Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Computing the minimal perimeter polygon for sets of rectangular tiles based on visibility cones

Petra Wiederhold

pwiederhold@gmail.com

Centro de Investigación y de Estudios Avanzados (CINVESTAV-IPN)

Research Article

Keywords: minimum perimeter polygon, regular complex in rectangular mosaic, rectangular tiling, boundary tracing, relative convex hull

Posted Date: October 10th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3412621/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Journal of Mathematical Imaging and Vision on June 24th, 2024. See the published version at https://doi.org/10.1007/s10851-024-01203-z.

Computing the minimal perimeter polygon for sets of rectangular tiles based on visibility cones

Petra Wiederhold

Received: date / Accepted: date

Abstract In the context of digital image modelling and convexity analysis of digital objects, the minimum perimeter polygon (MPP) was defined in the 1970s in several articles by Sklansky, Chazin, Hansen, and Kibler, where sets of pixels were identified with plane mosaics or polygonal tilings, and, the Sklansky-Chazin-Hansen algorithm (1972) and the Sklansky-Kibler algorithm (1976) were proposed for determining the MPP vertices. Both algorithms rely on constructing and iteratively restricting visibility cones, the MPP vertices result as special vertices of the tiles. This paper reviews both classical algorithms for regular complexes which are special sets of rectangular tiles, and an adaptation to square tiles recommended in widely used modern digital image analysis text books (2018, 2020), to construct approximations of simple digital 4-contours. We show that these three algorithms are erroneous, specially, that the Sklansky-Chazin-Hansen algorithm has numerous types of errors, and that their mathematical foundation contains incorrect details. After analyzing and correcting these errors, a new version of MPP algorithm for certain sets of rectangular tiles is presented, its correctness is proved, and the classical algorithms are corrected.

Keywords minimum perimeter polygon · regular complex in rectangular mosaic · rectangular tiling · boundary tracing · relative convex hull

Mathematics Subject Classification (2010) 52-08 · 68R99 · 68U05 · 68U10

Petra Wiederhold
Departamento de Control Automático, Centro de Investigación y de Estudios Avanzados (CINVESTAV-IPN), Av. I.P.N. 2508, San Pedro Zacatenco, México 07360, CDMX
E-mail: pwiederhold@gmail.com

1 Introduction

Motivated by digital image modelling, the *minimum perimeter polygon (MPP)* was defined in the 1970s in several articles by Sklansky, Chazin, Hansen, and Kibler [9,18–21], where pixels were identified with tiles in plane mosaics or tilings. The MPP has important applications in digital image analysis for object representation and approximation. The MPP faithfully describes convexity and concavity properties of the object [2,10,12,13,16,25]. Moreover, the MPP perimeter was proved to be a multigrid convergent perimeter estimator for a compact simply connected subset S of the Euclidean plane whose frontier is a smooth Jordan curve [22,24]. In this regard, the MPP frontier is the minimal length Jordan curve which circumscribes the Inner Jordan digitization of S and lies within the Outer Jordan digitization of S [12,22,24]. The MPP concept in [22, 24] corresponds to digitizations based on square pixels, the difference set between Outer and Inner Jordan digitizations is a polyomino supposed as simple grid-continuum, that is, corresponding to a simple digital 4-curve. For that special case, the MPP coincides with the MPP due to Sklansky et al. [9,18–21] which, at its origin, is more general and uses a distinct digitization.

In [20,21] the first two algorithms for determining the MPP vertices for digital objects represented by sets of polygonal tiles, were proposed. They are based on constructing and iteratively restricting cones of visibility, all MPP vertices result as vertices of the tiles. Under certain assumptions, the MPP of a set of tiles, called complex, coincides with the relative convex hull of certain subset of vertices of these tiles, with respect to the complex body [21]. The algorithms from [20,21] are many cited in the literature, including in widely used modern textbooks such as [4,6], where applica-

tions and adaptations to simple 4-contours, also called digital Jordan curves, are recommended to approximate objects in the digital plane $(c\mathbb{Z})^2$ [11,10,12,13, 15]. The present paper analyses the Sklansky-Chazin-Hansen MPP algorithm (1972) [20] and the Sklansky-Kibler MPP algorithm (1976) [21] under the same supposition of regular complexes in rectangular mosaics, although the Sklansky-Kibler algorithm originally was proposed for more general complexes and mosaics. We also review an adaptation of these classic algorithms to special complexes of square tiles, which is proposed in modern textbooks [4,6] (2018, 2020). We analyse these three algorithms and show that they all are erroneous. We develop a corrected version of MPP algorithm for regular complexes in rectangular mosaics, prove its correctness, and show how the classical algorithms may be corrected.

We also show that the mathematical foundation presented in [20,21] has incorrect details, in particular, the input data need to satisfy stricter requirements to make the algorithms applicable. The MPP algorithm we propose, applies to a special kind of boundary chain which can be obtained from any regular complex by boundary tracing. Our paper offers a general boundary tracing algorithm for complexes in rectangular mosaics, and deduces properties of so-called generic boundary chains.

Section 2 of this article summarizes mathematical preliminaries. Section 3 analyses the MPP algorithm from the textbooks [4,6,12] for simple digital 4-contours. Section 4 presents theoretical backgrounds of the MPP for complexes on polygonal mosaics due to [20,21]. The Sklansky-Chazin-Hansen MPP Algorithm from [20] is studied in detail in Section 5, the Sklansky-Kibler MPP Algorithm from [21] in Section 6. Section 7 explains a restriction to boundary chains generated by boundary tracing, and presents an algorithm for this. Section 8 presents a corrected version of MPP algorithm which joins the correct ideas of all algorithms studied before in this paper, but avoids their errors. The new algorithm is illustrated by examples, its correctness is proved, and its adaptation to correct both classical algorithms from [20,21] is shown. Some Conclusions complete the paper.

2 Preliminaries

We denote by $\mathbb{Z}$ the set of integers, $\mathbb{R}^2$ is the Euclidean plane with standard topology where cl, int, fr denote, resp., the topological closure, interior, and frontier. For $p, q \in \mathbb{R}^2$, $\overline{pq}$ is the straight line segment joining them. If $p \neq q$, $\overrightarrow{pq}$ is the directed straight line segment from p to q. All formulae and definitions are based on the standard right-hand Cartesian coordinate system in $\mathbb{R}^2$.

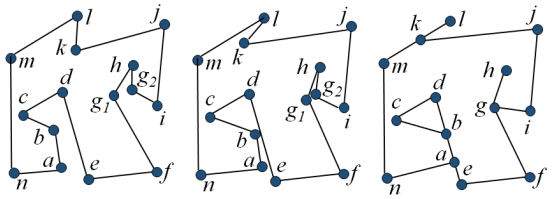

Fig. 1 From the left to the right: instances of a sequence of simple polygons which converges to a weakly simple polygon with cyclic vertex sequence $(m, n, a, b, c, d, e, f, g, h, g, i, j, k, l, k)$.

By a ***polygonal curve*** we mean a closed curve $\gamma = f([0,1])$ in $\mathbb{R}^2$, with a continues function f, where $f(0) = f(1)$ and there exist $n \in \mathbb{N}$ and $t_1, t_2, \cdots, t_n$ with $t_0 = 0 < t_1 < \cdots < t_n = 1$ such that all $f([t_{i-1}, t_i])$, $i \in \{1, 2, \cdots, n\}$, are straight line segments. Then γ has a finite *length* given as the sum of lengths of all its straight line segments, and a point $f(t_i) \in \gamma$ $(1 \leq i \leq n-1)$ is a ***vertex*** of γ if $f(t_i) \notin \overline{f(t_{i-1})f(t_{i+1})}$. A closed curve $\gamma = f([0,1])$ is *simple* or a *Jordan curve* if f is injective on $[0,1)$. A ***simple polygon*** is a compact connected subset of $\mathbb{R}^2$ whose frontier is a Jordan polygonal curve, such polygon has no holes, is two-dimensional, and has at least three vertices.

A *weakly simple polygon* is enclosed by a polygonal curve which may be not simple, it may touch itself or trace back on itself, but never does transversely cross itself [26,1]. Such a curve γ is obtained in [20,21] as the limit of a sequence of Jordan curves γ_j where for a finite number of curve points of γ, each such point is the limit of a sequence of corresponding vertices of the curves γ_j, see Figure 1. The region enclosed by γ is the limit of the sequence of simple polygons enclosed by the curves γ_j, hence, there is a Jordan curve arbitrarily near to γ:

Definition 1 (based on [1,20,21]) A ***weakly simple polygonal curve*** is a polygonal curve γ determined by $k \geq 3$ vertices $p_1, p_2, \cdots, p_k$, where for any $\epsilon > 0$, there exists a simple polygonal curve δ determined by k vertices $q_1, q_2, \cdots, q_k$ such that for each $j \in \{1, 2, \cdots, k\}$, $d(p_j, q_j) < \epsilon$, being d the Euclidean distance. Then the polygon P enclosed by γ is called ***weakly simple polygon***. Any vertex of γ is called ***vertex*** of P. Additionally, any single point p is a weakly simple polygonal curve having the unique vertex p, and, for distinct $p, q \in \mathbb{R}^2$, the straight line $\overline{pq}$ is a weakly simple polygonal curve with the cyclic vertex sequence (p, q).

Under a ***polygon*** we understand any compact subset of $\mathbb{R}^2$ which is a weakly simple polygon, clearly any simple polygon is weakly simple. We assume that Jordan curves and weakly simple polygonal curves always are traced in counterclockwise orientation, in concor-

dance with [10,12,13,20,21]. Fixed the tracing orientation, the polygon enclosed can be uniquely represented as cyclic sequence of its vertices $(p_1, p_2, \cdots, p_k)$, where $p_i \notin \overline{p_{(i-1 \bmod k)} p_{(i+1 \bmod k)}}$ for all $i \in \{1, \cdots, k\}$. For a simple polygon, the vertices in this sequence are pairwise distinct, this is not guaranteed for a weakly simple polygon, see Figure 1. If a polygon P is given by its cyclic sequence $(p_1, p_2, \cdots, p_k)$ of vertices, its ***perimeter*** is the sum of lengths of the straight line segments $\overline{p_i p_{i+1}}$, $1 \leq i \leq k-1$, and $\overline{p_k p_1}$. Denoting by d the Euclidean distance,

$$perimeter(P) = \sum_{i=1}^{i=k-1} d(p_i, p_{i+1}) \; + \; d(p_k, p_1).$$

The polygon $P = \overline{pq}$ has perimeter $2 \cdot d(p, q)$, a polygon given as a single point has perimeter zero.

We use the orientation of a triple of points $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3) \in \mathbb{R}^2$ given by the determinant

$$D(p_1, p_2, p_3) = \det \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}.$$

Any right-hand Cartesian coordinate system satisfies that $D(p_1, p_2, p_3) < 0$ if and only if p_3 lies on the right of $\overrightarrow{p_1 p_2}$, (p_1, p_2, p_3) forms a *right turn*. $D(p_1, p_2, p_3) > 0$ is equivalent to p_3 lying on the left of $\overrightarrow{p_1 p_2}$, (p_1, p_2, p_3) forms a *left turn*. $D(p_1, p_2, p_3) = 0$ is equivalent to collinearity, then p_1, p_2, p_3 belong to the same straight line segment which may be degenerated to a point. For consecutive points p_1, p_2, p_3 in a finite cyclic sequence of curve points of a polygonal curve γ, due to counterclockwise tracing, p_2 is called ***convex*** if $D(p_1, p_2, p_3) > 0$ (left turn), and ***concave*** if $D(p_1, p_2, p_3) < 0$ (right turn), *linear point* if $p_2 \in \overline{p_1 p_3}$, and a *peak* if $D(p_1, p_2, p_3) = 0$ but $p_2 \notin \overline{p_1 p_3}$. A simple polygonal curve has no peaks. For any weakly simple polygonal curve γ counterclockwisely traced, the polygon P enclosed lies on the left side of γ. This justifies to consider the *internal angle* at any vertex p_i, measured in mathematically positive sense between $\overline{p_{i-1} p_i}$ and $\overline{p_i p_{i+1}}$. The internal angle at any convex vertex is strictly between 0° and 180°, and between 180° and 360° at any concave vertex.

3 An MPP algorithm for square tiles and simple digital boundaries proposed in modern textbooks

The books "Digital Image Processing" [4] (4th edition 2018) and "Digital Image Processing using Matlab" [6] (3rd edition 2020), belong to the widely used textbooks and references on digital image processing and analysis. Part of Section 11.2 of [4], which appears also as part of Section 13.3 of [6], suggests the minimum perimeter polygon (MPP) to represent and approximate digital boundaries in 2D digital images[1]. The MPP algorithm in [4,6] is proposed for simple (edge-adjacency) paths of squares corresponding to pixels in the discrete plane $(c\mathbb{Z})^2$, in particular, for simple 4-contours being a sort of digital Jordan curves. Such a *simple path* is a cyclic sequence of square tiles where each two consecutive tiles share a side, and each tile shares some side with exactly two other tiles in the path, see our Section 4. Those paths informally are called "simply connected cellular complexes" in [4,6] where the authors cite the articles [20,24] and the Digital Geometry textbook [12].

In [20], the MPP is defined for complexes in polygonal tilings, see our Section 4, and the Sklansky-Chazin-Hansen algorithm to determine the MPP for certain complexes in rectangular tilings is proposed. This algorithm will be analysed in our Section 5, its application to simple edge-adjacency paths of square tiles corresponds to a very special case. The algorithm of [4,6] coincides, except the notation, with the algorithm from the book [12], which originally was published in [11,13] where the authors affirm to base their work on [24].

Due to the approach of [24], see also [22,23], the MPP is defined for simple polygons A, B in the plane, where A lies in the topological interior of B. The MPP is the *relative convex hull* of A with respect to B, also called *geodesic hull*, the MPP frontier is the shortest length Jordan curve which circumscribes A and does not leave B. To develop multigrid convergent perimeter estimators, the authors of [24] apply the MPP to simple 4-curves presenting digital boundaries: identifying the discrete plane $(c\mathbb{Z})^2$ with a square tiling, a bounded simply connected region $S \subset \mathbb{R}^2$ can be digitized to obtain its Inner Jordan digitization $J^-(S)$ (the set of all tiles completely contained in S), and Outer Jordan digitization $J^+(S)$ (the set of all tiles which intersect S). If S is bounded by a smooth Jordan curve, for sufficiently large digitization resolution c^{-1} , the 4-boundary of $J^+(S)$ given as $F = J^+(S) \setminus J^-(S)$, forms a simple 4-curve, also called simple 4-contour. For this special case, the MPP frontier is the minimal length Jordan curve which circumscribes $J^-(S)$ but is confined by the union of tiles of $J^+(S)$, hence the curve lies in F. The article [24] proposes some algorithms, but it does not contain any MPP algorithm strictly related to the algorithms from [4,6,12,20].

[1] The sections on the MPP algorithm and its foundation appear with the same content in the 3rd ed. (2007) of "Digital Image Processing" [3], and in the 2nd ed. (2009/2010) of "Digital Image Processing using Matlab" [5,7].

The union of tiles belonging to $F = J^+(S) \setminus J^-(S)$ forms an annular subset of $\mathbb{R}^2$ which has an inner frontier and an outer frontier, called *inner wall* and *outer wall* in [4,6], or *inner track* and *outer track* in [12], see the example in Figure 2. The MPP algorithm from [4, 6, 12] relies on the following:

– The uppermost-leftmost vertex of the inner wall is a convex vertex of the MPP [20].

