairing with 8-adjacency. The numbers would be the same with 4-adjacency, the only difference being the color of some pixels.

6.1 Results of Algorithms A and B

We developed a prototype implementation of Algorithms A and B, in Python. The results of the two algorithms on the test images are shown in Table 1. The suffix L or H refers to the same image at low and high resolution, respectively.

As expected, with Algorithm A, the size of all repaired images is less than twice the original one, and it depends heavily on the number of critical configurations in the input image. Also, the aspect ratio is relevantly changed. The size increment seems to be connected more with the number of critical configurations and less to resolution.

With Algorithm B, the aspect ratio is almost preserved, with changes occurring from the second decimal digit. On half of the images, the increment of image size is comparable to Algorithm A. On the other half, the size increases up to three times or more, especially on images with fine-grained patterns and many critical configurations (cfr. *birch*, *bird* and *train* in Figure 7).

Figure 7: Our binary versions of the original images. The shown images correspond to the low resolution and are scaled 15 percent to fit the page width.

6.2 Processing the repaired images

Image repairing is often used as a preprocessing stage, as many image analysis algorithms are simpler on well-composed images. In the following, we study the impact of the size of the repaired images produced by our Algorithms A and B and by the algorithms in [11] and [3] on the performance of image processing algorithms. We have chosen these competitors because they use the smallest additional memory among those which preserve image homotopy and produce an image in the square grid.

The algorithm by Stelldinger et al. [11] produces a repaired image whose size is four times the original one. The algorithm by Čomić and Magillo [3] doubles the size of the image, but the repaired image is in a grid rotated by 45 degrees. The size of the images repaired by Algorithms A and B is as in Table 1.

As meaningful examples of image processing tasks, we consider contour extraction and shrinking, i.e., a very simple and a rather complex task. A contour is a circular list of black pixels 8-adjacent to at least one white pixel, and 4-adjacent to the previous black pixel in the list (see [8], Chapter 7.5). Shrinking iteratively changes the color of simple (removable) black pixels into white (see [5], Chapter 16.2). For these programs, we used the C implementation from [3]. The results, obtained on a PC equipped with an Intel CPU i7-2600K CPU at 3.4 GHz with 32 GB RAM, are in Tables 2 and 3.

Table 1: Results with Algorithms A and B. Aspect ratios are rounded to the fourth decimal digit. The percentage of size growth is rounded to integer.

	input			version A			version B		
Image	size	crit. conf.	aspect ratio	size	aspect ratio	% incr. size	size	aspect ratio	% size
artL	640×426	378	1.5023	640×644	0.9938	151	785×523	1.5010	151
artH	1920×1280	5390	1.5	1920×2403	0.7990	188	2957×1972	1.4995	237
birchL	640×426	8253	1.5023	640×851	0.7521	200	1209×805	1.5019	257
birchH	1920×1279	74917	1.5012	1920×2557	0.7509	200	3702×2466	1.5012	372
birdL	640×426	1283	1.5023	640×801	0.7990	188	968×644	1.5031	229
birdH	1920×1279	8465	1.5012	1920×2531	0.7586	198	3217×2143	1.5012	280
carL	640×401	1003	1.5960	640×714	0.8964	178	928×581	1.5972	210
carH	1920×1205	5617	1.5934	1920×2314	0.8297	191	3037×1906	1.5934	250
flamingoL	425×640	175	0.6641	553×640	0.8641	130	480×722	0.6648	127
flamingoH	1276×1920	565	0.6646	1663×1920	0.8661	130	1441×2169	0.6644	128
fogL	640×235	320	2.7234	640×378	1.6931	161	786×289	2.7197	151
fogH	1920×706	2170	2.7195	1920×1298	1.4792	184	2655×977	2.7175	191
kiteL	640×524	717	1.2214	640×826	0.7748	158	824×674	1.2226	166
kiteH	1920×1571	3147	1.2222	2928×1571	1.8638	153	2549×2085	1.2225	176
staircaseL	640×417	1012	1.5348	640×646	0.9907	155	830×541	1.5342	168
staircaseH	1920×1253	6946	1.5323	1920×1970	0.9746	157	2598×1695	1.5327	183
stargazerL	640×426	108	1.5023	716×426	1.6808	112	690×459	1.5033	116
stargazerH	1920×1280	343	1.5	1920×1507	1.2741	118	2064×1377	1.4989	116
trainL	640×471	3438	1.3588	640×885	0.7232	188	1036×762	1.3596	262
trainH	1920×1421	15814	1.3512	1920×2752	0.6977	194	3235×2394	1.3513	284

Both processing tasks are faster on the images repaired by Algorithms A and B, than on the ones repaired by [11]. On the output images of Algorithm A, they are faster than on the output images of [3] in all cases with the exception of some images, where the times are comparable. On the images repaired by Algorithm B, contour extraction and shrinking are faster than on the images repaired by [3] in half of the cases.

Repaired images by Algorithm B give a better performance in cases where the original image had a small number of critical configurations (e.g., *flamingo, fog, stargazer*), but also in some other cases (e.g., *kite*). In these latter cases, probably many critical configurations were aligned, and therefore repaired by adding a single new row or column.

