Repairing Binary Images through the 2D Diamond Grid

POST-PRINT

Article published on Lecture Notes in Computer Science, volume 12148
https://link.springer.com/chapter/10.1007/978-3-030-51002-2_13

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

Paola Magillo
DIBRIS, University of Genova, Italy
magillo@dibris.inige.it

Abstract

A 2D binary image is well-composed if it does not contain a 2×2 configuration of two diagonal black and two diagonal white squares. We propose a simple repairing algorithm to construct two well-composed images I_4 and I_8 starting from an image I, and we prove that I_4 and I_8 are homotopy equivalent to I with 4- and 8-adjacency, respectively. This is achieved by passing from the original square grid to another one, rotated by $\pi/4$, whose pixels correspond to the original pixels and to their vertices. The images I_4 and I_8 are double in size with respect to the image I. Experimental comparisons and applications are also shown.

keywords: Digital topology, Well-composed images, Repairing 2D digital binary images

1 Introduction

In 2D, an image I in the square grid is well-composed if it does not contain blocks of 2×2 squares with alternating colors in chessboard configuration [1, 2, 18, 19]. The process of transforming a given image into a well-composed one, that is in some sense similar to the original, is called *repairing*. Many image processing algorithms are simpler and faster when applied on well-composed images, making image repairing and study of different types of well-composedness a vivid research area [3, 4, 5, 8, 9].

Here, we address 2D image repairing by passing to another square grid, rotated by $\pi/4$ with respect to the original one and scaled by factor $1/\sqrt{2}$, in which each diamond (rotated square) corresponds either to a square or to a vertex in the original grid. In the rotated grid, we construct two well-composed images I_4 and I_8, homotopy equivalent to I with 4- and 8-adjacency, respectively.

The advantages of our approach are that the two repaired images are still in a square grid, so they can be processed with classical methods, in a simplified way thanks to well-composedness; we can choose between two types of adjacency in repairing; and the size of the resulting image in the diamond (rotated square) grid is just double that of the original one.

The contributions of this paper are:

- A simple repairing procedure, which results in two well-composed images I_4 and I_8, each of which is twice as large as the initial image I.
- A proof that the two repaired images I_4 and I_8 are well-composed and homotopy equivalent to the image I with 4- and 8-adjacency, respectively.
- Comparison with the state-of-the-art, showing the usefulness or our approach.

2 Background Notions

The square grid is the tessellation of the plane into Voronoi regions associated with points with integer coordinates. Each region (pixel) is a unit square, with sides parallel to the coordinate x and y axes [16, 17]. (For a set S of points in the plane, the Voronoi region associated to a point p in S contains all points in the plane that are closer to p than to any other point in S [10, 21].) Given a square P, the four squares sharing an edge with P are said to be *4-adjacent* to P; the eight squares sharing a vertex with P are said to be *8-adjacent* to P, and the four of them which are not 4-adjacent are called *strictly 8-adjacent*.

A 2D *binary digital image* I is a finite set of squares in the square grid. The squares in I are called *black* (object) squares. The squares in the complement I^c of I are called *white* (background). Connectedness is an equivalence relation obtained as the reflexive and transitive closure of adjacency. The classes of the image I with respect to α-connectedness are called connected α-components, for $\alpha \in \{4, 8\}$. Finite connected components of the complement of I are called *holes*. To maintain some similarity between the digital and continuous topology, the components and holes of I are defined with opposite types of adjacency.

A 2D image I has a *gap* at a vertex v if v is incident to two strictly 8-adjacent white squares (and two strictly 8-adjacent black ones) [6, 7, 12]. A gap-free image is called *well-composed* [18].

Depending on the chosen adjacency relation, two cell complexes can be naturally associated with a given image I. Recall that a cell complex Q is a collection of cells (homeomorphic images of the unit ball, that fit nicely together: the boundary of each cell and the intersection of any two cells (if nonempty) is composed of cells of lower dimension). In the plane, the underlying space $|Q|$ of a complex Q is the set of points in $\mathbb{R}^2$ that belong to some cell in Q.

For an image with 8-adjacency, the associated complex Q_8 is cubical and consists of the squares in I and all their edges and vertices. For an image with 4-adjacency, the associated polygonal complex Q_4 can be obtained from Q_8 by inserting a vertex at the center of each edge incident to a critical vertex, duplicating the critical vertices, and moving the two copies slightly towards the interior of the two incident black squares. The difference between the two complexes around a critical vertex is illustrated in Figure 1.

A formal definition of homotopy [15] is out of the scope of this paper. Intuitively, if one shape can be continuously deformed into the other then the two shapes are homotopy equivalent. Two 2D homotopy equivalent shapes have the same number of connected components and holes, with the same containment relations among them.

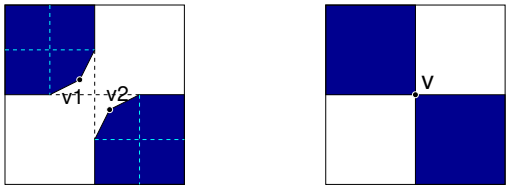


Figure 1: Complexes Q_4 (a) and Q_8 (b) in the neighborhood of a critical vertex v. The transformation in (a) affects only one quarter of each pixel incident to v.

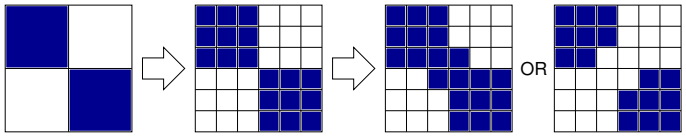


Figure 2: Repairing process according to Rosenfeld et al. [23]. Each square becomes a block of 3×3 squares, which become black or white according to 8-adjacency or 4-adjacency (last two images).