– There is a bijective correspondence between the concave vertices of the inner wall and the concave vertices of the outer wall, called *mirrored concave vertices* in [4, 6], which were first used in [11–13]. This fact is only true if F forms a simple 4-contour which is a type of digital Jordan curve [12].

– Each convex vertex of the MPP is a convex vertex of the inner wall, and each concave vertex of the MPP is a mirrored concave vertex [20, 24].

As consequence, the MPP vertices can be found based on tracing the inner wall of F, starting with its uppermost-leftmost vertex V_0 which provides the first (convex) MPP vertex. The algorithm in [4,6] traces the inner wall in counterclockwise sense and analyses each convex vertex and each mirrored concave vertex. Only those vertices are MPP vertex candidates, they are not at the original grid point positions: for pixels in $\mathbb{Z}^2$, these candidates and hence all MPP vertices have coordinates of type $z + \frac{1}{2}, z \in \mathbb{Z}$. The algorithm uses two so-called crawler points (named beetles in [11,13,12]): W crawls along convex vertices, B along mirrored concave vertices, see Algorithm 1.

As stated in [4,6], "the algorithm stops when it reaches the first vertex, again". This is included in Algorithm 1 in Step 4: if there is no more element in L, tracing L as cyclic list would find V_0 as next element. Fixed the last found MPP vertex V_L, the crawlers define a cone of visibility through the forthcoming tiles of the input chain. The cone is expected to become narrower while the crawlers are updated, the last found convex vertex W determines the left cone border, the last found mirrored concave vertex B determines its right border. Hence, the cases of Step 5 mean the following:

(a) If V lies strictly on the left outside the current cone, W is found as next (convex) MPP vertex.

(b) If V lies inside the cone, or on its borders, then the convex/concave crawler, resp., is updated if V is convex/concave. As updating result, one of the border cones is restricted, or, the cone is confirmed.

(c) If V lies strictly on the right outside the current cone, B is found as next (concave) MPP vertex.

Our Examples 1 and 2 show that these actions may generate erroneous MPP vertices. The articles [11,13] affirm that the correctness of the actions in cases (a) and (c) was proven in [24]. We found it difficult to locate that proof in the extensive paper [24] which develops theoretical bases for shortest paths problems in $\mathbb{R}^2$, and mentions ideas for MPP algorithms based on iterative constructions of convex hulls and polygon partitions, in particular, into visibility polygons. The work [11,13] does not cite the much earlier papers [20,21] but the strategies used in all these algorithms are similar.

Algorithm 1 Algorithm due to [4,6] to determine the MPP of a complex of square tiles.

Input: A simple 4-contour in the digital plane $(c\mathbb{Z})^2$, that is, a simple edge-adjacency path of tiles forming the boundary chain $\beta(\mathcal{C})$, corresponding to counterclockwise tracing, of a complex $\mathcal{C}$ of square tiles.

Output: List *MPP* of MPP vertices.

1: Determine a convex MPP vertex V_0 as the uppermost--leftmost vertex of the inner wall of $\beta(\mathcal{C})$.
Set $V_L = V_0$, add V_L to the list *MPP*.

2: Starting with V_0, trace $\beta(\mathcal{C})$ to construct the candidate list L: add V_0 as first element of L, each convex vertex found on the inner wall is added to L. For each concave vertex found on the inner wall, its mirrored concave vertex is added to L.

3: Set $W = V_L$ (crawler along convex vertices).
Set $B = V_L$ (crawler along concave vertices).
Situate on the last found MPP vertex V_L in the list L.

4: The algorithm stops if there is no more element in L.
Otherwise, read the next element V from the list L.

5: **(a)** If $D(V_L, W, V) > 0$ then set $V_L = W$ and add V_L to *MPP* (new convex MPP vertex found).
Then go to Step 3.
(b) If $D(V_L, W, V) \leq 0$ and $D(V_L, B, V) \geq 0$ then update a crawler: if V is convex then set $W = V$, if V is concave then set $B = V$. Then go to Step 4.
(c) If $D(V_L, W, V) < 0$ then set $V_L = B$ and add V_L to *MPP* (new concave MPP vertex found).
Then go to Step 3.

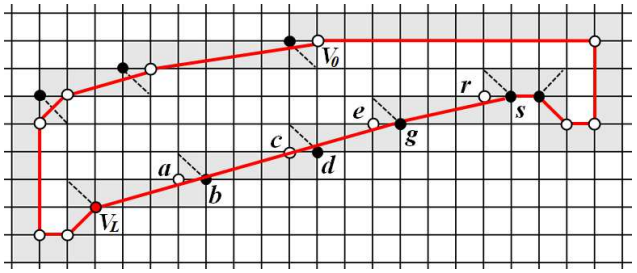

Fig. 2 A complex with its simple 4-contour shaded, white discs present convex vertices of the inner wall, mirrored concave vertices (belonging to the outer wall) are drawn as black discs, broken black lines show their bijection to the concave vertices of the inner wall. The MPP frontier is drawn in red, g is the unique MPP vertex between V_L and s.

Example 1: We apply Algorithm 1 from [4,6] to the complex of Figure 2. Starting from the MPP vertex V_L, the next candidates are given as a, b, c, d, e, g, r, s. The next MPP vertices after V_L are given as g and s, g is needed since $\overrightarrow{V_L s}$ has slope 4/15, but $\overrightarrow{V_L g}$ has the larger slope 3/11, $\overrightarrow{V_L s}$ does not fit in the boundary.

The algorithm obtains an initial cone with left border determined by $W = a$ ($\overrightarrow{V_L a}$ has slope 1/3) and right border by $B = b$, $\overrightarrow{V_L b}$ and $\overrightarrow{V_L d}$ have slope 1/4. The next candidate $W = c$ restricts the left border, $\overrightarrow{V_L c}$ has slope 2/7, $B = d$ confirms the right border. The slope of $\overrightarrow{V_L e}$ is 3/10 which exceeds 2/7, hence e lies on the left outside the cone. Step 5(a) implies that a new polygon vertex is found as $W = c$, which is a *false result*. The algorithm finds c as additional vertex which makes the perimeter of the polygon larger than that of the MPP.

Example 2: We apply Algorithm 1 from [4,6] to the complexes of Figure 3, consider the point 1 as the last found MPP vertex. For the smallest complex (d), point 6 is the unique MPP vertex between the MPP vertices 1 and 8, the polygonal curve 1-6-8 has length $\sqrt{3^2+4^2}+\sqrt{2} \approx 6.41$. The algorithm initializes a cone with the ray $\overrightarrow{1,2}$ (W=2) as left border and the ray $\overrightarrow{1,3}$ (B=3) as right border. Then, W=4 confirms the left border, B=5 restricts the right border (slope 2/3), finally W=6 restricts the left border (slope 3/4), the cone is well-defined since $2/3 < 3/4$. Point 7 lies on the right outside the cone, the ray $\overrightarrow{1,7}$ has slope $3/5 < 2/3$. As consequence, point 5 is found as MPP vertex which is a *false result*. It is easy to see that point 8 is found as next MPP vertex, the curve 1-5-8 has length $\sqrt{3^2+2^2}+2\sqrt{2} \approx 6.43$, which makes the perimeter of the polygon larger than that of the MPP.

Figures 3(a,b,c) are generated by repeating the pattern of the sequence 2-3-4-5, before reaching the same situation as for 6-7-8 in (d). The sequence 2-3-4-5 appears four times in (a), and the pattern starting at point 1 is the same as those starting at points 5,9,13. The MPP for (a) has the unique vertex 18 between the vertices 1 and 20, the curve 1-18-20 has length $\sqrt{9^2+13^2}+\sqrt{2} \approx 17.225$. The algorithm finds points 5,9,13,17 as erroneous MPP vertices, the curve 1-5-9-13-17-20 has length $4\sqrt{13}+2\sqrt{2} \approx 17.251$. Similarly, instead of the MPP vertex 14, points 5,9,13 are found for (b), and, points 5,9 are found for (c) instead of the MPP vertex 10. By repeating the pattern of 2-3-4-5 a number k of times, complexes may be constructed where Algorithm 1 finds k erroneous points as MPP vertices, causing en error between the length $\sqrt{(1+2k)^2+(1+3k)^2}+\sqrt{2}$ of the MPP frontier part from point 1 to the point numbered $4(k+1)$, and the curve length $k\sqrt{13}+2\sqrt{2}$ based on the points computed by the algorithm. The sequence 1-2-$\cdots$-12 of complex (c) was reported in the appendix of [27] to show that the MPP algorithm from [12] may fail.

4 Minimum perimeter polygon for complexes in mosaics

A ***mosaic*** is defined in [20,21] as a locally finite family of compact convex subsets of $\mathbb{R}^2$, there called *cells*, each of them having non-empty interior, whose union covers the plane, and whose interiors are pairwise disjoint. We will name these sets as ***tiles***. The family is *locally finite* if for any $p \in \mathbb{R}^2$ there exists an open disc centered at p which meets only a finite number of tiles. It follows that each tile is a convex polygon, then the mosaic is a *polygonal plane tiling* due to [8,17], that conclusion is part of the folklore, for a proof see [29]. The intersection of any two tiles, either is empty, or is a non-zero length straight line segment called an ***edge*** of the tiling, or is a single point which is a vertex of at least one of these tiles. Any edge belongs to the frontier of exactly two tiles, in the case that each edge coincides with a full side of each of these two tiles, the mosaic is called *edge-to-edge* [8]. A ***rectangular mosaic*** is an edge-to-edge mosaic whose elements all are rectangles [20,21].

Two distinct tiles sharing an edge are called ***edge-adjacent***, ***adjacent***, or ***edge-neighbors***. For a rectangular mosaic, edge-adjacency generalizes the relation of 4-neighbors used in the discrete plane $\mathbb{Z}^2$. Recall that $p = (x_1, y_1), q = (x_2, y_2) \in \mathbb{Z}^2$ are *4-neighbors* if $|x_1 - y_1| + |x_2 - y_2| = 1$. The ***4-neighborhood graph*** is given by the node set $\mathbb{Z}^2$ and the 4-neighbors relation, this provides 4-paths and 4-connectivity [12]. By identifying each pixel $p \in \mathbb{Z}^2$ with the unitary square centered at p, $\mathbb{Z}^2$ becomes identified with a plane tiling of squares, where two tiles are edge-neighbors if they share a side. These terms are easily generalized to the plane $(c\mathbb{Z})^2$ with $c \in \mathbb{R}$, being $c > 0$ the distance between adjacent pixels. Here, each pixel is identified with a square tile of side length c. A rectangular mosaic $\mathcal{M}$ represents a more general discrete plane where each pixel p is identified with the rectangle centered at p, each rectangle has four edge-neighbors. The ***(edge-)adjacency graph*** $G(\mathcal{M})$ given by the node set $\mathcal{M}$ and the edge-adjacency relation, is a straightforward generalization of the 4-neighborhood graph, it provides ***adjacency-paths*** and ***adjacency-connectivity***. In analogy to *simple 4-curves* used in Digital Geometry [12], also called *Jordan digital curves*, we define a ***simple adjacency-path*** as closed adjacency-path whose each element has exactly two edge-neighbors in this path.

In more general models, a digital image is defined on a set of pixels (a discrete set in $\mathbb{R}^2$) identified with the elements of a polygonal tiling, for example, of the triangular or the hexagonal tilings. As defined in [21], an ***acute mosaic*** satisfies that the union of any two adjacent tiles forms a convex set. Evidently any acute

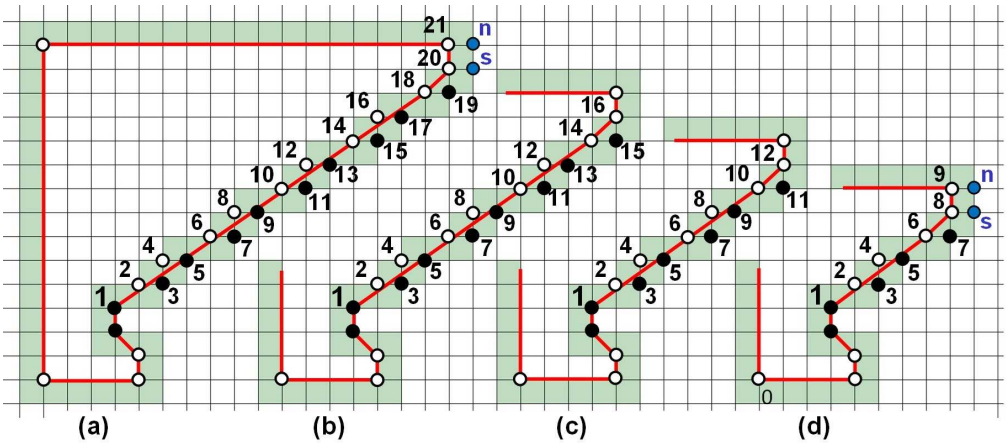

Fig. 3 Four complexes with their simple 4-contours shaded (only the boundary of the largest one is completely drawn), white discs present the convex vertices of the inner wall, the mirrored concave vertices are drawn as solid black discs, the MPP frontiers are outlined in red (see Examples 2 and 11).

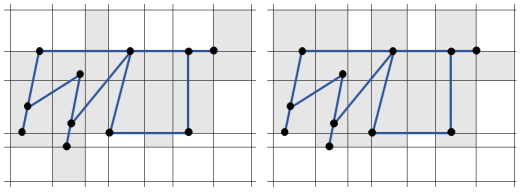

Fig. 4 Two complexes (shaded grey) which are images of the same polygon.

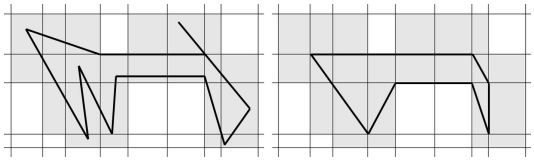

Fig. 5 Two polygons which are preimages of the same complex (shaded grey), the right polygon is the MPP.

mosaic is edge-to-edge, both the quadratic and triangular tilings are acute mosaics, but the hexagonal tiling is not. Modelling a digital image as defined on a mosaic, objects of interest become sets of tiles, they are understood as the image under some digitization mapping from the Euclidean plane $\mathbb{R}^2$ onto the mosaic. The authors of [21] interpret that digitization as a "many-to-one transformation produced by an artificial retina".

Now let $\mathcal{M}$ be a mosaic fixed in $\mathbb{R}^2$. We adopt from [20,21] the notion of ***complex*** defined as any finite non-empty subset of $\mathcal{M}$. We will not use notions from [20, 21] such as cellular mosaic, cell (instead of tile), cellular complex, cellular image, or cellular boundary, to avoid confusions with the structure of cellular complex or cell complex known from combinatorial topology which also is used for modelling discrete sets [12,14,29]. For any complex $\mathcal{C} \subseteq \mathcal{M}$, its point set union $|\mathcal{C}| = \bigcup \mathcal{C} = \{p \in \mathbb{R}^2 : p \in T \text{ for some tile } T \in \mathcal{C}\} \subset \mathbb{R}^2$ is named its *body*.

Definition 2 (from [20,21]) Assume that $\mathcal{C} \subseteq \mathcal{M}$ is a complex and $P \subset \mathbb{R}^2$ is a polygon. Then C is called an ***image*** of a P, and P is called a ***preimage*** of $\mathcal{C}$, if $P \subseteq |\mathcal{C}|$ and $T \cap P \neq \emptyset$ for each tile $T \in \mathcal{C}$.
Any polygon that has shortest perimeter among all polygons which are preimages of $\mathcal{C}$, is called a ***minimum perimeter polygon (MPP)*** of $\mathcal{C}$.