In presence of many critical configurations (e.g., *birch, bird, car, train*), the time for processing the output images of Algorithm B can be up to twice w.r.t. [3] (e.g., see shrinking on *birch* and *bird*). In the presence of few critical configurations, the opposite may happen (e.g., *flamingo* and *stargazer*). We remember that, unlike Algorithms A and B, the algorithm in [3] rotates the grid.

7 Conclusion

We proposed two repairing algorithms having the advantages that the obtained well-composed images lie in the square grid (thus, all existing image processing tools can be applied to the repaired images) and that the size

Table 2: Execution times (in milliseconds) of contour extraction on the images repaired by the four algorithms.

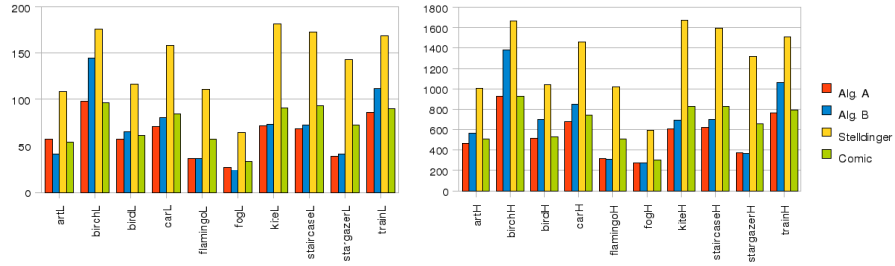
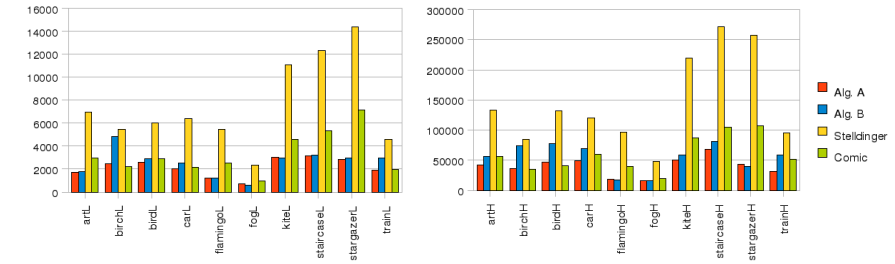


Table 3: Execution times (in milliseconds) of shrinking on the images repaired by the four algorithms.



of the output is sensitive to the number of critical configurations in the input.

Algorithm A has the further advantage of a reduced size of the repaired images, compared with all previous approaches. Its main drawback is a directional asymmetry: the repaired images are non-uniformly stretched in either vertical or horizontal direction. This makes Algorithm A less suitable for tasks that require the computation of numerical image properties such as area or perimeter, or of the Boolean operations on two distinct images, while algorithms for the computation of the topological properties can benefit from it.

Algorithm B solves this drawback at the expense of a larger size of the repaired images, if many critical configurations were present. It can be a good compromise for those cases where critical configurations are known to be few (e.g., coming from rasterization of vector formats and conversion errors).

In the future, we plan to introduce a mechanism to balance the number of added black and white pixels, to preserve the darkness/shininess of the image as well. This will improve the similarity of the repaired image with he original one. We also plan to port our prototype implementation into an efficient programming language, such as C.

Acknowledgement

This research has been partially supported by the Ministry of Education, Science and Technological Development through project no. 451-03-68/2022-14/ 200156

References

[1] V. E. Brimkov, A. Maimone, G. Nordo, R. P. Barneva, and R. Klette. The Number of Gaps in Binary Pictures. In *Advances in Visual Computing, First International Symposium, ISVC*, pages 35–42, 2005.

[2] V. E. Brimkov, D. Moroni, and R. P. Barneva. Combinatorial Relations for Digital Pictures. In *Discrete Geometry for Computer Imagery, 13th International Conference, DGCI*, pages 189–198, 2006.

[3] L. Čomić and P. Magillo. Repairing Binary Images through the 2D Diamond Grid. In *International Workshop on Combinatorial Image Analysis IWCIA*, volume 12148 of *Lecture Notes in Computer Science*, pages 183–198, 2020.

[4] R. González-Díaz, M.-J.. Jiménez, and B. Medrano. 3D well-composed polyhedral complexes. *Discrete Applied Mathematics*, 183:59–77, 2015.

[5] R. Klette and A. Rosenfeld. *Digital geometry. Geometric methods for digital picture analysis.* Morgan Kaufmann Publishers, San Francisco, Amsterdam, 2004.

[6] T. Y. Kong and A. Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48(3):357–393, 1989.

[7] L. J. Latecki, U. Eckhardt, and A. Rosenfeld. Well-Composed Sets. *Computer Vision and Image Understanding*, 61(1):70–83, 1995.

[8] T. Pavlidis. *Algorithms for Graphics and Image Processing.* Computer Science Press, 1982.

[9] PIXABAY. Image repository. `https://pixabay.com/en/photos/grayscale/`.

[10] A. Rosenfeld, T. Y. Kong, and A. Nakamura. Topology-Preserving Deformations of Two-Valued Digital Pictures. *Graphical Models and Image Processing*, 60(1):24–34, 1998.

[11] P. Stelldinger, L. J. Latecki, and M. Siqueira. Topological Equivalence between a 3D Object and the Reconstruction of its Digital Image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):126–140, 2007.