3 Related Work

Several repairing algorithms have been proposed in the literature, each with its features (regarding size, topology preservation, choice of adjacency and type of the output complex), as well as its benefits and drawbacks. We review briefly the repairing algorithm proposed for 2D images [23], and the 2D versions of the algorithms proposed for 3D images [13, 24, 25].

The method by Rosenfeld et al. [23] for 2D binary images uses a rescaling by factor 3 in both x and y directions of the square grid. In the rescaled grid, changing all black squares involved in a critical configuration to white squares, or vice versa, removes all critical configurations, without creating other ones (see Figure 2). The repaired image is homotopy equivalent to the initial image with the appropriate adjacency relation.

The randomized algorithm by Siqueira et al. [24] iteratively changes white squares to black ones, until a well-composed image is obtained. The grid resolution of the repaired image is the same as that of the input image, but there is no guarantee that the topology (homotopy) of the input image is preserved.

The algorithm by Gonzalez-Diaz et al. [13] creates a polygonal well-composed complex homotopy equivalent to the input image with 8-adjacency by increasing the grid resolution four times in each coordinate direction, thickening the neighborhood of critical vertices and subdividing it into polygons. The idea of the method is illustrated in Figure 3. Later [14], the rescaling factor is reduced to 2.

The algorithm by Stelldinger et al. [25] increases the grid resolution twice in both coordinate directions, by creating an additional square for each edge and each vertex in the grid. If 4-adjacency is considered for I, the squares corresponding to the edges and vertices in Q are black only if all the incident squares in I are black. If 8-adjacency is considered, then the squares corresponding to the edges and vertices in the associated complex Q (edges and vertices incident to some black square in I) are also black (see Figure 4).

The algorithms by Čomić and Magillo [8, 9] repair 3D images by passing from the cubic to the body centered cubic (BCC) or the face centered cubic (FCC) grid, respectively. These grids are defined by the Voronoi tessellation of $\mathbb{R}^3$ associated with the centers and the vertices, or the centers and the midpoints of the edges of each unit cube. The 2D repairing algorithm proposed here uses the same basic idea of passing from

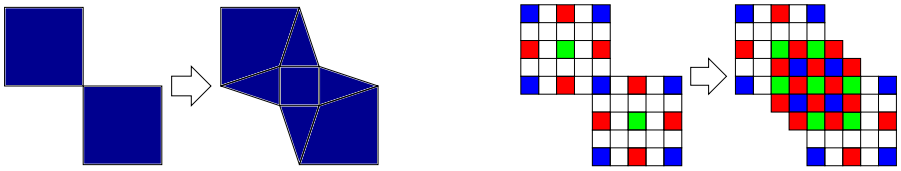


Figure 3: Repairing process according to Gonzalez-Diaz et al. [13]. The square complex becomes a polygonal complex, original squares become 2-cells of different shapes, depending on how many of their vertices are critical. (a) Geometry, (b) matrix representation, where blue, red and green squares correspond to 0-, 1- and 2-dimensional cells, as in [13].

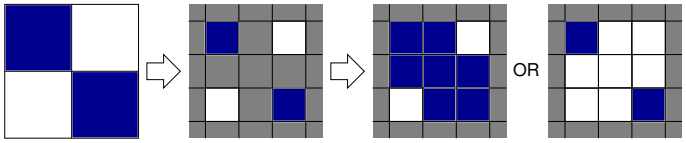


Figure 4: Repairing process according to Stelldinger et al. [25]. Each square, edge and vertex becomes a square, which becomes black or white according to 8-adjacency or 4-adjacency (last two images).

the square to an alternative (diamond) grid, but is necessarily different, due to specific properties of this grid.

4 Repairing Algorithm

We propose to pass from the square to the 2D diamond grid, by extending the set $\mathbb{Z}^2$ of square centers with the set $(\mathbb{Z}+1/2)^2$ of square vertices and tessellating the plane into Voronoi regions associated with the extended set. Each region is a rotated square (a diamond) with sides parallel to the lines $x \pm y = 0$. We call even diamonds the ones associated with square centers, and odd diamonds the ones associated with square vertices.

We create two well-composed images I_4 and I_8 in the diamond grid, homotopy equivalent to the image I with 4- and 8-adjacency, respectively. We include in the repaired images I_4 and I_8 all even diamonds corresponding to the black squares. We include also all odd diamonds corresponding to

1. the vertices incident to four black squares;
2. the vertices incident to three black and one white square;
3. the vertices incident to two vertically (horizontally) 4-adjacent black squares and two white ones, if the vertex is in the direction of the positive x axis (y axis) of the shared edge.

In the repaired image I_8, we include also the odd diamonds corresponding to

4. the critical vertices,

while we do not include those diamonds in the image I_4.

There are 16 different types of vertices in the square grid, depending on the configuration of the black and white incident squares. We show these configurations and the corresponding black diamonds in Figure 5.