This definition does not provide a digitization mapping since the image of a polygon is not a unique complex, see Figures 4, 5. In general, the MPP of $\mathcal{C}$ is not unique, and its vertices are not guaranteed to belong to the set of vertices of the tiles of $\mathcal{C}$ [20,21]. To report on restrictions on complexes and properties of the MPP which lead to its uniqueness and then to an algorithm for determining its vertices, some more definitions are quoted below from [20,21]. For this, let $\mathcal{C}$ be a complex in a rectangular mosaic $\mathcal{M}$.

In [20,21], an adjacency-path $\beta = (T_1, T_2, \cdots, T_k)$ in $\mathcal{C}$ is called a ***chain***, it is *closed* if T_k and T_1 also are adjacent. A chain is named ***regular*** if $T_{i-1} \neq T_{i+1}$ for each $2 \leq i \leq k-1$ (and $T_{k-1} \neq T_1$, $T_k \neq T_2$, for a closed chain). As already defined above for paths, β is a ***simple chain*** if it is closed, and each tile of β has exactly two edge-neighbors in β. For any two adjacent tiles S, T in a chain, the edge shared by S and T is called a ***transversal edge***. A non-empty adjacency-connected complex is called *chained complex*.

The ***boundary*** $\mathcal{B}(\mathcal{C})$ of $\mathcal{C}$ is defined as the set of tiles of $\mathcal{C}$ that meet the frontier $fr(|\mathcal{C}|)$. Any closed chain which consists exactly of all tiles of $\mathcal{B}(\mathcal{C})$, is called a ***boundary chain*** of $\mathcal{C}$, $\mathcal{B}(\mathcal{C})$ is *non-repeating* if its elements are pairwise distinct. $\mathcal{B}(\mathcal{C})$ is a uniquely defined set, but there may exist several boundary chains of $\mathcal{C}$, some of them being regular or non-repeating, or simple,

others not. If $|\mathcal{C}|$ has holes, $\mathcal{B}(\mathcal{C})$ may result as not edge-adjacency-connected, then $\mathcal{C}$ has no boundary chain.

If $\mathcal{C}$ has at least two elements, a tile $T \in \mathcal{C}$ is called ***end tile***[2] if T is adjacent to exactly one other tile of $\mathcal{C}$. If $\mathcal{C}$ has at least two elements and $|\mathcal{C}|$ is connected, a vertex q of a tile $T \in \mathcal{C}$ is called ***cut point***[3] for $\mathcal{C}$ if the set $(|\mathcal{C}| \setminus \{q\})$ is not connected in $\mathbb{R}^2$. $\mathcal{C}$ is called ***regular complex*** if the frontier $fr(|\mathcal{C}|)$ is a Jordan curve (then $\mathcal{C}$ is called *simply chained* in [20,21]) and $\mathcal{C}$ has a regular boundary chain. A regular complex $\mathcal{C}$ has no end tile and no cut point. If $fr(|\mathcal{C}|)$ is a Jordan curve then $|\mathcal{C}|$ is (simply) connected, and there is no cut point for $\mathcal{C}$, hence $\mathcal{C}$ is edge-adjacency-connected. $\mathcal{C}$ is a ***normal complex*** if it has no end tile and $|\mathcal{C}|$ is a simply connected subset of $\mathbb{R}^2$. Clearly, any regular complex is normal.

As defined in [21], the ***core*** of $\mathcal{C}$ is the point set union of all tiles of $\mathcal{C}$, all edges belonging to tiles of $\mathcal{C}$, and all vertices of tiles of $\mathcal{C}$, whenever these do not meet the curve $fr(|\mathcal{C}|)$. We will denote this as $core(\mathcal{C})$, note that $core(\mathcal{C}) \subset \mathbb{R}^2$. If any of the end points of an edge belongs to $fr(|\mathcal{C}|)$, that edge does not contribute to the core. Also, if the frontier of a tile of $\mathcal{C}$ intersects $fr(|\mathcal{C}|)$, this tile is not part of the core. $\mathcal{C}$ is a ***simply cored complex*** if $core(\mathcal{C}) \neq \emptyset$, and $\mathcal{C}$ has a non-repeating boundary chain, which implies that $fr(|\mathcal{C}|)$ is a Jordan curve (Theorem 5 of [20]).

The definition of core in [20] (1972) depends on a boundary chain which, in general, is not unique for the boundary $\mathcal{B}(\mathcal{C})$. To avoid vagues details resulting from this, we quoted above the definition of core from [21] (1976), which also needs additional effort to relate the convex vertices of the MPP with certain points of the core. Even for a regular complex, $core(\mathcal{C})$ not necessarily is a polygon, $core(\mathcal{C})$ may be non-connected, its connected components are weakly simple polygons. In generalized sense, all vertices of all connected components of $core(\mathcal{C})$ are considered as ***vertices of the core***, for example, both end points of each component consisting in a straight line segment, and also each isolated point of $core(\mathcal{C})$, are convex vertices of $core(\mathcal{C})$.

Part (1) of the following Lemma 1 was proved in [20] (Theorem 6), part (2) was stated in Theorem 8 of [20], partly as consequence of facts affirmed to be shown in [18]. As shown in [21], for any points a, b in a polygon B, there is a unique polygonal shortest path, called *geodesic* in B, which connects a and b. For two polygons $A \subseteq B$, A is called *convex relative* to B if A contains each geodesic in B which connects points a, b from A. The relative convex hull of A with respect to B (also called *geodesic hull*, see our Section 3) is the intersection of all sets which contain A and are convex relative to B. For a normal complex $\mathcal{C}$, the set of all its cut points and all vertices of its core $\mathcal{C}$ is called the ***spread*** of $\mathcal{C}$ in [21], where the MPP of $\mathcal{C}$ was shown to coincide with the relative convex hull of the spread of $\mathcal{C}$ with respect to $|\mathcal{C}|$. The spread of a regular complex coincides with the set of all vertices of its core, which implies part (3) of Lemma 1.

Lemma 1 *Any regular complex $\mathcal{C}$ in a rectangular mosaic satisfies the following:*
(1) $\mathcal{C}$ has a uniquely determined MPP.
(2) Any concave vertex of the MPP is a concave vertex of $fr(|\mathcal{C}|)$, and any convex vertex of the MPP is a convex vertex of $core(\mathcal{C})$.
(3) The MPP of $\mathcal{C}$ coincides with the relative convex hull of $core(\mathcal{C})$ with respect to $|\mathcal{C}|$. In particular, the MPP contains $core(\mathcal{C})$ and is contained in $|\mathcal{C}|$.

The Sklansky-Kibler MPP algorithm was proposed in [21] for normal complexes in acute mosaics, but in the present article, we only consider it for the restricted case of regular complexes in rectangular mosaics.

The algorithm from [4,6] described in our Section 3 works with simple 4-contours of special 4-connected subsets of $(c\mathbb{Z})^2$. Comparing with the concepts from [20,21], $J^+(S)$ and $J^-(S)$ are edge-adjacency-connected complexes of square tiles, $F = J^+(S) \setminus J^-(S)$ is the boundary of the complex $\mathcal{C} = J^+(S)$ representable by a simple boundary chain $\beta(\mathcal{C})$. Under this special condition, $core(\mathcal{C})$ is connected, tracing $\beta(\mathcal{C})$ corresponds to trace the inner wall $fr(core(\mathcal{C}))$ which contains the convex MPP vertices. The concave MPP vertices belong to the outer wall $fr(|\mathcal{C}|)$ but are in bijective relation to the concave vertices of the inner wall. This latter may be not valid for more general complexes. For example, the core of the complex in Figure 8 is not connected, and the "inner wall" of its boundary does not exist.

5 Sklansky-Chazin-Hansen MPP algorithm

5.1 Suppositions and main ideas

The Sklansky-Chazin-Hansen MPP algorithm (1972) from [20] applies to a boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ of any regular complex $\mathcal{C}$ in a rectangular mosaic, $\beta(\mathcal{C})$ corresponds to counterclockwise tracing of the Jordan curve $fr(|\mathcal{C}|)$. Then $|\mathcal{C}|$ has no hole, no cut

[2] In [20], T is defined as end cell "if $S = T \cap (|\mathcal{C}| \setminus T)$ is a single edge of T", but then $S = \emptyset$ since T is closed. A correct version would require that $S = T \cap cl(|\mathcal{C}| \setminus T)$ is a single edge (of $\mathcal{M}$) which makes sense only if $\mathcal{C}$ has at least two elements, this is our definition above.

[3] A cut point is defined in [21] for any connected subset of $\mathbb{R}^2$, but later used as special element of a complex, however, a complex only contains tiles.

point, but has a regular boundary chain. It is not stated in [20] whether $\beta(\mathcal{C})$ should be regular or not. It is assumed that the mosaic is in axis parallel orientation due to the standard Cartesian coordinate system.

The algorithm pretends to construct the MPP frontier of $\mathcal{C}$ as polygonal curve traced in counterclockwise sense, which intuitively is a "stretched elastic thread" lying within the boundary (body) $|\mathcal{B}(\mathcal{C})|$ and passing through all tiles of $\beta(\mathcal{C})$ due to their order in a given boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$. Since each MPP vertex coincides either with a concave vertex of $fr(|\mathcal{C}|)$, or with a convex vertex (in generalized sense) of $core(\mathcal{C})$, all MPP vertices are vertices of the tiles of $\beta(\mathcal{C})$. During tracing this chain, the algorithm constructs a joint ordered list of MPP vertex candidates, being concave vertices of $fr(|\mathcal{C}|)$ and convex vertices of $core(\mathcal{C})$.

The algorithm starts with finding the lower right corner point m_1 of the leftmost tile T_0 in the top row of $\mathcal{C}$, then m_1 is a convex MPP vertex of $\mathcal{C}$ as proved in [20]. From each last found MPP vertex m_n, a straight line segment confined by $|\mathcal{B}(\mathcal{C})|$, is traced as long as possible up to the next MPP vertex m_{n+1}. It is constructed as belonging to a cone rooted at m_n which shows visibility from m_n through the forthcoming tiles of $\beta(\mathcal{C})$. The left and right cone borders are updated as more tiles are taken into account, the cone is expected to become more and more narrow. Meanwhile the cone is well-defined, the straight line segment to be constructed may be enlarged. When a contradiction about that cone is generated, the next MPP vertex m_{n+1} is found. To determine the cone borders, the frontier of each tile of $\beta(\mathcal{C})$ is decomposed in two parts, and some angle calculus is done. If $core(\mathcal{C})$ is connected, tracing $\beta(\mathcal{C})$ means to trace both the whole "outer" frontier $fr(|\mathcal{C}|)$ and "inner" frontier $fr(core(\mathcal{C}))$.

5.2 Tools

Definition 3 (from [20]) For any regular complex $\mathcal{C}$ with boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \beta_2, \cdots, \beta_t)$ and $1 \le i \le t$, decompose $fr(\beta_i) = w_i \cup v_i \cup u_i$ as follows: Let $w_0 = \beta_0 \cap fr(|\mathcal{C}|)$, and w_i be the set of points of $\beta_i \cap fr(|\mathcal{C}|)$ which can be connected to w_{i-1} via a path in $fr(|\mathcal{C}|)$. Let $v_i = (\beta_i \cap \beta_{i-1}) \cup (\beta_i \cap \beta_{i+1})$ (union of two edges). Let $u_i = fr(\beta_i) \setminus w_i \setminus \hat{v_i}$ where $\hat{v_i}$ is the union of the edges $(\beta_i \cap \beta_{i-1})$ and $(\beta_i \cap \beta_{i+1})$ but excluding their end points.

It is stated in [20] that, up to 90° rotations, only three situations may occur for any triple $(\beta_{i-1}, \beta_i, \beta_{i+1})$ of tiles in $\beta(\mathcal{C})$, see Figure 6, for $1 \le i \le t-1$ or $(i-1, i, i+1) = (t-1, t, 0)$ or $(i-1, i, i+1) = (t, 0, 1)$:

Case (a) $w_i = \beta_i \cap fr(|\mathcal{C}|)$ is a single point which is a concave vertex of $fr(|\mathcal{C}|)$, u_i is the connected union of two sides of β_i. Abusing from notation, we will denote by w_i (which is an one point set) also the point itself, it is a candidate for a concave MPP vertex. Informally said, β_i performs a right turn.
Case (b) $w_i = \beta_i \cap fr(|\mathcal{C}|)$ is the connected union of two sides of β_i, u_i is a single point which is a convex vertex of $core(\mathcal{C})$. Abusing from notation, we will denote by u_i (being an one point set) also the point itself, it is a candidate for a convex MPP vertex. Informally said, β_i performs a left turn.
Case (c) $(\beta_{i-1}, \beta_i, \beta_{i+1})$ performs straight passing, $w_i = \beta_i \cap fr(|\mathcal{C}|)$ is a side of β_i which belongs to $fr(|\mathcal{C}|)$, u_i is the opposite side of β_i.

Each MPP vertex p coincides with the point w_i found in Case (a), or with the point u_i found in Case (b), for a unique i, $0 \le i \le t$. Then, if b_i denotes the centre point of the tile β_i, b_i is a convex or concave vertex of the curve $\overline{b_{i-1}b_i} \cup \overline{b_ib_{i+1}}$. As explicitly indicated in Figure 9 of [20], the ***reference vector***[4] $\boldsymbol{r}$ is defined as the ray which starts at the point p and whose direction is opposite to the direction of the vector $\overrightarrow{b_{i-1}b_i}$, $\boldsymbol{r}$ will be used for angle measurements which are essential to find the MPP vertices.

Vectors from p to points of the sets w_k and u_k of any tile $\beta_k \in \beta(\mathcal{C})$ with $i+1 \le k \le t$ are considered in [20], and angles are defined as follows:

$$h_k = \max\{\measuredangle(\mathbf{r}, \overrightarrow{pz}), z \in u_k\},\ g_k = \min\{\measuredangle(\mathbf{r}, \overrightarrow{pz}), z \in w_k\}$$

where $\measuredangle(\mathbf{r}, \overrightarrow{pz})$ denotes the angle at the point p from the vector $\boldsymbol{r}$ to the vector $\overrightarrow{pz}$ measured in mathematical negative (clockwise) sense, quoting the definition[5] from Figure 11 of [20].

The sets w_i and u_i from Definition 3 are of essential importance for the Sklansky-Chazin-Hansen Algorithm. Regularity of $\mathcal{C}$ is required for Definition 3 to guarantee that any situation presents one of the cases

[4] We quote the definition of reference vector $\boldsymbol{r}$ from [20] as given there in Figure 9. The text on page 266 of [20] defines $\boldsymbol{r}$ as starting at p and pointing clockwise along $fr(|\mathcal{C}|)$ or on the core boundary, depending on whether p is concave (authors say "lies on $|\mathcal{C}|$", but it lies on $fr(|\mathcal{C}|)$) or convex (authors say "lies on core($\mathcal{C}$)", but it should lie on the core boundary, or, on the spread, since not each core point can be an MPP vertex). This declaration contradicts the indications of Figure 9 which are applied in the example of Figure 11. It is quite unclear what does it mean for a vector to point clockwise along the core boundary when this latter set may be disconnected or even a set of isolated points.

[5] The text on page 266 of [20] defines $\measuredangle(\mathbf{r}, \overrightarrow{pz})$ as "the absolute value of the angle swept out by a vector through p starting from $\boldsymbol{r}$ and rotating counterclockwise to p." This statement contradicts Figure 11 on the same page where all angles are measured in clockwise sense from $\boldsymbol{r}$ for a convex MPP vertex in an example.

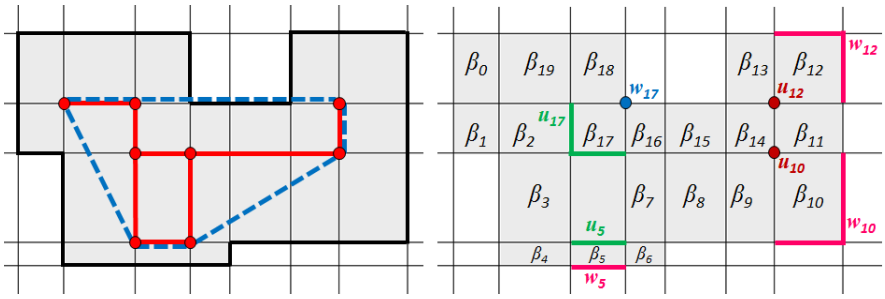

Fig. 6 A complex $\mathcal{C}$ shaded grey with its frontier $fr(|\mathcal{C}|)$ (black), the frontier of $core(\mathcal{C})$ (red), and the MPP frontier (blue broken lines). Right figure: a regular boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_{19})$ illustrating the possible cases for $(\beta_{i-1}, \beta_i, \beta_{i+1})$ and w_i, u_i due to Definition 3. Case (a) is valid, for example, for $i = 17$, the point w_{17} (concave vertex of $fr(|\mathcal{C}|)$) is drawn blue and the set u_{17} green. Case (b) is satisfied for $i = 10, 12$, the points u_{10}, u_{12} (convex vertices of $core(\mathcal{C})$) are marked brown, the sets w_{10}, w_{12} pink. Case (c) occurs for $i = 5$, the line segments w_5 and u_5 are drawn pink and green, respectively.

(a),(b),(c), as stated in [20]. A non-regular complex may have end tiles where a distinct situation would occur. By definition, all sets w_i lie on $fr(|\mathcal{C}|)$, visiting them in the order $w_0, w_1, \cdots, w_t$ is expected to correspond to counterclockwise tracing of $fr(|\mathcal{C}|)$. Proposition 1 of [20] states that all sets u_i belong to the frontier of $core(\mathcal{C})$ if $\mathcal{C}$ is simply cored[6]. As example, the complex presented in Figure 8 has thin parts, hence it is not simply cored since any boundary chain repeats tiles, some sets u_i do not lie on the core but on $fr(|\mathcal{C}|)$ as subsequent figures show. Even so, w_i and u_i are well-defined for all tiles for this example.

It is not stated in [20] whether the type of boundary chain $\beta(\mathcal{C})$ should be restricted, neither for Definition 3 nor for the algorithm. It turns out that even regularity of the boundary chain of a regular complex does not guarantee the concepts of Definition 3 to be well-defined. Figure 7 shows a complex $\mathcal{C}$ with a regular boundary chain $\beta(\mathcal{C})$, where Definition 3 gives nonsense results which make fail the sets w_i and u_i to provide MPP vertex candidates. They also make false the fact stated in [20] that the sequence $(w_0, w_1, \cdots, w_t)$ corresponds to tracing the frontier of $|\mathcal{C}|$. That complex $\mathcal{C}$ has non-empty core. Additionally to $\beta(\mathcal{C})$, $\mathcal{C}$ also has a non-repeating boundary chain given as (β_0, β_1, β_2, β_5, β_6, β_7, β_4, β_3), using the notation from Figure 7. Hence $\mathcal{C}$ is simply cored. However, for the chain $\beta(\mathcal{C})$, some u_i do not lie on the frontier of $core(\mathcal{C})$, declaring false Proposition 1 of [20].

Complexes as the last example which are "two tiles thick", may have arbitrary length and may be parts of any regular complex, giving rise to regular boundary chains with numerous tiles where the sets w_i and u_i are not well-defined, hence the Sklansky-Chazin-Hansen algorithm would not be applicable. Definition 3 and the algorithm must be restricted to a class of boundary chains which guarantees that all w_i are non-empty subsets of $fr(|\mathcal{C}|)$, that a concave MPP vertex candidate is provided in each Case (a), and a convex candidate in each Case (b). The class of non-repeating boundary chains is too restrictive since a regular complex may have thin parts where any boundary chain passes twice. In Section 7, we will restrict our new algorithm to boundary chains obtained by *boundary tracing*.

[6] Proposition 1 of [20] affirms a fact about the sets u_i, but does not restrict the type of boundary chain.

5.3 Determination of MPP vertices

The first part of the algorithm finds a special convex MPP vertex m_1. This is performed by scanning some finite simply chained complex $\mathcal{N}$ in (the infinite mosaic) $\mathcal{M}$ with $\mathcal{C} \subset \mathcal{N}$ where the tiles are labeled to belong to $\mathcal{C}$ or to the background $(\mathcal{N} \setminus \mathcal{C})$. Applying that the rectangular tiles are in axis parallel position and $\mathcal{N}$ is edge-to-edge, scanning is performed from the top to the bottom of $\mathcal{N}$ on horizontal lines, each one from the left to the right, until finding the leftmost tile T_0 in the top row of $\mathcal{C}$. The lower right corner point C_0 of T_0 is a convex MPP vertex (Theorem 10 in [20]). As $\beta(\mathcal{C})$ contains all tiles of $\mathcal{B}(\mathcal{C})$, $T_0 \in \beta(\mathcal{C})$, cyclic shifting of the list $\beta(\mathcal{C})$ is performed until $\beta_0 = T_0$. The resulting chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \beta_2, \cdots, \beta_t)$ is used as input to generate the output sequence of polygon vertices.

The second part, called "MPP Finder" in [20], assuming that the last MPP vertex was found as vertex of a tile $\beta_i \in \beta(\mathcal{C})$, pretends to find the next MPP vertex as vertex of some $\beta_j \in \beta(\mathcal{C})$, $j \geq i + 1$. Starting with $m_1 = C_0$, the next vertex m_2 is found, then, using m_2, m_3 is found, and so on. When an MPP vertex m_{s+1} found coincides with m_1, the algorithm stops, then the list of MPP vertices $(m_1, m_2, \cdots, m_s)$ is completed. Algorithm 2 quotes the MPP algorithm as proposed on page 267 in [20] (where it was less detailed), although we have changed some notations.

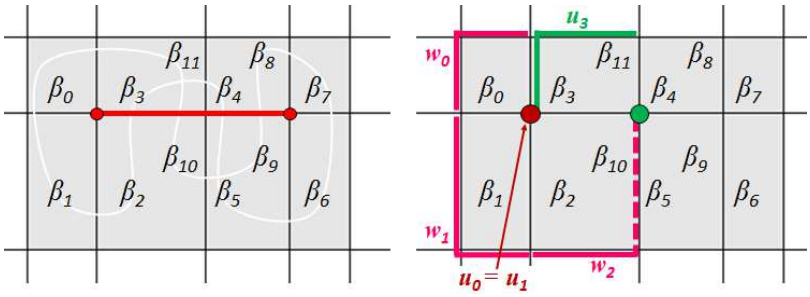

Fig. 7 The regular complex $\mathcal{C}$, shaded grey, coincides with its boundary $\mathcal{B}(\mathcal{C})$, $core(\mathcal{C})$ is marked red in the left figure, it is connected and coincides with its frontier. Moreover, $core(\mathcal{C})$ coincides with the MPP of $\mathcal{C}$, having two convex vertices. The boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_{11})$ is regular but causes problems for Definition 3: w_0, w_1 (marked pink) and $u_0 = u_1$ are well-defined, but w_2 and w_3 are not. Since β_2 corresponds to a left turn, u_2 should result as equal to u_1, providing a convex MPP vertex candidate, and the β_2-side drawn as broken pink line, should belong to w_2. But Definition 3 gives that this side together with the point u_1 forms the set u_2. Even worst for β_3 which defines a right turn and hence should provide a concave MPP vertex candidate w_3: due to Definition 3, w_3 is empty since no point of $\beta_3 \cap fr(|\mathcal{C}|)$ can be connected to w_2 by a path. As result, u_3 (marked green) is a disconnected set which contains the right lower vertex of β_3 (intuitively, that vertex should be w_3). The sets u_2 and u_3 are not subsets of the core of $\mathcal{C}$.

Algorithm 2 Sklansky-Chazin-Hansen algorithm due to [20] to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic.

Input: List $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$, a boundary chain of $\mathcal{C}$.
Output: Lists *MPP* (MPP vertices) and L (candidates).

1: Determine a convex MPP vertex m_1 as the lower right corner of the leftmost tile T_0 in the top row of $\mathcal{C}$. Perform cyclic shifting on $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ until β_0 becomes the tile T_0. Add m_1 to *MPP*, add $C_0 = m_1$ to L. Set $n = 1$, $l = 0$, $p_0 = m_1$ (first found candidate).
2: For the last found MPP vertex m_n, determine $\boldsymbol{r}$ the reference vector and $K(n)$: $K(1) = 1$, for $n \geq 2$, $K(n)$ should satisfy that m_n is the upper right corner of $\beta_{K(n)}$ when $\boldsymbol{r}$ points on the right.
3: Set $k := K(n)$ (continue tracing $\beta(\mathcal{C})$ with fixed m_n).
4: Continue if $k \leq t$, otherwise, the algorithm is finished.
5: Determine w_k, u_k of β_k, and compute angles:
$h_k := \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_n z}), z \in u_k\}$,
$g_k := \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_n z}), z \in w_k\}$,
$H_k := \max\{h_j : K(n) \leq j \leq k\}$,
$G_k := \min\{g_j : K(n) \leq j \leq k\}$.
6: Update the last found candidate: If w_k is a single point then $l := l + 1$, add $D_l = w_k$ to L, $p_k := w_k$. If u_k is a single point then $l := l + 1$, add $C_l = u_k$ to L, $p_k := u_k$. Otherwise $p_k := p_{k-1}$.
7: If $H_k < G_k$ then set $k := k + 1$, then go to Step 4.
If $H_k > G_k$ then set $n := n + 1$, $m_n := p_k$ (new MPP vertex found).
If $m_n = m_1$, the algorithm finishes, otherwise, add m_n to the list *MPP*, then go to Step 2.

In Step 1 of Algorithm 2, n is the index in the list *MPP*, l for the list L. In Step 2, if the last found MPP vertex m_n was provided by β_i, the algorithm restarts from m_n first analysing β_{i+1}, this is the idea behind the function $K(n)$ whose value should then be $i + 1$. The first MPP vertex m_1 comes from β_0, justifying $K(1) = 0 + 1 = 1$. The definition of $K(n)$ from [20] is correct only if m_n is convex, as examples below will show. Step 5 determines the sets w_k, u_k, and calculates the angles. If u_k is a single point then $h_k = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_n z}), z \in u_k\} = \measuredangle(\mathbf{r}, \overrightarrow{m_n u_k})$, similarly $g_k = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_n z}), z \in w_k\} = \measuredangle(\mathbf{r}, \overrightarrow{m_n w_k})$ for a single point w_k. The angles H_k and G_k define the left and right border of a cone of visibility based at m_n through the forthcoming tiles of $\beta(\mathcal{C})$, it is expected to become moren narrow for increasing k.

As affirmed in [20], each β_k satisfies exactly one of the cases (a),(b),(c) related to Definition 3. This permits to construct the candidate list L in Step 6: the point w_k is added as C_l to L in Case (a), the point u_k is added as D_l to L in Case (b). In Case (c), β_k does not provide an MPP vertex candidate. L is a joint list of all MPP vertex candidates, appearing as they are detected tracing $\beta(\mathcal{C})$. Its points are labelled: each C_l is a convex candidate, each D_l is concave, but the algorithm does not use these labels later.

Updating the last found candidate p_k in Step 6, just after having computed H_k and G_k, and before checking whether $H_k < G_k$ or $H_k > G_k$, is the exact transcription of the indications in the algorithm, stated in text and pseudocode on page 267 of [20]: p_k is the convex candidate u_k, or, p_k is the concave candidate w_k, or, $p_k = p_{k-1}$ is confirmed if β_k provides no candidate. Note that p_{k-1} always exists, even for $k = K(1) = 1$ we have $p_0 = C_0$ from Step 1. In Step 7, p_k provides the next MPP vertex if the condition $H_k > G_k$ is satisfied.

5.4 Examples to analyse the Sklansky-Chazin-Hansen algorithm (Algorithm 2)

Example 3: We apply Algorithm 2 to the complex from Figure 8. After finding the first MPP vertex m_1, $\beta(\mathcal{C})$ is shifted to become $\beta(\mathcal{C}) = (\beta_0 = T_0, \beta_1, \beta_2, \cdots,$

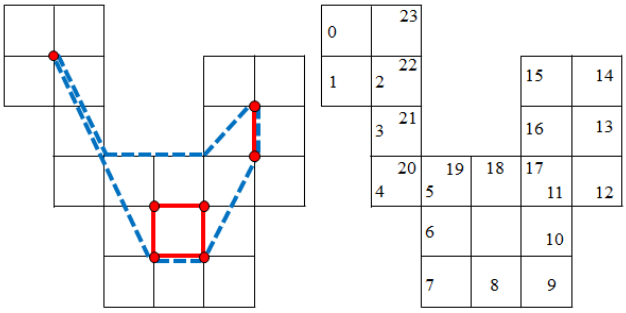

Fig. 8 A regular complex $\mathcal{C}$ of square tiles with boundary chain $\beta(\mathcal{C})$ indicated by the numbers of its tiles $\beta_0, \beta_1, \cdots, \beta_{23}$, the frontier of $core(\mathcal{C})$ is drawn as red, the MPP frontier by broken blue lines.

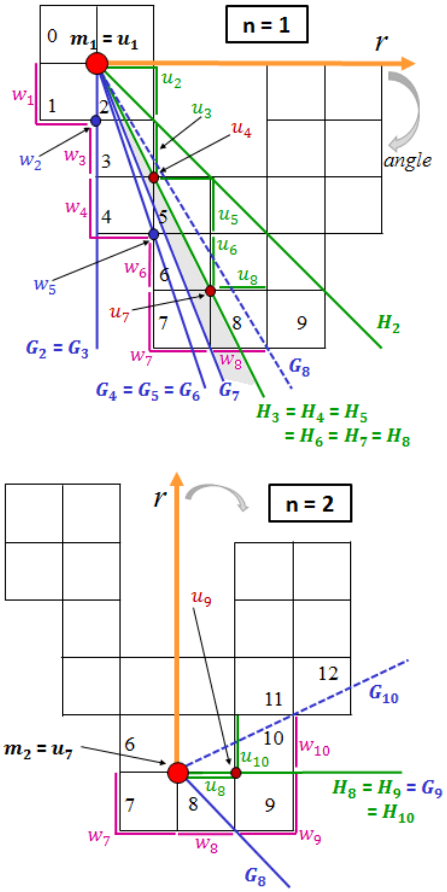

Fig. 9 Algorithm 2 for Example 3, $n = 1$ and $n = 2$.

β_{23}), $p_0 = C_0 = m_1$ is the first found candidate and added to L. Now **n=1**, $K(1) = 1$, the reference vector $\boldsymbol{r}$ is shown in Figure 9 which illustrates the advances of the algorithm. For **k=1**, the candidate $u_1 = m_1$ is found, $h_1 = \measuredangle(\mathbf{r}, \overrightarrow{m_1 m_1})$ cannot be measured since $\overrightarrow{m_1 m_1}$ has zero length. This shows an *error* of the rules in Step 5: if a vector $\overrightarrow{m_n z}$ is degenerated to a single point, h_k,g_k may be not well-defined. *Correction*: H_k and G_k should remain as unchanged from the earlier (k-1)th step; for the starting case $k = K(n)$, it is convenient to set $H_k = 0°$, $G_k = 360°$, then augment k and go to Step 4.