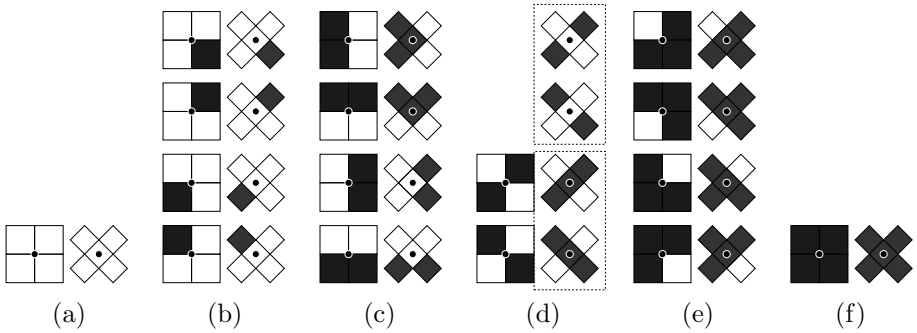


Figure 5: All possible configurations at a vertex in the square grid, and corresponding repaired configurations in the diamond grid. The four incident squares of the vertex can be: (a) all white, (b) one black and three white, (c) two black and two white, (d) three black and one white, (e) three black and one white, (f) all black. The diamond corresponding to the vertex is added in the two upper cases of (c), in (e), in (f) and, for the second version of the algorithm, in (d). Note that y-axis points downwards, as usual for images.

Figure 6 presents the pseudocode of the repairing algorithm. Figure 7 shows the effect of the two versions of the repairing algorithm on a sample image.

5 Well-Composedness and Homotopy Equivalence

We show that our repairing algorithm produces well-composed images homotopy equivalent to the given image I with the chosen adjacency relation.

5.1 Well-Composedness

Proposition 1 *The two images I_4 and I_8 produced by our repairing algorithm are well-composed.*

Proof: A critical vertex in the diamond grid is incident to exactly two black diamonds that are both even or both odd. Our rules prevent the creation of either type of criticality:

- For each pair of 8-adjacent black even diamonds, corresponding to a pair of 4-adjacent squares in I, the odd diamond (4-adjacent to both) in the conventional direction, corresponding to a vertex incident to both squares in I, is in I_4 and I_8, thus preventing the creation of critical vertices incident to two even black diamonds.
- For each pair of 8-adjacent black odd diamonds, corresponding to two adjacent vertices (incident to the same edge e) in the square grid, the two even diamonds, corresponding to the two squares incident in e, cannot be both white. Since a filled vertex has always at least two incident black squares, the only possible configuration would be an array of 2×3 or 3×2 squares, where the two central ones are white, and the remaining four are black. In this case, thanks to the choice of the conventional direction, rule 3 of our algorithm would not fill both vertices. □

```
Repair(Image Q, int a, Image D)
// Q is the input image, a is the adjacency type ∈ {4, 8},
// D is the output image (rotated 90 degrees), given as all white
1    for each black pixel (x, y) in Q
2    // (x, y) is a square, (x − y, x + y) the corresponding even diamond
3        set (x − y, x + y) as black in D
4    for each black pixel (x, y) in Q
5    // v will contain the odd diamonds corresponding to the vertices of the square (x, y)
6        v[0..3] = Get4Adj(x − y, x + y)
7        for i = 0..3 // for each vertex
8            if MustFill(v[i], D, a)
9                set v[i] as black in D

MustFill(int x, int y, Image D, int a)
// (x, y) is an odd diamond, a is the adjacency type ∈ {4, 8},
// D is the output image where black even pixels have been set,
// the function checks if (x, y) must be filled
1    p[0..3] = Get4Adj(x, y) // even diamonds
2    b = the number of black diamonds among p[0..3]
3    if (c ≤ 1) return false
4    if (c ≥ 3) return true
5    // c = 2, check configuration
6    if (a = 4) // 4-adjacency
7        return (v[0] is black) and (v[2] is white)
8    // 8-adjacency
9    return (v[0] is black) or ((v[1] is black) and (v[3] is black))

Get4Adj(int x, int y)
// (x, y) is a diamond, the function returns its four 4-adjacent diamonds
1    return [(x, y − 1), (x + 1, y), (x, y + 1), (x − 1, y)]
```

Figure 6: Pseudocode of the repairing algorithm and of auxiliary functions used in it.

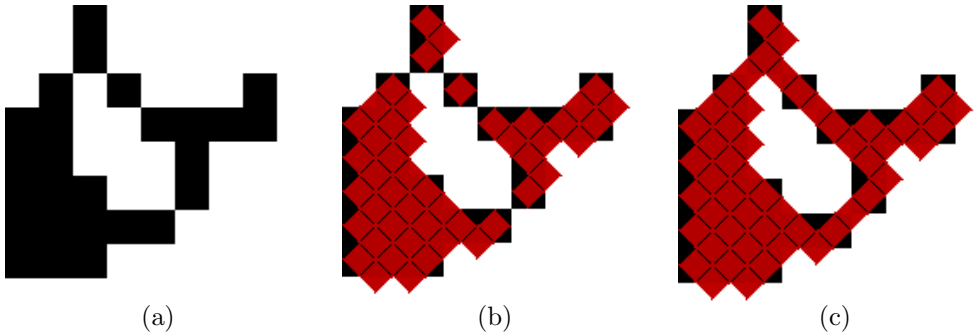


Figure 7: (a) An image I in the square grid and the two repaired images I_4 (b) and I_8 (c) in the rotated grid (red diamonds) with the two versions of the algorithm.

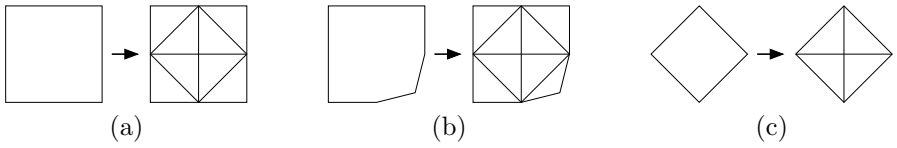


Figure 8: Triangulation of (a) squares and (b) polygonal cells (for $i = 4$) in Q_i, and (c) of diamonds in I_i.

5.2 Homotopy Equivalence

Proposition 2 *The spaces $|I_4|$ and $|I_8|$ are homotopy equivalent to the spaces $|Q_4|$ and $|Q_8|$, respectively.*

Proof: For $i \in \{4, 8\}$, we construct simplicial complexes ΣQ_i and ΣI_i that triangulate Q_i and I_i, respectively. Recall that a k-simplex is the convex hull of $k + 1$ affinely independent points. In 2D, simplexes are vertices, edges and triangles. A simplicial complex Σ is a finite set of simplexes, such that

- for each simplex in Σ, all its faces are in Σ and
- the intersection of two simplexes in Σ is either empty or composed of their common faces.

We show that $|Q_i|$ and $|I_i|$, $i \in \{4, 8\}$, are homotopy equivalent by constructing a sequence of collapses and expansions [27] that transform ΣQ_i to ΣI_i. In 2D, an elementary collapse removes from a simplicial complex Σ

- a triangle t and and a (free) edge e, if t is the only triangle incident to e, or
- an edge e and a (free) vertex v, if e is the only edge incident to v.

Expansion is inverse to collapse, it introduces into Σ a pair of simplexes such that one is a free face of another. Both operations preserve the homotopy type of $|\Sigma|$ [27].

We triangulate Q_i by introducing a vertex at the midpoint of each edge, inscribing the corresponding even diamond in each square C in Q_i (for $i = 4$, in the corresponding polygonal cell if some vertex of C is critical), and connecting the center c of each diamond to the four diamond vertices. We triangulate I_i by introducing the center c of each diamond and connecting c to the four diamond vertices. The triangulations ΣQ_i and ΣI_i are illustrated in Figure 8.

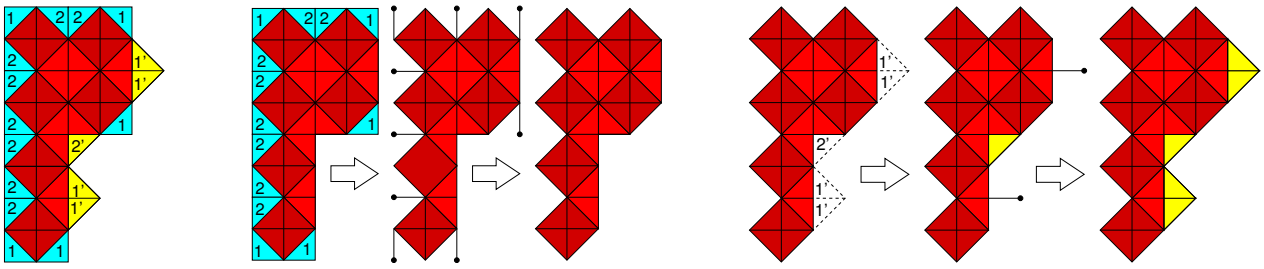


Figure 9: (a) Superposition of the triangulations ΣQ_i and ΣI_i for an example without critical configurations (there is no difference between $i = 4$ and $i = 8$). Red triangles belong to both triangulations, yellow triangles belong to ΣQ_i only, and cyan triangles belong to ΣI_i only. Yellow and cyan triangles are labeled with their type. (b) Triangles of ΣQ_i, which are not in ΣI_i, are removed through collapse. (c) Triangles of ΣI_i, which are not in ΣQ_i, are created through expansion.

The two triangulations ΣQ_i and ΣI_i coincide on $|Q_i| \cap |I_i|$. We will transform ΣQ_i into ΣI_i through a process that first collapses extra triangles of ΣQ_i, and then creates the missing triangles through expansion.

We collapse the triangles in ΣQ_i that are outside ΣI_i. They are contained in odd diamonds centered at the vertices in I incident to

1. exactly one black square,
2. exactly two edge-adjacent black squares and are not in the conventional direction, or
3. (for $i = 4$) exactly one of the two polygonal 2-cells corresponding to the two strictly vertex-adjacent black squares (these are the two copies of the critical vertices).

For each triangle t of type 1 or 3, we collapse t with one of its two free edges, and we collapse the remaining edge with its unique free vertex. Triangles of type 2 come in pairs. We collapse the two triangles, each with its unique free edge, and we collapse their shared edge with its unique free vertex. This stage is illustrated in Figures 9 (b) and 10 (b), (d).

We expand the triangles in ΣI_i outside of ΣQ_i. They are contained in odd diamonds centered at the vertices in I incident to

1'. exactly two edge-adjacent black squares and are in the conventional direction,
2'. exactly three black squares, or
3'. (for $i = 8$) exactly two strictly vertex-adjacent black squares (critical vertices).

We expand each triangle t of type 2' or 3' by adding t together with its unique free edge. Triangles of type 1' come in pairs. We expand first their shared edge together with its unique free vertex, and we expand the two triangles, each with its unique free edge. This stage is illustrated in Figures 9 (c) and 10 (d).