For **k=2**, the candidate $w_2 = D_1$ is found, $h_2 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in u_2\} = 45° = H_2$, $g_2 = \measuredangle(\mathbf{r}, \overrightarrow{m_1 w_2}) = 90° = G_2$, $p_2 = D_1 = w_2$, $H_2 < G_2$. The angles H_2, G_2 define a cone rooted at m_1, $\boldsymbol{r}$ lies on the left from this cone, H_2 defines the left border, G_2 the right border. For **k=3**, β_3 gives no candidate, $h_3 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in u_3\} > 45°$, $H_3 = h_3$ reduces the cone, $g_3 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in w_3\} = 90° = G_3$, $p_3 = p_2$, $H_3 < G_3$. **k=4** gives $u_4 = C_2$, $h_4 = \measuredangle(\mathbf{r}, \overrightarrow{m_1 u_4}) = H_3 = H_4$, $g_4 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in w_4\} < G_3$, $G_4 = g_4$ restricts the cone, $p_2 = D_1 = w_2$, $H_4 < G_4$.

For **k=5,6,7**, the algorithm constructs the cone shadowed grey in Figure 9, $w_5 = D_3$ is found from β_5. **k=8** provides no candidate, $h_8 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in u_8\} = H_7 = H_8$. $g_8 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_1 z}), z \in w_8\} < G_7$, $G_8 = g_8$, $p_8 = p_7$. Since $H_8 > G_8$, we get n=2, the next MPP vertex $m_2 = p_8 = p_7 = u_7 = C_4$ is found. Note that $H_8 > G_8$ means that the cone is empty, its right border defined by G_8, lies on the left from the left border determined by H_8.

Restarting with **n=2** in Step 2, $K(1) = 8$, $\boldsymbol{r}$ is shown in Figure 9. For **k=8**, β_8 gives no candidate, $h_8 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_2 z}), z \in u_8\} = 90° = H_8$, $g_8 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_2 z}), z \in w_8\} = G_8 > H_8$. A cone based at m_2 is opened by H_8 (left border), G_8 (right border).

For **k=9**, the candidate $u_9 = C_5$ is found, $h_9 = \measuredangle(\mathbf{r}, \overrightarrow{m_2 u_9}) = 90° = H_8 = H_9$, $g_9 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_2 z}), z \in w_9\} = 90° < G_8$, $G_9 = g_9 = 90°$, $p_9 = u_9$, $H_9 = G_9$. *Error*: $H_9 = G_9$ is not considered in Algorithm 2. *Correction*: the construction of the MPP frontier should be continued when the visibility cone is reduced to a ray, $H_k < G_k$ in Step 7 should be substituted by $H_k \le G_k$.

For **k=10**, β_{10} provides no new candidate, $h_{10} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_2 z}), z \in u_{10}\} = 90° = H_9 = H_{10}$, $g_{10} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_2 z}), z \in w_{10}\} < G_9$, $G_{10} = g_{10}$, $p_{10} = p_9$. Since $H_{10} > G_{10}$, we get $n = 3$, $m_3 = p_{10} = p_9 = u_9 = C_5$ is the next MPP vertex.

Restarting in Step 2 with **n=3**, $K(3) = 10$, observe the reference vector $\boldsymbol{r}$ in Figure 10. For **k=10**, there is no new candidate, $h_{10} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_3 z}), z \in u_{10}\} = 90° = H_{10}$, $g_{10} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_3 z}), z \in w_{10}\} = 135° = G_{10}$, $p_{10} = p_9 = m_3$, $H_{10} < G_{10}$. For **k=11,12,13**, the candidates $w_{11} = D_6$ and $u_{12} = C_7$ are found, $H_{11} = h_{11}$ and $G_{13} = g_{13}$ reduce the cone which is shaded grey in Figure 10.

k=14 gives $u_{14} = C_8$, $h_{14} = \measuredangle(\mathbf{r}, \overrightarrow{m_3 u_{14}}) < H_{13}$, $H_{14} = H_{13}$, $g_{14} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_3 z}), z \in w_{14}\} < G_{13}$, $G_{14} = g_{14}$, $p_{14} = u_{14}$. Since $H_{14} > G_{14}$, we get $n = 4$, $p_{14} = u_{14}$ is considered as next MPP vertex m_4. This is a *false result* and shows an *error* of Algorithm 2. The

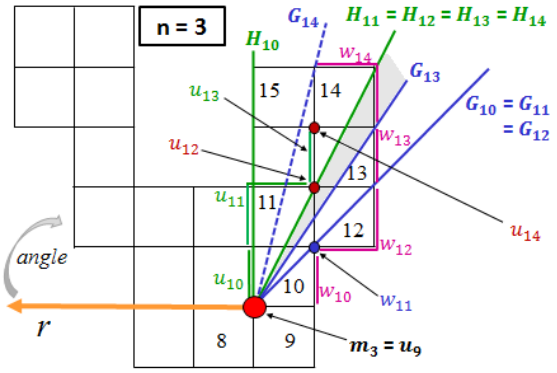

Fig. 10 Advances of Algorithm 2 for Example 3, $n = 3$.

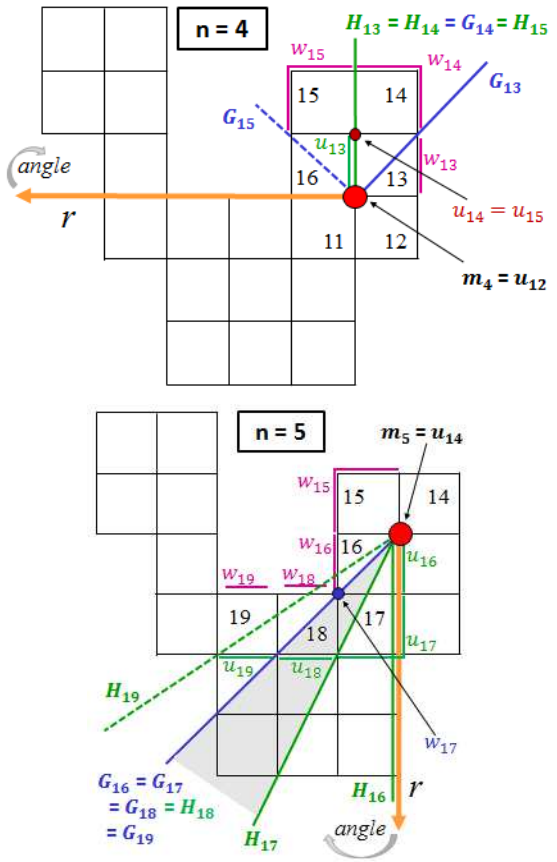

Fig. 11 Algorithm 2 for Example 3, $n = 4$ and $n = 5$.

next MPP vertex should be $p_{13} = p_{12} = u_{12}$, see Figure 10. Skipping u_{12} and taking u_{14} as next polygon vertex, the polygon would not result as preimage of $\mathcal{C}$ since β_{12} would lie in its complement.

Let us restart with **n=4** but using the correct MPP vertex $m_4 = u_{12}$: $K(4) = 13$, $\boldsymbol{r}$ and the cone borders are shown in Figure 11. Using our corrections, for **k=15**, the next MPP vertex $m_5 = p_{14} = u_{14}$ is found. Restarting with **n=5** and $K(5) = 15$, for **k=18**, the cone is reduced to a ray. Proceeding with our correction, for **k=19** one finds the first concave MPP vertex $m_6 = p_{19} = p_{18} = p_{17} = w_{17} = D_9$.

For **n=6** see Figure 12, $K(6)$ should have value 18, but the rule in Step 2 gives the result 17. Starting with β_{17}, $\measuredangle(\mathbf{r}, \overrightarrow{m_6 w_{17}})$ cannot be measured, which leads to start with $k = 18$, using our correction, $\boldsymbol{r}$ is shown in Figure 12. *Error and correction*: The geometric rule is not correct if m_n is concave, then $K(n)$ should satisfy that m_n is the lower right corner of $\beta_{K(n)}$ when $\boldsymbol{r}$ points on the right.

Continuing with **k=18**, β_{18} provides no candidate. Recall that all angles are measured in clockwise sense from $\boldsymbol{r}$ to the vector $\overrightarrow{m_6 z}$. As now m_6 is a concave MPP vertex, the angles turn out as larger that 180°. We get $g_{18} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in w_{18}\}$ which is not defined since $m_6 \in w_{18}$, using our correction, set $H_{18} = 0°$, $G_{18} = 360°$. For **k=19** (no candidate), $h_{19} = H_{19} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in u_{19}\}$, $g_{19} = G_{19} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in w_{19}\} = 270°$, $H_{19} < G_{19}$. A cone based at m_6 is opened by H_{19} (left border) and G_{19} (right border).

For **k=20**, the candidate $w_{20} = D_{10}$ is found, $h_{20} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in u_{20}\} = 270° > H_{19}$, $H_{20} = h_{20}$, $g_{20} = \measuredangle(\mathbf{r}, \overrightarrow{m_6 w_{20}}) = 270° = G_{20} = H_{20}$, $p_{20} = w_{20}$. The cone became reduced to a ray, we proceed due to our correction: **k=21** gives no new candidate, $h_{21} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in u_{21}\} > 270°$, $H_{21} = h_{21}$, $g_{21} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_6 z}), z \in w_{21}\} = 270° = G_{20} = G_{21}$, $p_{21} = p_{20}$. Since $H_{21} > G_{21}$, $n = 7$, $m_7 = p_{21} = p_{20} = w_{20}$ is the next MPP vertex.

Restarting with **n=7**: $K(7) = 21$ due to our correction, $\boldsymbol{r}$ is shown in Figure 12. There are no new candidates for **k=21,22**, $h_{21} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in u_{21}\} = H_{21}$ is well-defined, but $g_{21} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in w_{21}\}$ is not since $m_7 \in w_{21}$. By our correction, set $H_{21} = 0°$, $G_{21} = 360°$. Then $h_{22} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in u_{22}\} = H_{22}$, $180° < H_{22} < 270°$, $g_{22} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in w_{22}\} = 270° = G_{22}$, $H_{22} < G_{22}$.

For **k=23**, $u_{23} = C_{11}$ is found, $h_{23} = \measuredangle(\mathbf{r}, \overrightarrow{m_7 u_{23}}) = H_{22} = H_{23}$, $g_{23} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in w_{23}\} < G_{22}$, $G_{23} = g_{23}$, $p_{23} = u_{23}$, $H_{23} < G_{23}$. The cone defined by H_{23} and G_{23} is shadowed grey in Figure 12. Algorithm 2 ends since β_{23} is the last element of $\beta(\mathcal{C})$, returning the lists $MPP = (m_1, m_2, m_3, m_4, m_5, m_6, m_7)$ and $L = (C_0, D_1, C_2, D_3, C_4, C_5, D_6, C_7, C_8, D_9, D_{10}, C_{11})$ where $C_{11} = u_{23} = u_0 = C_0$.

As another *error*, the stop condition is never reached. It is necessary to visit β_0 after β_{23}, to obtain the condition $H_k > G_k$ satisfied, and so to find the MPP vertex m_1, again. This can be achieved by copying β_0 at the end of the input list $\beta(\mathcal{C})$. Proceeding in this way, $\beta_{24} = \beta_0$, we get **k=24** and $u_{24} = u_{23}$. Then $h_{24} = \measuredangle(\mathbf{r}, \overrightarrow{m_7 u_{24}}) = H_{23} = H_{24}$, $g_{24} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{m_7 z}), z \in w_{24}\} < G_{23}$, $G_{24} = g_{24}$, $p_{24} = u_{24}$. Now $H_{24} > G_{24}$

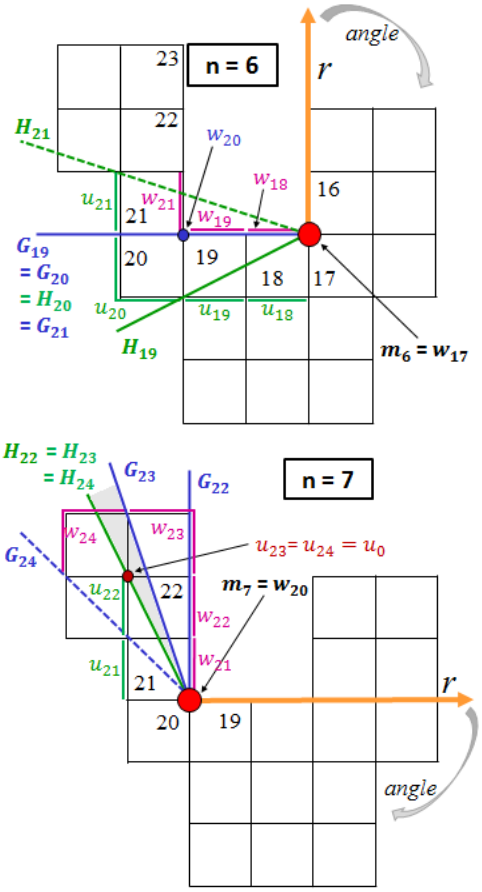

Fig. 12 Algorithm 2 for Example 3, $n = 6$ and $n = 7$.

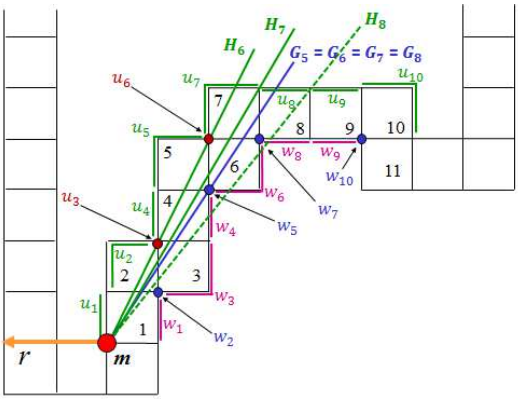

Fig. 13 Algorithm 2 for Example 4.

leads to $n = 8$, $m_8 = p_{24} = u_{24}$ is the next MPP vertex, but $m_8 = m_1$, so it is discarded, the algorithm finishes with the complete list of MPP vertices.

For this small example, Algorithm 2 presents several types of errors which occur a large number of times, it would not determine the correct MPP.

Example 4: We apply Algorithm 2 to the complex in Figure 13, showing a last found MPP vertex m and the cone borders up to $k = 6$, $p_6 = u_6$. For **k=7**,

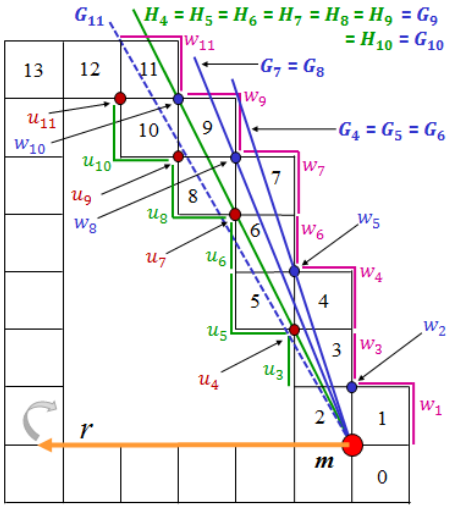

Fig. 14 Complex boundary for Example 5.

the candidate w_7 is found, $h_7 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in u_7\} > H_6$, $H_7 = h_7$ defines a new left cone border. $g_7 = \measuredangle(\mathbf{r}, \overrightarrow{mw_7}) > G_6$ hence $G_7 = \min\{g_7, G_6\} = G_6$, $p_7 = w_7$, $H_7 \leq G_7$. **k=8** provides no candidate, $h_8 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in u_8\} > H_7$, $H_8 = h_8$ defines a new "left border", $g_8 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in w_8\} = G_7 = G_8$, $p_8 = p_7$. Now $H_8 > G_8$, a new MPP vertex $p_8 = p_7 = w_7$ is found. This is a *false result* because $\overline{mw_7}$ does not fit in the boundary $|\mathcal{B}(\mathcal{C})|$. The last candidate found before was $p_6 = u_6$, which also is not the next MPP vertex, although $\overline{mu_6}$ fits in the boundary. The correct next MPP vertex is given as $p_5 = w_5$, but this cannot be found by the algorithm.