6 Experimental comparisons and results

We implemented our image repairing algorithm and, for comparison purposes, the algorithm in [25]. We have chosen this algorithm as our competitor because it is the one using the least additional memory among those

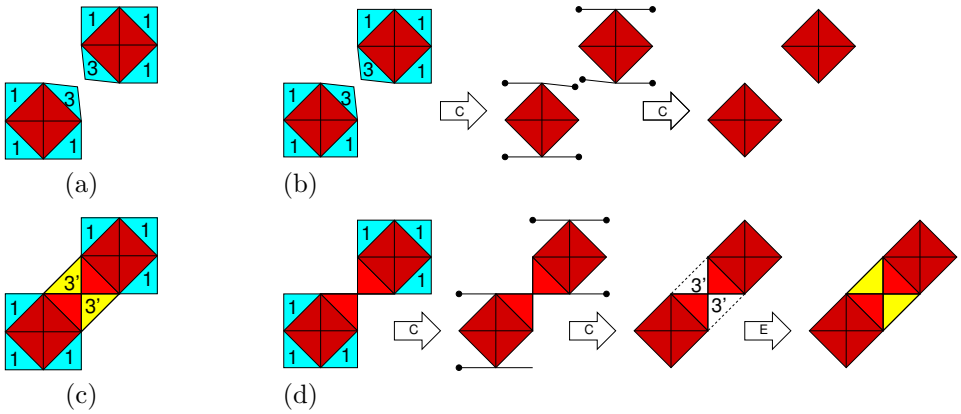


Figure 10: (a) Triangle types of ΣQ_4 and ΣI_4 for a critical configuration of 2×2 pixels. (b) Transformation of ΣQ_4 into ΣI_4. (c) Triangle types of ΣQ_8 and ΣI_8 for the same configuration. (d) Transformation of ΣQ_8 into ΣI_8. In (b) and (d), arrows with letters C and E denote collapsing of extra triangles and expansion to create missing triangles. In (b) expansion is not needed.

which are able to preserve image homotopy and which produce an image in the square grid. Both algorithms have been implemented in the two versions, i.e., producing a repaired image homotopy equivalent to the given one with both 4- and 8-adjacency.

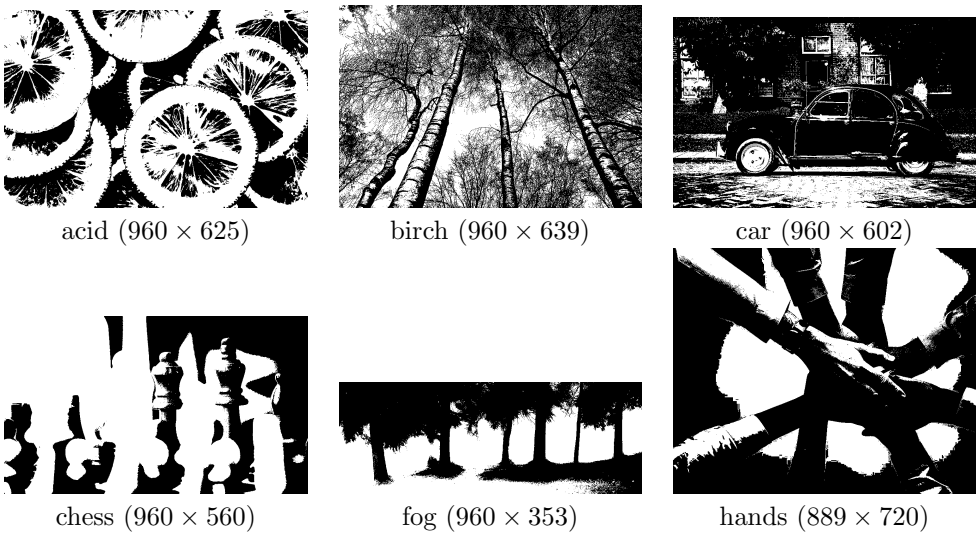
We tested the algorithms on several gray-scale images from the pixabay repository (https://pixabay.com/en/photos/grayscale after converting them to binary images by applying a threshold equal to 128 (where gray values are from 0 to 255). The used images are shown in Figure 11. In order to analyze a possible dependency of the results from image resolution, for each image we produced two lower resolutions by resizing it to 1/2 and 1/4 of the original size (in both x and y direction).

All algorithms are implemented in C language and executed on a PC equipped with an Intel CPU i7-2600K CPU at 3.4 Gigahertz with 32 Gigabytes of RAM.

Table 1 shows the sizes of the input images, the number of critical configurations in them, the sizes of the repaired images, and the execution times. As expected, our repaired image has half the size of the one produced by our competitor [25]. The time taken by our algorithm is from 67% to 75% that of the competitor algorithm in the case of repairing with 8-adjacency, and from 55% to 62% with 4-adjacency.

Figure 12 shows a repaired image. From the point of quality, the result produced by our competitor [25] is lighter than the original with 4-adjacency and darker with 8-adjacency. We analyze this behavior in detail on toy inputs in Figure 13. The input images have black and white lines of the same width. This property is maintained in the images repaired by our algorithm. The competitor algorithm [25], instead, shrinks black lines and expands white lines with 4-adjacency, and does the opposite with 8-adjacency. This may be a problem when preserving the area of black zones is important.

Getting back to Figure 12, the original image has 72964 black pixels. Our repaired images have 142434 and 150106 black pixels, and the pixel size is 1/2 of the original one. Thus, the areas of black zones in our repaired images are 71217 (97.6% of the original) and 75053 (102.9% of the original), respectively. The difference in areas with respect to the original image is below 3%. Images repaired with our competitor [25] have 198640 and 384452 black pixels, and the size of each pixel is 1/4 of the original one. Thus, the areas of black zones in the repaired images are 49660 (68.0% of the original) and 96113 (131.7% of the original), respectively. Here,



acid (960×625) birch (960×639) car (960×602)

chess (960×560) fog (960×353) hands (889×720)

Figure 11: Our binary versions of original images, with the size of bounding box.

the difference with respect to the original is more than 30%.

On the other side, our algorithm tends to smooth right angles, as we see in Figure 13 (b), and is not completely symmetric in the four cardinal directions. Another evident feature of our results is rotation. This is a problem when the image is seen by human users (where well-composedness is not relevant), while it does not affect computations made on images (where well-composedness may be important).

As the diamond grid is a (rotated) square grid, all classical image processing algorithms can be applied to the two repaired images. Moreover, some algorithms are simpler on well-composed images. Of course, execution time increases with image size. In the following, we analyze the impact of the size of the repaired image on the performance of image processing algorithms, since our repaired images are half in size, with respect to the ones repaired by our best competitor [25] (as we have just shown). As meaningful examples, we consider contour extraction and shrinking, i.e., a very simple and a rather complex task.

Finding the contours of an image means finding sequences of black border squares bounding each connected component and each hole of the image I. Connected components and holes can be considered with either 4- or 8-adjacency. For a well-composed image, the adjacency type makes no difference, and 4-connected contours can be extracted: each contour is a circular list of black squares where each square is 8-adjacent to at least one white square, and 4-adjacent to the previous square in the list. We implemented the extraction of 4-connected contours on well-composed images by means of the classical contour following approach [11, 20]. We have a current border square at each step, and we decide how to extend the contour based on the configuration (black or white) of the eight squares adjacent to it. We adopt a compressed representation of the contour, from [26], that is, we do not record all squares of the contour, but just the ones corresponding to the corners. This saves space and allows easy rescaling of the contour when the image is enlarged or shrunk.

Table 2 (a) compares the times for contour extraction from images repaired with our algorithms and with the competitor one [25]. As image sizes are halved, execution times on our repaired images are almost halved, with more gain for 8-adjacency. The ratio is from 0.44 to 0.55 for 8-adjacency and from 0.5 to 0.75 for 4-adjacency, and there is no specific trend with image resolution.

The aim of shrinking is to reduce each connected component C of an image I to a single pixel if C is

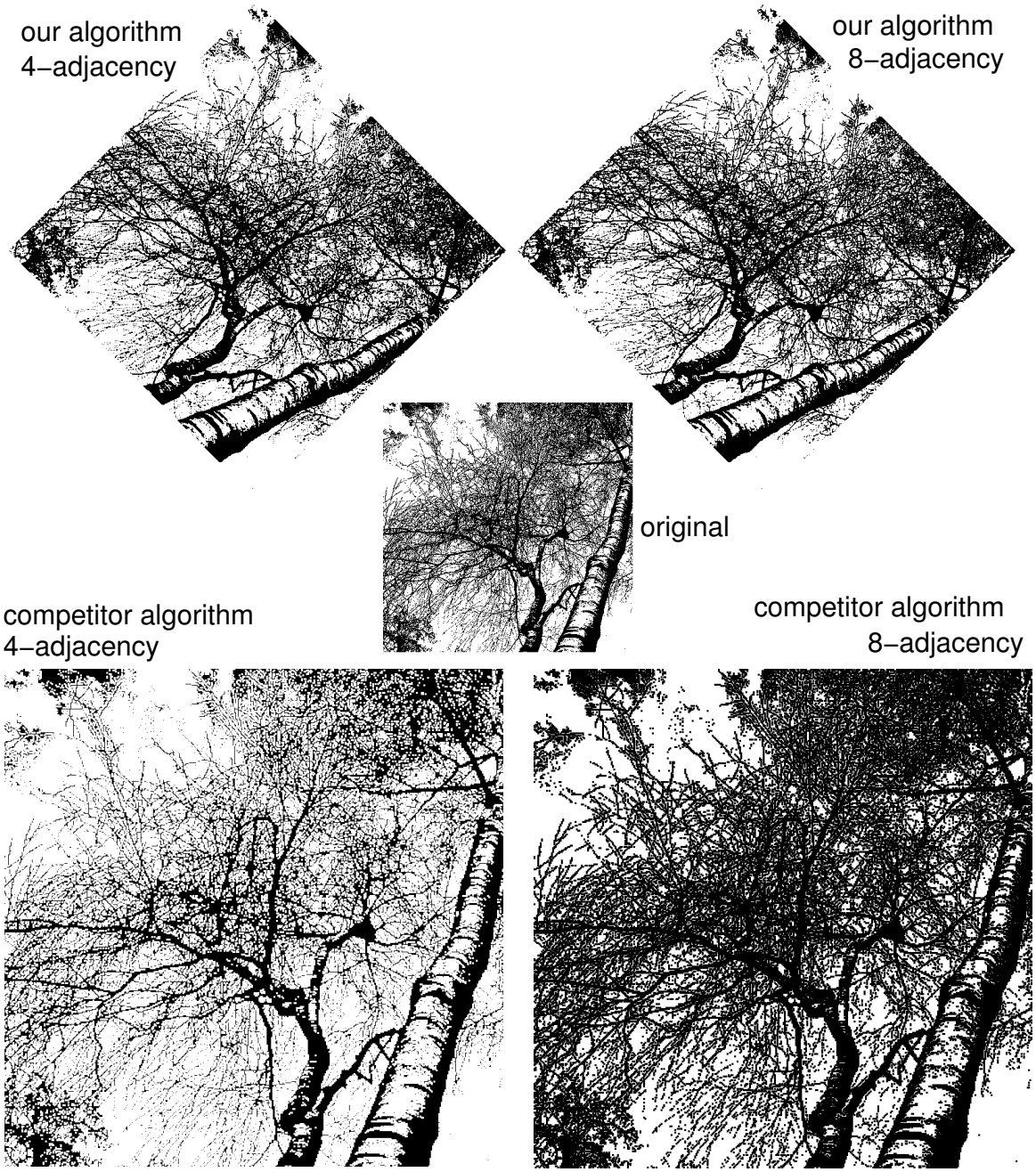
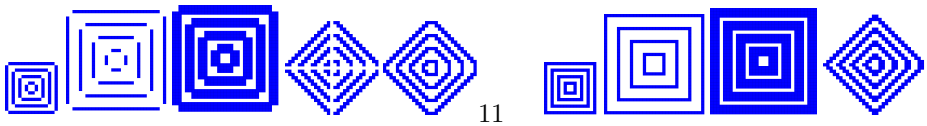


Figure 12: A portion of test image "birch" with its repaired versions.