Example 5: For the complex in Figure 14, up to $k = 5$ we have $p_4 = u_4$ (convex), $p_5 = w_5$ (concave), $H_5 < G_5$. For **k=6**, $h_6 = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in u_6\} < H_5$, hence h_6 is ignored, $H_6 = H_5$, $g_6 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in w_6\} = G_5 = G_6 > H_6$, there is no new candidate, $p_6 = p_5$.

For **k=7,8**, we find $p_7 = u_7$ ($H_7 = H_6$, G_7 defines a new right border, $H_7 < G_7$), and $p_8 = w_8$ (cone borders confirmed), $H_8 < G_8$. For **k=9**, u_9 is a new candidate, $h_9 = \measuredangle(\mathbf{r}, \overrightarrow{mu_9}) < H_8$ hence $H_9 = H_8$, h_9 is ignored, $g_9 = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in w_9\} < G_8$, $G_9 = g_9 = H_9$, $p_9 = u_9$. The cone has become a ray, by our correction, we proceed with **k=10**, to find the candidate w_{10}, $h_{10} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in u_{10}\} < H_9$ hence $H_{10} = H_9$, h_{10} is ignored. $g_{10} = \measuredangle(\mathbf{r}, \overrightarrow{mw_{10}}) = G_9 = G_{10} = H_{10}$, the cone remains, $p_{10} = w_{10}$. By our correction, we continue.

For **k=11**, u_{11} is found, $h_{11} = \measuredangle(\mathbf{r}, \overrightarrow{mu_{11}}) < H_{10}$ hence $H_{11} = H_{10}$, h_{11} is ignored, $g_{11} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in w_{11}\} < G_{10}$, $G_{11} = g_{11}$ (new right border), $p_{11} = u_{11}$. Now $H_{11} > G_{11}$ which makes us find the next MPP vertex as $p_{11} = u_{11}$ which is a *false result* since $\overrightarrow{mu_{11}}$ leaves the boundary.

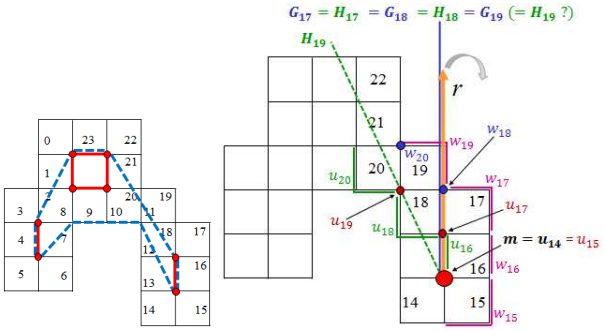

Fig. 15 A complex with the core frontier marked red, the MPP frontier drawn by blue lines. Below: the sets u_k, w_k and the cone borders for $m = u_{14}$ as last found MPP vertex. The reference vector results as lying inside the cone, causing ambiguities for angle calculations and errors.

We have continued twice when the cone has become a ray. Without this correction, Algorithm 2 stops at $k = 9$ or $k = 10$ and finds $p_9 = u_9$ or $p_{10} = w_{10}$ as MPP vertex, the candidate found before was w_8. All these candidates, u_{11}, w_{10}, u_9, w_8, do not provide the next MPP vertex, as may be observed in Figure 14. Clearly u_{11} is an MPP vertex, but not the next one. The curves $m - w_{10} - u_{11}$ and $m - w_8 - u_{11}$ both fit in $|\mathcal{B}(\mathcal{C})|$, the first one has length $\sqrt{36+9} \approx 7.71$, the second one has length $\sqrt{25+4} + \sqrt{4+1} \approx 7.62$. The correct MPP vertex is given as u_7 which is a candidate found long before, the curve $m - u_7 - u_{11}$ fits in $|\mathcal{B}(\mathcal{C})|$ and has the shortest length $\sqrt{16+4} + 2\sqrt{2} \approx 7.30$.

Example 6: For the complex in Figure 15, let us restart from the MPP vertex $m = u_{14}$, the reference vector $\boldsymbol{r}$ is distinct from that of Example 3, we proceed with $k = k_{Start} = 15$. Since $m \in u_{15}$ and $m \in u_{16}$, h_{15}, h_{16} are not well-defined. Let us ignore β_{15}, β_{16}, to continue with **k=17**: the candidate u_{17} is found, $h_{17} = \measuredangle(\mathbf{r}, \overrightarrow{mu_{17}}) = H_{17} = 0$, $g_{17} = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}) : z \in w_{17}\} = G_{17} = 0$, $p_{17} = u_{17}$. The cone is reduced to a ray. Note that β_{15}, β_{16}, β_{17}, all lie on the right of $\boldsymbol{r}$.

k=18 provides the candidate w_{18}. It is not clear whether $g_{18} = \measuredangle(\mathbf{r}, \overrightarrow{mw_{18}}) = 360°$ (β_{18} lies on the left of $\boldsymbol{r}$), or $g_{18} = 0°$ since $w_{18} \in \mathbf{r}$, in any case $G_{18} = 0$. The same doubt applies to $h_{18} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}) : z \in u_{18}\}$: is it 360° ($\beta_{18}$ lies on the left of $\boldsymbol{r}$), or 0° since u_{18} intersects $\boldsymbol{r}$?

– For $h_{18} = 360°$ we get $G_{18} = 0 < 360° = H_{18}$. Algorithm 2 finds w_{18} as MPP vertex which is a *false result*. Moreover, G_{18} and H_{18} describe the same ray, there is no geometric contradiction for the cone, the process should continue.

– For $h_{18} = 0$ we get $G_{18} = 0 = H_{18}$, the cone still is a ray (which is correct). Proceeding to $k = 19$, the candidate u_{19} defines a new left cone border by $H_{19} = h_{19} = \measuredangle(\mathbf{r}, \overrightarrow{mu_{19}})$. On the other hand $g_{19} = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}) : z \in w_{19}\}$ may be interpreted as 360° (β_{19} lies on the left of $\boldsymbol{r}$) or as 0 (w_{19} intersects $\boldsymbol{r}$), in any case $G_{19} = 0 < H_{19}$ which is the condition of Line 18 of Algorithm 2. Hence the last candidate u_{19} is found as MPP vertex which is a *false result* (w_{18} which was found before, would also be false). Note that $G_{19} < H_{19}$ means that the new cone is empty, that is, its right border lies strictly on the left of the previous cone, or, its left border lies strictly on the right of the previous cone, but none of these situations does occur.

5.5 Analysis of Algorithm 2

All examples exposed above use square tiles and regular boundary chains where all sets u_i, w_i due to Definition 3, are well-defined. Even under that ideal conditions, the examples reveal various types of errors.

Procedure errors. .

– The values h_k, g_k are based on angles $\measuredangle(\mathbf{r}, \overrightarrow{pz})$ for $z \in u_k$ or $z \in w_k$, resp., which are not defined for $p = z$. The situation $p \in u_k$ or $p \in w_k$ may occur at each restart after finding a new MPP vertex, β_k then does not provide any candidate, the algorithm should ignore β_k, skip the calculation of h_k, g_k, maintain H_k, G_k as unchanged from the previous (k-1)th step, and then advance to the next tile of $\beta(\mathcal{C})$. For $k = K(n)$, it is justified to define initial values $H_k = 0°$ (H_k is a maximum over angles) and $G_k = 360°$ (G_k is a minimum over angles), this may be added to Step 2.

– Algorithm 2 does not consider[7] the case $H_k = G_k$. Constructing a straight line segment belonging to the MPP frontier, may be continued if the cone is reduced to a ray, the line still fits in $\mathcal{B}(\mathcal{C})$. The condition $H_k < G_k$ in Step 7 should be corrected to $H_k \leq G_k$.

– The rule stated in [20] for defining $K(n)$ is correct only if m_n is convex. For concave m_n, $K(n)$ should satisfy that m_n is the lower right corner of $\beta_{K(n)}$.

– In Example 3, the stop condition never was satisfied. The algorithm pretends to trace $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ until the first MPP vertex m_1 is found, again. This requires to analyse β_0 after β_t.

– Updating p_k is made too early: suppose that p_k is updated in Step 6 and then $H_k > G_k$ in Step 7, that is, $H_{k-1} \leq G_{k-1}$. Then p_k is considered as next MPP vertex which is a false result, because p_k just caused the contradiction $H_k > G_K$ meaning that the cone is empty. Then p_k already fails to be the possible next MPP vertex since $\overline{m_n p_k}$ is not longer guaranteed to fit

[7] If $H_k = G_k$, both conditions of Step 7 are not satisfied, this leads to increase k with no previous updating, which is not always correct.

in $|\mathcal{B}(C)|$. The true next MPP vertex is then given as a candidate found in some previous step.

Theoretical errors. The input data are assumed in [20] as any boundary chain $\beta(\mathcal{C})$ of a regular complex. As observed in Section 5.2, this does not guarantee that each tile β_k satisfies one of the three cases (a),(b),(c) related to Definition 3, and, that cases (a),(b) provide MPP vertex candidates, not even if $\beta(\mathcal{C})$ is regular. This errors makes the algorithm not applicable in some cases. Example 6 shows another error: the reference vector $\boldsymbol{r}$ may result as lying inside the cone which causes ambiguities in angle calculations and erroneous results.

Errors in the correctness proof. Algorithm 2 is affirmed in Theorem 11 of [20] as determining the ordered sequence of MPP vertices. The main argument of the proof supposes that m_n is a MPP vertex, and Algorithm 2 has reached Step 7 and $H_k > G_k$ is true, that is, $H_{k-1} \leq G_{k-1}$. Let s_k be the interior (that is, without its end points) of a segment of the MPP frontier joining m_n by counterclockwise tracing to any point of the tile β_k, assuming also that $K(n) \leq j < k$ for some tile β_j which provides a candidate. The proof in [20] (page 267) states that then s_k contains exactly one MPP vertex which precisely is given as the last found candidate p_k. Both these affirmations are false.

To see this, consider Example 3 from Section 5.4 for $n = 3$ and $k = 14$, see Figure 10. The MPP frontier intersects β_{14} only at the point u_{14}, hence $(\overline{m_3u_{12}} \cup \overline{u_{12}u_{14}}) \setminus \{m_3, u_{14}\}$ is the unique possibility for s_{14}. It is true that s_{14} contains exactly one MPP vertex, but this is u_{12} which does not coincide with the last found candidate $p_{14} = u_{14}$. In Example 4 from Section 5.4, see Figure 13, $H_k > G_k$ is satisfied for the first time for $k = 8$, β_8 provides no candidate, $p_8 = p_7 = w_7$. The MPP frontier tracing from m passes first w_5, then w_7, then continues at the inferior horizontal side of β_8. Hence, for any point $p \neq w_7$ lying on that side of β_8, the curve $\overline{mw_5} \cup \overline{w_5w_7} \cup \overline{w_7p}$ (excluding its end points m and p) is a possible case for s_8. Each such s_8 contains the two MPP vertices w_5 and w_7, but the last found candidate w_7 is not the correct next MPP vertex which should be found for $k = 8$.

Strategy analysis. The determination of the sets u_k, w_k and laborious angle calculus make Algorithm 2 difficult to understand and to implement. From the last found MPP vertex m provided by $\beta_s \in \beta(\mathcal{C})$, Algorithm 2 pretends to construct a straight line segment to the next MPP vertex. This line goes through $\beta_{s+1}, \beta_{s+2}, \beta_{s+3}, \cdots$, until its prolongation would not fit in $|\mathcal{B}(\mathcal{C})|$. All sets u_k, $k = s+1, s+2, \cdots$, are situated on the left, all sets w_k are on the right, of that line. A *current cone* based at m, which contains the desired line, is generated using β_{s+1} and then updated for each k, $k = s+2, s+3, \cdots$, by comparison with the *k-th cone*.

Let us ignore all definition and procedures errors and assume that h_k, g_k are well-defined and $\mathbf{r}$ correctly describes the situation of a vector $\overrightarrow{mz}$ with respect to $\mathbf{r}$, for z belonging to u_k or w_k, or to a cone border, where relations of lying on the left or on the right, are encoded by angles between $\mathbf{r}$ and $\overrightarrow{mz}$. The left border of the k-th cone is given as $\overrightarrow{ma_k}$ where a_k is the point of u_k that lies furthest the right, represented by $h_k = \measuredangle(\mathbf{r}, \overrightarrow{ma_k}) = \max\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in u_k\}$. The angle $g_k = \min\{\measuredangle(\mathbf{r}, \overrightarrow{mz}), z \in w_k\} = \measuredangle(\mathbf{r}, \overrightarrow{mb_k})$ corresponds to the point c_k of w_k which lies furthest to the left, it determines the right cone border $\overrightarrow{mc_k}$. The current (updated) cone, initially given as the $(s+1)$-th cone, is described by H_k (left border) and G_k (right border), it is well-defined if $H_k \leq G_k$. For $k \geq s+2$, the main actions of Algorithm 2 may be described as follows:
– Determine u_k, w_k, a_k, c_k, h_k, g_k.
– If a_k lies strictly on the right of the left current cone border ($h_k > H_{k-1}$) then a_k determines a new left border represented by $H_k = h_k = \max\{h_j : s+1 \leq j \leq k\}$, otherwise $H_k = H_{k-1}$.
– If c_k lies strictly on the left of the right current cone border ($g_k < G_{k-1}$), then c_k determines a new right border represented by $G_k = g_k = \min\{g_j : s+1 \leq j \leq k\}$, otherwise $G_k = G_{k-1}$.
– Update the last found candidate: if u_k is a single point then $p_k = u_k = a_k$, if w_k is a single point then $p_k = w_k = c_k$, otherwise $p_k = p_{k-1}$.
– Final decision: If $H_k \leq G_k$ (originally only stated as $H_k < G_k$) then increase k and continue, incorporating the next tile. If $H_k > G_k$ (the current cone is not longer well-defined), p_k is declared as the next found MPP vertex.

The most important *strategy errors* of Algorithm 2 are that, it does not use the distinction between convex and concave candidates, and, it ignores that not each candidate is prospective to be an MPP vertex.

6 Sklansky-Kibler MPP algorithm

6.1 Strategy, tools, algorithm

We consider the Sklansky-Kibler Algorithm (1976) from [21], originally proposed for normal complexes in acute mosaics, under the restricted suppositions that $\mathcal{C}$ is a regular complex in a rectangular mosaic, with all tiles in axis parallel orientation in the standard Cartesian coordinate system, and $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ is a regular boundary chain of $\mathcal{C}$.