(a) (b)

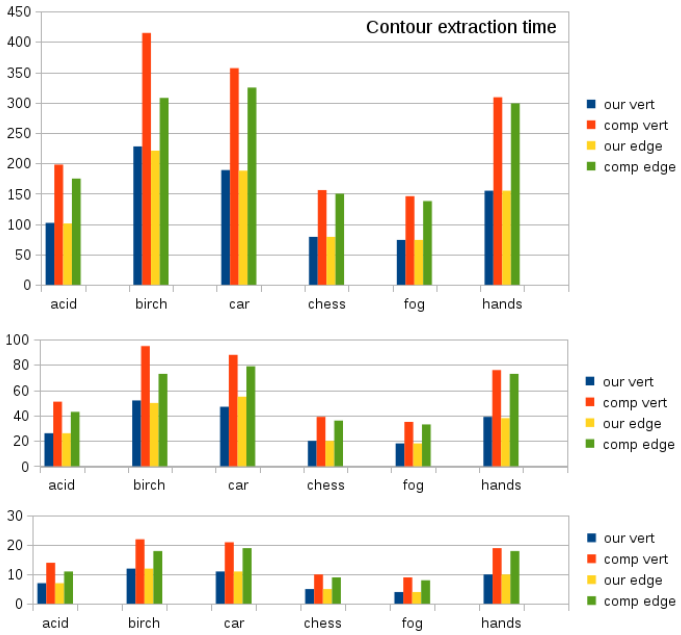
Figure 13: Two toy images and their repaired versions. From left to right: input image, output images by competitor algorithm [25] with 4- and 8-adjacency, output images by our algorithm with 4- and 8-adjacency. Image (b) is well-composed and our algorithm gives the same output with both adjacencies.

Table 1: The table shows: number of black squares (size) of input images and of output repaired images obtained by our method (our) and by the competitor one in [25] (comp.); number of critical vertices in the input images; execution times (in milliseconds); ratio between running times of our method and of the competitor one [25]. Images have been repaired according to 4- and 8-adjacency.

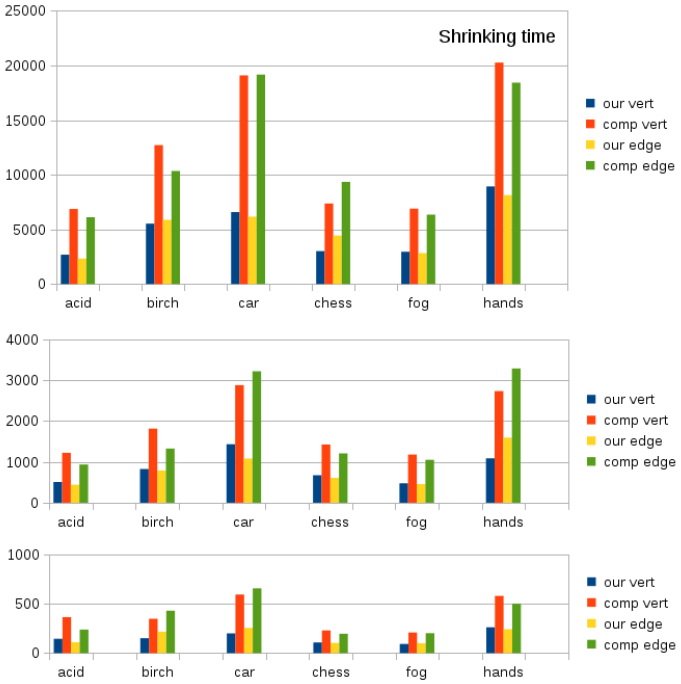
	input image		4-adjacency					8-adjacency				
image		critical	our		comp.[25]		time	our		comp.[25]		time
	size	vertices	size	time	size	time	%	size	time	size	time	%
acid 1	234K	1308	466K	122	870K	204	60	467K	122	999K	167	73
acid 1/2	58K	387	115K	31	208K	51	61	115K	31	253K	42	74
acid 1/4	14K	218	28K	7	48K	13	54	28K	7	65K	10	70
birch 1	340K	22K	678K	206	1038K	330	62	700K	203	1665K	274	74
birch 1/2	88K	4485	176K	51	288K	83	60	181K	50	414K	68	74
birch 1/4	23K	574	46K	12	79K	21	57	47K	12	105K	17	71
car 1	448K	2239	896K	230	1697K	386	59	898K	229	1880K	316	72
car 1/2	112K	540	225K	57	424K	97	59	225K	57	473K	79	72
car 1/4	28K	103	56K	14	106K	24	58	56K	14	119K	19	74
chess 1	213K	180	425K	105	834K	179	59	426K	105	867K	146	72
chess 1/2	53K	29	106K	26	205K	45	58	106K	26	219K	36	72
chess 1/4	13K	6	26K	6	50K	11	55	26K	6	56K	9	67
fog 1	196K	700	391K	98	758K	166	58	392K	97	805K	135	72
fog 1/2	49K	146	98K	24	190K	41	59	98K	24	202K	34	71
fog 1/4	12K	22	25K	6	47K	10	60	25K	6	51K	8	67
hands 1	422K	1030	845K	209	1659K	356	59	846K	209	1720K	290	72
hands 1/2	106K	280	212K	52	414K	89	58	212K	52	433K	73	71
hands 1/4	27K	33	53K	13	103K	22	59	53K	13	110K	18	72