In distinction to the Sklansky-Chazin-Hansen Algorithm, instead of tracing the entire "outer and inner frontiers" $fr(|\mathcal{C}|)$ and $fr(core(\mathcal{C}))$, the idea now is to jump from one transversal edge to the next. These edges are named windows in [21]: for each $\beta_i, \beta_{i+1} \in \beta(\mathcal{C})$, the edge $e_i = \beta_i \cap \beta_{i+1}$ is *the i-th window* with end points[8] $x_i \in fr(|\mathcal{C}|)$ and y_i. Since tracing $\beta(\mathcal{C})$ follows $fr(|\mathcal{C}|)$ counterclockwise, points x_i determine the right border vector of the cone of visibility through the next windows, points y_i determine its left border. The cone is expected to become more narrow as more forthcoming windows are taken into account. As affirmed in [21], finally, when the next window "is not in sight at all", a new MPP vertex m_{n+1} is found: if that window falls on the right (resp., left) hand side of the cone then m_{n+1} lies on the right (resp., left) cone border line, it is the last found vertex candidate on that line. This marks an important difference to the Sklansky-Chazin-Hansen Algorithm, where each new MPP vertex was obtained as the last point of a joint list of candidates. To determine the cone borders, some angle calculus is done with respect to a reference vector distinct from that of Algorithm 2.

"Algorithm M" in [21] starts at a first MPP vertex m given as "any point of the spread of $\mathcal{C}$". For a regular complex, that means that m could be any vertex of $core(\mathcal{C})$. We comment that this is *false* since no concave vertex of the core could be an MPP vertex. There is no specific suggestion in [21] how to find m. So, let us find m as a first convex MPP vertex, as explained in Section 5.3 for the Sklansky-Chazin-Hansen Algorithm. Then m is a vertex of a special tile T_0 of $\mathcal{C}$, suppose now that T_0 coincides with β_0 of $\beta(\mathcal{C})$. Algorithm 3 presents a pseudocode of the Sklansky-Kibler algorithm[9]. It uses the tools from the following definition based on informal definitions from [21].

Definition 4 For any regular boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$, a vertex $m \in \beta_i$ of the MPP of $\mathcal{C}$, $0 \le i \le t$, and a tile β_j with $j \ge i+1$ which is the first one such that m does not belong to the window $e_j = \beta_j \cap \beta_{j+1}$ (that is, $m \notin e_j$ and $m \in e_z$ for all $i+1 \le z \le j-1$), define a reference system and angles: If e_j has end points $x_j \in fr(|\mathcal{C}|)$ and y_j, the straight ray starting at m and passing through x_j is named the ***base line*** (of m). The following values of angles of vectors starting at m are measured from the base line in counterclockwise sense: for any window e_k with $k \ge j$, let $\theta_k = \measuredangle(\overrightarrow{mx_j}, \overrightarrow{mx_k})$, $\theta'_k = \measuredangle(\overrightarrow{mx_j}, \overrightarrow{my_k})$. The ***cone*** $[\theta_k, \theta'_k]$ is defined for any $0 \le \theta_k \le \theta'_k \le 360°$ as the union of all rays which start at m and form angles with the base line of values between θ_k and θ'_k. The cone $[\theta_k, \theta'_k]$ is defined as the empty set if $\theta_k > \theta'_k$.

The angles for the window e_j determine a cone based at m with right border $\overrightarrow{mx_j}$ and left border $\overrightarrow{my_j}$. For forthcoming windows e_k with $k \ge j+1$, the angles serve in the algorithm to update the right (θ_k) and left (θ'_k) cone borders. A cone $[\theta_k, \theta'_k]$ is not empty but degenerated to a straight ray if $\theta_k = \theta'_k$.

Algorithm 3 Sklansky-Kibler algorithm due to [21], to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic.

Input: boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ (counterclockwise tracing), special convex MPP vertex $m \in \beta_0$.
Output: List *MPP* of MPP vertices.

1: Add $m_1 = m$ to the list *MPP*, set $i := 0$, $n := 1$.
2: **if** $i > t$ **then** STOP, **end if**.
3: Determine end points x_i (right) and y_i (left) of window $e_i = \beta_i \cap \beta_{i+1}$.
4: **if** ($m \neq x_i$ and $m \neq y_i$) **then**
5: $\theta_i = 0, \theta'_i = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{my_i})$ ($\overrightarrow{mx_i}$ is a new base line)
6: $\theta = 0$, $\theta' = \theta'_i$ (new right and left cone borders)
7: $k := i$
8: **while** $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ **do**
9: $k := k+1$ (continue with fixed base line)
10: **if** $k > t$ **then** STOP, **end if**.
11: Determine end points x_k, y_k of $e_k = \beta_k \cap \beta_{k+1}$.
12: $\theta_k = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{mx_k}), \theta'_k = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{my_k})$
13: **if** $\theta_k \ge \theta$ **then** ($p := x_k$ and $z_p := k$), **endif**.
14: **if** $\theta'_k \le \theta'$ **then** ($q := y_k$ and $z_q := k$), **endif**.
15: update cone borders:
$\theta := \max\{\theta, \theta_k\}$, $\theta' := \min\{\theta', \theta'_k\}$
16: **end while**
17: (Now $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, a new MPP vertex is found:)
18: **if** $\theta_k < \theta$ **then** ($m := p$ and $z := z_p$), **end if**.
19: **if** $\theta_k > \theta$ **then** ($m := q$ and $z := z_q$), **end if**.
20: $n := n+1$, add the point $m_n = m$ to the list *MPP*.
21: $i := z+1$ (continue searching with β_{z+1})
22: Go to Line 2.
23: **end if**
24: $i := i+1$
25: Go to Line 2.

Algorithm 3 determines in Line 3 the end points x_0, y_0 of window e_0. Since $m_1 = m = y_0$, Line 4 causes to increase i and return to Line 3, this is repeated until ($m \neq x_i$ and $m \neq y_i$) for some i, which is equivalent to $m \notin e_i$. Also for any other last found MPP vertex m which was provided by some β_{i-1}, when returning to Lines 2-3, e_i is ignored if $m \in e_i$. When having found an index i with $m \notin e_i$, Lines 5-6 are reached, $\overrightarrow{mx_i}$ defines

[8] In [21], x_i and y_i are called "coordinates", but they are the end points of the straight line segment $e_i = \beta_i \cap \beta_{i+1}$.

[9] The "Implementation of Algorithm M" in [21] contains four steps. The first two ones describe a boundary detection technique where for a known boundary tile β_0 of $\mathcal{C}$, the next boundary tile β_1 is found, that permits to locate the window b_0. Then β_2 is found and hence the window b_1, and so on. In the present article, we ignore these parts since we are interested only in the MPP algorithm, we use the complete input boundary chain $\beta(\mathcal{C})$ to obtain all windows.

a new base line. The i-th cone $[\theta_i, \theta'_i]$ with $\theta_i = 0$, is opened by x_i, y_i, it provides the initial values $\theta = \theta_i = 0$, $\theta' = \theta'_i$ for the current cone $[\theta, \theta']$. We mention that the latter instruction is explicitly stated in [21] only for the first window (there given for $i = 1$), and erroneously as $\theta = 0, \theta' = \theta_1$ (instead of $\theta = 0$, $\theta' = \theta'_1$), which evidently is a typing error there.

The condition of Line 8 is trivially true for $k = i$. The aim is now to go through forthcoming windows e_k from the list $\beta(\mathcal{C})$, $k \geq i + 1$, with fixed base line $\overrightarrow{mx_i}$, while the cone $[\theta_k, \theta'_k]$ has non-empty intersection with the current cone $[\theta, \theta']$. Note that this condition requires that $[\theta_k, \theta'_k] \neq \emptyset$. It is expected that in most situations, the new angles θ_k, θ'_k restrict, or confirm, the current cone, that is, $\theta_k \geq \theta$ (right cone border) and $\theta'_k \leq \theta'$ (left cone border). This justifies updating by $\theta := \max\{\theta, \theta_k\}$ and $\theta' := \min\{\theta', \theta'_k\}$ in Line 15. For $\theta_k \geq \theta$, θ is updated as θ_k, in case $\theta_k < \theta$, θ_k is ignored. For $\theta'_k \leq \theta$, θ' is updated as θ'_k, in case $\theta'_k > \theta'$, θ'_k is ignored. We mention that the updating instruction for θ' is stated in [21] erroneously as $\theta' := \min\{\theta', \theta_i\}$, which clearly is a typing error there.

Part of "Step 3" of the algorithm on page 647 in [21] states that "As long as $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$, find another boundary cell and its associated window, and set θ to $\max[\theta, \theta_i]$ and θ' to $\min[\theta_i, \theta']$" (they use index i instead of k, and it includes the typing error in the rule for θ'). Transcribing this into our pseudocode, increasing k in Line 9 enables to search for the next boundary cell whose window is determined in Line 11, the new angles θ_k, θ'_k are calculated in Line 12 and are compared with the current cone dates in Lines 13-14.

The algorithm on page 647 in [21] continues with "If $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, then let l be the line ...". For this, note first that $\theta_k \geq \theta$ means restriction or confirmation of the right cone border, then $p := x_k$ is the last found convex candidate on the ray from m_n in direction θ_k. $\theta'_k \leq \theta'$ means restriction or confirmation of the left cone border, then $q := y_k$ is the last found concave candidate on the ray from m_n in direction θ'_k. The index of the tile which provides p or q in case of being updated, will be needed in case that one of these points later becomes the next found MPP vertex, it is saved as z_p or z_q in Lines 13-14, and later in Lines 18-19 as z.

For $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, the instructions in [21] state that a new MPP vertex is found, as follows: in case that $\theta_k < \theta$, let l be "the line" from "the origin" m in direction θ, and, let l be "the line" from m in direction θ' in case that $\theta_k > \theta$. The new MPP vertex is given as "the last vertex on l, excluding its end point", there is no explanation in [21] on this. The "line" l is the ray starting at the current last found MPP vertex, "the origin" m (which is the unique "end point" of l),

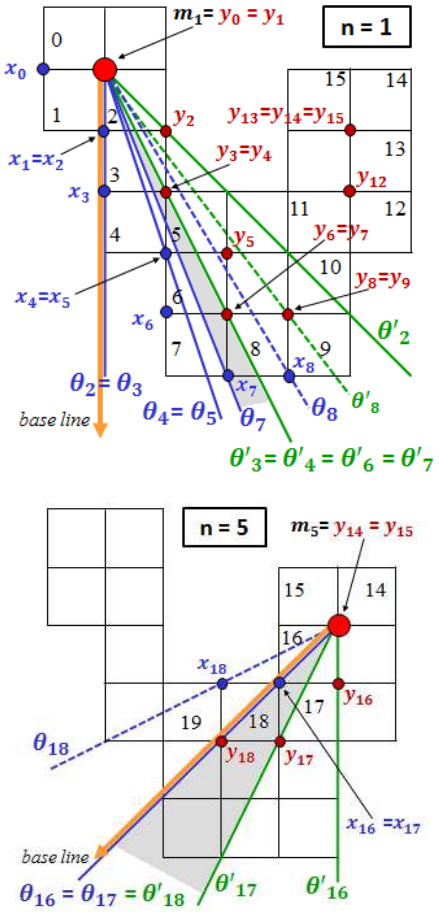

Fig. 16 Algorithm 3 for Examples 7 and 8.

determined by the angle θ or θ'. The angle is defined by the point x_k (θ) or y_k (θ') which lies on l. It is possible that during Lines 8-14 of Algorithm 3, the same angle θ had been defined by some previous point x_j (or θ' by y_j) with $j < k$, so that the points m, x_j, x_k (or m, y_j, y_k) are collinear on l, but evidently, x_k or y_k is the last window end point ("vertex") found on l. Hence, the original instruction is understood such that x_k, or y_k, is "the last vertex on l", the exclusion of its "end point" m is ensured by Line 4.

6.2 Examples and a partial correction

Note that the base line is distinct from the reference vector used in Algorithm 2, and angles now are measured in counterclockwise sense.

Example 7: We apply Algorithm 3 to the complex from Example 3, see Figure 16. For **n=1**, since $y_0 = y_1 = m$, consider $i = 2$: $e_2 = \beta_2 \cap \beta_3$, $x_2 = x_1 \neq m$ and $y_2 \neq m$, a new base line is defined as $\overrightarrow{mx_2}$, $\theta = \theta_2 = 0$, $\theta' = \theta'_2 = 45°$. For **k=2**, $[\theta_2, \theta'_2] \cap [\theta, \theta'] = [\theta_2, \theta'_2] \neq \emptyset$. For **k=3**, $e_3 = \beta_3 \cap \beta_4$, $p = x_3$ since $\theta_3 = \theta$, $q = y_3$

since $\theta'_3 < \theta'$, θ remains the same, $\theta' := \theta'_3$. For **k=4**, $e_4 = \beta_4 \cap \beta_5$, $p = x_4$ since $\theta_4 > \theta$, $q = y_4 = y_3$ since $\theta'_4 < \theta'$, $\theta := \theta_4$, θ' stays the same. For **k=5**, $e_5 = \beta_5 \cap \beta_6$, $p = x_5 = x_4$ since $\theta_5 = \theta$, q is not updated since $\theta'_5 > \theta'$, θ, θ' both do not change. For **k=6**, $e_6 = \beta_6 \cap \beta_7$, p is not updated since $\theta_6 < \theta$, $q = y_6$ since $\theta'_6 = \theta'$, θ, θ' both stay the same. For **k=7**, $e_7 = \beta_7 \cap \beta_8$, $p = x_7$ since $\theta_7 > \theta$, $q = y_7 = y_6$ since $\theta'_7 = \theta'$, $\theta := \theta_7$ (new right cone border), θ' remains the same.

For **k=8**, $e_8 = \beta_8 \cap \beta_9$, $\theta_8 > \theta > \theta'$ implies updating $p = x_8$, since $\theta'_8 > \theta'$, q is not updated ($q = y_7$), $[\theta_8, \theta'_8]$ is a non-empty cone outside the current cone $[\theta, \theta']$. Line 15 requires updating $\theta := \max\{\theta, \theta_8\} = \theta_8$ (new right cone border), $\theta' := \min\{\theta', \theta'_8\} = \theta'$ stays the same. Returning to Line 8, now $[\theta_8, \theta'_8] \cap [\theta, \theta'] = \emptyset$ since $[\theta, \theta'] = \emptyset$ ($\theta > \theta'$). We reach Line 17, a new MPP vertex is found: still $k = 8$, but θ, θ' have already been updated using θ_8, θ'_8, hence now $\theta_8 = \theta$ and $\theta'_8 > \theta'$. The decision in Lines 18-19 whether p or q provides the next MPP vertex, cannot be taken since both conditions $\theta_8 < \theta$ and $\theta_8 > \theta$, are not satisfied. This shows a subtle but important *error* of Algorithm 3: The condition for $k = 8$ makes find a new MPP vertex, comparison of θ_8, θ'_8 with the current cone $[\theta, \theta']$ should be used to decide which of the stored candidates, p or q, provides the next MPP vertex. The updating in Line 15 destroys the possibility of that comparison. *This error may cause that no next MPP vertex is found ever, after the first one* m_1.

The error just detected cannot be corrected by interchanging Lines 8-16 with Lines 17-22 in the pseudocode, that is, first treating the case that $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, up to the command of returning to Line 2, and then dealing with the case that $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$. The updating, currently performed in Line 15, still would be wrongly placed and causes the same problem as described above. That line has to be moved such that updating θ and θ' is performed "at the beginning of the next step". This can be achieved by substituting Lines 8-16 of Algorithm 3 by the Lines 8-16 of Algorithm 4 which presents a partially corrected version of the Sklansky-Kibler algorithm.