homotopy equivalent to a disk, to a simple closed chain of pixels if C is homotopy equivalent to a circle, etc. This is commonly achieved by iteratively removing squares from the image, i.e., changing their status from black to white. Squares can only be removed if their removal does not change either the number of components or the number of holes according to the chosen (4- or 8-) adjacency. The process is iterated until no more squares may be removed. We consider shrinking of a well-composed image, with 4-adjacency. The decision whether a square is removable or not only depends on the status of the eight squares in its neighborhood (see [16, 22] for details and algorithms). Squares with disjoint neighborhoods do not affect the removability of each other. In the method we implemented, black squares are classified into four subsets according to the parity of their (x, y) coordinates. The algorithm performs a cycle on the four subsets, examining one of them at a time. Removable squares of the current subset do not interfere and are removed together. Table 2 (b) compares the times for shrinking images repaired with our algorithms and with the competitor one [25]. Here running time also depends heavily on image characteristics, besides input size, but we can see that execution times are roughly halved also in this case. Ratios are a bit smaller than for contour extraction, ranging from 0.34 to 0.5 with 8-adjacency and from 0.32 to 0.6 with 4-adjacency, and present no specific trend with resolution.

Table 2: Running times for (a) contour extraction and (b) shrinking, applied on images repaired by our approach and by the competitor one [25] (in milliseconds). From top to bottom: full resolution, scaled 1/2, and scaled 1/4.



(a)



(b)

7 Summary

We have proposed a simple method to transform a given 2D binary image into two homotopy equivalent well-composed images (depending on the chosen adjacency) by using the 2D diamond grid. The resulting images are just double in size with respect to the original one, and this improves the best existing homotopy preserving method [25], where the image size is doubled in both coordinate directions (leading to a factor 4). The diamond grid is a rotated square grid, thus all known image processing algorithms can be applied to our repaired images, and their smaller size, with respect to the state-of-the-art of homotopy equivalent image repairing, saves processing time.

8 Acknowledgments

This work has been partially supported by the Ministry of Education and Science of the Republic of Serbia within the Project No. 34014.

References

[1] N. Boutry, T. Géraud, and L. Najman. How to make nD images well-composed without interpolation. In *2015 IEEE International Conference on Image Processing, ICIP 2015*, pages 2149–2153, 2015.

[2] N. Boutry, T. Géraud, and L. Najman. A Tutorial on Well-Composedness. *Journal of Mathematical Imaging and Vision*, 60(3):443–478, 2018.

[3] N. Boutry, T. Géraud, and L. Najman. How to Make n-D Plain Maps Defined on Discrete Surfaces Alexandrov-Well-Composed in a Self-Dual Way. *Journal of Mathematical Imaging and Vision*, 2019.

[4] N. Boutry, R. González-Díaz, and M. J. Jiménez. One More Step Towards Well-Composedness of Cell Complexes over nD Pictures. In *Discrete Geometry for Computer Imagery - 21st IAPR International Conference, DGCI*, pages 101–114, 2019.

[5] N. Boutry, R. González-Díaz, and M.-J.. Jiménez. Weakly well-composed cell complexes over nD pictures. *Information Sciences*, 499:62–83, 2019.

[6] 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.

[7] 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.

[8] L. Čomić and P. Magillo. Repairing 3D binary images using the BCC grid with a 4-valued combinatorial coordinate system. *Information Sciences*, 499:47–61, 2019.

[9] L. Čomić and P. Magillo. Repairing 3D Binary Images Using the FCC Grid. *Journal of Mathematical Imaging and Vision*, 61(9):1301–1321, 2019.

[10] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[11] S. J. E. Gose, R. Johnsonbaugh. *Pattern Recognition and Image Analysis*. Prentice-Hall, Inc., 1996.

[12] J. Françon, J. Schramm, and M. Tajine. Recognizing arithmetic straight lines and planes. In *Discrete Geometry for Computer Imagery, 6th International Workshop, DCGI*, pages 141–150, 1996.

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

[14] R. González-Díaz, M.-J.. Jiménez, and B. Medrano. Efficiently Storing Well-Composed Polyhedral Complexes Computed Over 3D Binary Images. *Journal of Mathematical Imaging and Vision*, 59(1):106–122, 2017.

[15] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

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

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

[18] L. J. Latecki. 3D Well-Composed Pictures. *CVGIP: Graphical Model and Image Processing*, 59(3):164–172, 1997.

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

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

[21] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[22] K. Preston Jr. and M. J. B. Duff. *Modern Cellular Automata*. Advanced Applications in Pattern Recognition. Springer US, 1984.

[23] 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.

[24] M. Siqueira, L. J. Latecki, N. J. Tustison, J. H. Gallier, and J. C. Gee. Topological Repairing of 3D Digital Images. *Journal of Mathematical Imaging and Vision*, 30(3):249–274, 2008.

[25] 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.

[26] M. E. T. Miyatake, H. Matsushima. Contour representation of binary images using run-type direction codes. *Machine Vision and Applications*, 9:193–200, 1997.

[27] J. H. C. Whitehead. Simplical spaces, nuclei and m-groups. *Proceedings of the London Mathematical Society*, 45:243–327, 1938.