Using Algorithm 4, when having restarted in Line 2, and having defined a new base line and initial values for $\theta_i = \theta$, $\theta'_i = \theta'$, $k = i$ (Lines 5-7), the condition of Line 8 $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ is trivially satisfied, and the updating now performed in Line 9 means that θ, θ' remain the same. In any subsequent step, suppose that for some $k \geq i + 1$, after determining θ_k, θ'_k and updating p, q (Lines 14-15), returning to Line 8, $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ turns out to be false, a fact which was caused by the new data θ_k, θ'_k. Line 17 is reached, when the current cone data θ, θ' still are untouched by θ_k, θ'_k, hence one of the two comparisons in Lines 18-19 will be true, and the next MPP vertex is obtained. In the following examples, we will use Algorithm 4, the partially corrected Sklansky Kibler Algorithm.

Algorithm 4 Partially corrected version of the Sklansky-Kibler algorithm, to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic.

Input: boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ (counterclockwise tracing), special convex MPP vertex $m \in \beta_0$.
Output: List *MPP* of MPP vertices.

1: Add $m_1 = m$ to the list *MPP*, set $i := 0$, $n := 1$.
2: **if** $i > t$ **then** STOP, **end if**.
3: Determine end points x_i (right) and y_i (left) of window $e_i = \beta_i \cap \beta_{i+1}$.
4: **if** ($m \neq x_i$ and $m \neq y_i$) **then**
5: $\theta_i = 0, \theta'_i = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{my_i})$ ($\overrightarrow{mx_i}$ is a new base line)
6: $\theta = 0$, $\theta' = \theta'_i$ (new right and left cone borders)
7: $k := i$
8: **while** $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ **do**
9: update cone borders:
$\theta := \max\{\theta, \theta_k\}$, $\theta' := \min\{\theta', \theta'_k\}$
10: $k := k + 1$ (continue with fixed base line)
11: **if** $k > t$ **then** STOP, **end if**.
12: Determine end points x_k, y_k of $e_k = \beta_k \cap \beta_{k+1}$.
13: $\theta_k = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{mx_k}), \theta'_k = \measuredangle(\overrightarrow{mx_i}, \overrightarrow{my_k})$
14: **if** $\theta_k \geq \theta$ **then** ($p := x_k$ and $z_p := k$), **endif**.
15: **if** $\theta'_k \leq \theta'$ **then** ($q := y_k$ and $z_q := k$), **endif**.
16: **end while**
17: (Now $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, a new MPP vertex is found:)
18: **if** $\theta_k < \theta$ **then** ($m := p$ and $z := z_p$), **end if**.
19: **if** $\theta_k > \theta$ **then** ($m := q$ and $z := z_q$), **end if**.
20: $n := n + 1$, add the point $m_n = m$ to the list *MPP*.
21: $i := z + 1$ (continue searching with β_{z+1})
22: Go to Line 2.
23: **end if**
24: $i := i + 1$
25: Go to Line 2.

Returning to our **Example 7** for **k=8**, as obtained above, Lines 8-16 result in $p = x_8$ since $\theta_8 > \theta$, $q = y_7$ since $\theta'_8 > \theta'$. Due to Algorithm 4, θ, θ' are not yet updated, they are still given as $\theta = \theta_7$, $\theta' = \theta'_3$, see Figure 16 for n=1. Returning to Line 8, we have $[\theta_8, \theta'_8] \cap [\theta, \theta'] = \emptyset$, hence Line 17 is reached. Now $\theta_8 > \theta$ (actually $\theta_8 > \theta'$, x_8 lies on the left outside the current cone), hence Lines 18-19 give $n = 2$, $m_2 = m = q = y_7$ correctly as the next MPP vertex, $i = 8$, to restart in Line 2. Algorithm 4 also detects the MPP vertices $m_3 = y_9$, $m_4 = y_{12}$, $m_5 = y_{15}$.

Example 8: We continue the last example restarting with the MPP vertex $m_5 = y_{15} = y_{14}$, then **n=5**, see Figure 16. For $i = 16$, $\overrightarrow{mx_{16}}$ is a new base line, $\theta = \theta_{16} = 0$, $\theta' = \theta'_{16} = 45°$. Then **k=16**, $[\theta_{16}, \theta'_{16}] \cap [\theta, \theta'] \neq \emptyset$ is trivially true, updating in Line 9 maintains θ, θ'. For **k=17**, $x_{17} = x_{16}$ hence $\theta_{17} = \theta$ and $p = x_{16}$. θ'_{17} restricts the left cone border since $\theta'_{17} < \theta'$, then $q =$

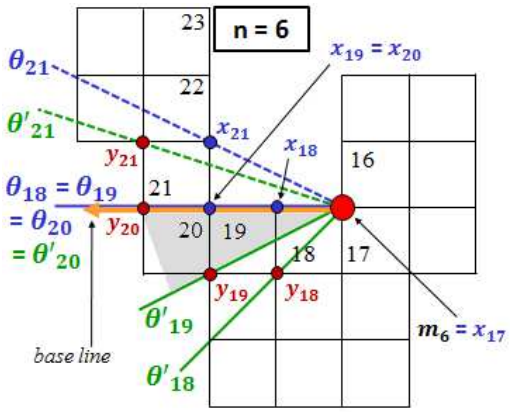

Fig. 17 Algorithm 3 for Example 8, second occurrence of a base line inside the cone.

y_{17}. Returning to Line 8, $[\theta_{17}, \theta'_{17}] \cap [\theta, \theta'] = [\theta_{17}, \theta'_{17}] \neq \emptyset$, updating in Line 9: θ stays the same, $\theta' := \theta'_{17}$.

For **k=18**, $\theta_{18} = \measuredangle(\overrightarrow{mx_{16}}, \overrightarrow{mx_{18}}) > \theta$, hence $p = x_{18}$ and $\theta'_{18} = \measuredangle(\overrightarrow{mx_{16}}, \overrightarrow{my_{18}}) = 0$, θ'_{18} restricts the left cone border, $\theta'_{18} \leq \theta'$ implies $q = y_{18}$. Now $[\theta_{18}, \theta'_{18}] = \emptyset$ because $\theta_{18} > \theta'_{18}$, the condition of Line 8 is not satisfied. This leads to Lines 17-19: $\theta_{18} > \theta$ hence the next MPP vertex is found as $m_6 = m = q = y_{18}$, which is a *false result* and shows an *error* of Algorithm 4. The base line which is the reference vector for angle calculation, does not permit to analyse correctly the situation of x_{18} which lies on the right outside the current cone, the base line "lies inside the scene".

Let us continue with the correct MPP vertex $m = m_6 = x_{17}$, **n=6**, then $z = 17$ and $i = 18$, $\overrightarrow{mx_{18}}$ is a new base line, $\theta = \theta_{18} = 0$, $\theta' = \theta'_{18} = 45°$, see Figure 17. For **k=18**, $[\theta_{18}, \theta'_{18}] \cap [\theta, \theta'] \neq \emptyset$ is trivially true, updating in Line 9 maintains θ, θ'. For **k=19**, x_{19} lies on the base line, hence $\theta_{19} = \theta$ and $p = x_{19}$, θ'_{19} gives a new left border since $\theta'_{19} < \theta'$, then $q = y_{19}$. Returning to Line 8, $[\theta_{19}, \theta'_{19}] \cap [\theta, \theta'] = [\theta_{17}, \theta'_{17}] \neq \emptyset$, updating maintains $\theta = 0$, $\theta' := \theta'_{19}$. For **k=20**, $\theta_{20} = \theta$ since $x_{20} = x_{19}$, hence $p = x_{20}$, $\theta'_{20} = \measuredangle(\overrightarrow{mx_{18}}, \overrightarrow{my_{20}}) = 0$ since y_{20} lies on the base line, then $q = y_{20}$. The cone $[\theta_{20}, \theta'_{20}]$ is on the base line. The condition of Line 8 is true, updating in Line 9 gives $\theta = \theta' = 0$.

Now **k=21**, x_{21} lies on the right outside the current cone, which is not correctly described by $\theta_{21} = \measuredangle(\overrightarrow{mx_{18}}, \overrightarrow{mx_{21}}) \approx 330°$, $\theta_{21} > \theta$ implies $p = x_{21}$. The point y_{21} also lies on the right outside the cone, $\theta'_{21} = \measuredangle(\overrightarrow{mx_{18}}, \overrightarrow{my_{21}}) \approx 340°$, $\theta'_{21} > \theta'$ hence q is not updated. Returning to Line 8, $[\theta_{21}, \theta'_{21}] \cap [\theta, \theta'] = \emptyset$ since the cone $[\theta_{21}, \theta'_{21}]$ is not empty but does not contain the current cone $[\theta, \theta']$ which equals the base line. This leads to Lines 17-19, where $\theta_{21} > \theta$ implies that the next MPP vertex is found as $m_7 = m = q = y_{20}$, which is a *false result*. Again, the base line does not permit correct angle computations.

6.3 Analysis of the Sklansky-Kibler algorithm

Laborious angle calculations and complicated management of intersections of cones, make this algorithm difficult to understand and to implement. The *procedure error* of the original Algorithm 3 described in the last section does not permit the algorithm to find the MPP vertices except the first one. Even after correcting this, Algorithm 4 faces the problem that *the base line may result as situated inside the cone, causing errors* in the angle calculus and hence in the determination of MPP vertices, for certain complexes.

The MPP algorithm in [21] was proposed as a new algorithm for much more general complexes as the previous Algorithm 2 from [20] (partly by the same authors). There is no mention in [21] about any error of the previous version of theory and algorithm from [20]. Even so, Algorithm 4 turns out as correcting important errors of Algorithm 2, mostly in the strategy, but also some procedure details. For example, restarting from a last found MPP vertex m, the first tiles β_i, $\beta_{i+1}, \cdots$ are ignored when containing m, they caused in Algorithm 2 that angles h_k, g_k are not well-defined.

Strategy analysis. Starting with the last found MPP vertex m provided by a window $e_s = \beta_s \cap \beta_{s+1}$, a straight line segment is constructed up to the next MPP vertex, that line goes through e_{s+1}, $e_{s+2}, \cdots$, until its prolongation would not fit in $|\mathcal{B}(\mathcal{C})|$. All window end points y_k, $k = s+1, s+2, \cdots$, are situated on the left, all end points x_k are on the right, of that line. A *current cone* $[\theta, \theta']$ based at m, which contains the desired line, is generated using the first window e_i, $i \geq s+1$, which does not contain m, and then updated for each k, $k = i+1, i+2, \cdots$, by comparison with the *k-th cone* $[\theta_k, \theta'_k]$. In contrast to Algorithm 2 which in any moment stores the last found candidate, Algorithms 3and 4 separately store the last found concave candidate p (some x_k) and convex candidate q (some y_k).

Let us ignore the base line error, supposing that θ_k, θ'_k correctly describe the positions of vectors $\overrightarrow{mx_k}, \overrightarrow{my_k}$ with respect to the base line. The left border of the k-th cone is given as $\overrightarrow{my_k}$ corresponding to θ'_k, θ_k describes the right cone border $\overrightarrow{mx_k}$, the cone is non-empty if $\theta'_k \leq \theta_k$. The current cone $[\theta, \theta']$ (θ' describes its left border, θ the right border) initially is given as the i-th cone, it is non-empty if $\theta' \leq \theta$. The comparison between both cones is performed by analysing the set $[\theta_k, \theta'_k] \cap [\theta, \theta']$: if it is non-empty, certain last found

candidate is updated before incorporating the next window. But $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$ indicates that a new MPP vertex was found, this occurs when the k-th cone is empty, or, when the k-th cone lies strictly outside the current cone. For a base line and a current cone $[\theta, \theta']$ initialized from $k = i$ (Lines 5-6), Algorithm 4 performs the following:
(a) While $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ do (a1),(a2),(a3),(a4).
(a1) Update the current cone $[\theta, \theta']$: if x_k lies on the left of the current right border ($\theta_k > \theta$) then $\theta := \theta_k$ (new right border), if y_k lies on the right of the current left border ($\theta'_k < \theta'$) then $\theta' := \theta'_k$ (new left border).
(a2) $k := k + 1$ to involve the next tile, then, for the new k:
(a3) Determine $e_k = \beta_k \cap \beta_{k+1}$, its end points x_k, y_k, and the angles θ_k, θ'_k.
(a4) Update a candidate: if x_k lies on the left of (or on) the current right border ($\theta_k \geq \theta$) then set $p := x_k$ (concave), if y_k lies on the right of (or on) the current left border ($\theta'_k \leq \theta'$) then set $q := y_k$ (convex).
(b) If $[\theta_k, \theta'_k] \cap [\theta, \theta'] = \emptyset$, a new MPP vertex m is found: if x_k lies strictly on the right of the current right border ($\theta_k < \theta$) then set $m := p$ (concave), if x_k lies strictly on the left of the current right border ($\theta_k > \theta$) then set $m := q$ (convex).

Part (a) is the while-loop of Lines 8-16. Its condition is trivially true for $k = i$, hence the initial current cone remains as $[\theta, \theta'] = [\theta_i, \theta'_i]$. Steps (a3) and (a4) are performed for $k \geq i + 1$. The while-loop continues augmenting k to involve subsequent tiles, while the k-th cone $[\theta_k, \theta'_k]$ results well-defined and intersecting $[\theta, \theta']$.

Algorithm 4 distinguishes between convex and concave candidates, and in (a4), the last candidate p is not updated (as its corresponding angle) if x_k lies strictly on the right outside the current cone, similarly, q is not updated if y_k lies strictly on the left outside the cone. These actions correct strategy errors of Algorithm 2.

Errors in the correctness proof based on an incorrect Proposition. Although the strategy of the algorithm in [21] is mainly correct as analysed above, its correctness proof in Theorem 5.3 of [21] is not. The proof is based on Proposition 5.2 of [21] which is presented there with a non-clear proof. That proposition affirms that for any normal complex $\mathcal{C}$ in an acute polygonal mosaic, any polygon P which satisfies the following conditions 1), 2), 3), is the MPP of $\mathcal{C}$.
1) P is a preimage of $\mathcal{C}$ and contains the core of $\mathcal{C}$,
2) each concave vertex of P is a concave vertex of $fr(|\mathcal{C}|)$,
3) each convex vertex of P is a convex vertex of the core of $\mathcal{C}$, or, is a cut point point of $\mathcal{C}$.

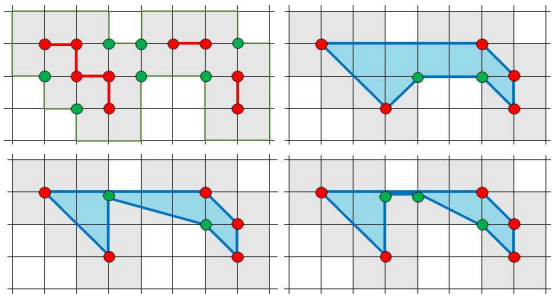

Fig. 18 A complex $\mathcal{C}$ (shaded grey) showing that Proposition 5.2 of [21] is false. First row: In the left figure, the core of $\mathcal{C}$ is drawn red with its convex vertices marked, concave vertices of $fr(|\mathcal{C}|)$ as green points, the right figure presents the MPP. The second row shows two polygons which are distinct from the MPP while satisfying all conditions of that proposition.

In the present article we consider the much more restricted conditions of rectangular mosaics and regular complexes (which have no cut points). Consider the example of a regular complex in the square mosaic in Figure 18 presenting polygons which are distinct from the MPP although they satisfy the conditions 1), 2), 3). Hence Proposition 5.2 of [21] is false.

7 Restriction to boundary chains generated by boundary tracing

Section 5.2 showed that some tools for the Sklansky-Chazin-Hansen Algorithm are not well-defined for any boundary chain of a regular complex in a rectangular mosaic, not even for a regular chain. A convenient more restrictive class contains all boundary chains generated by *boundary tracing*, also called *boundary following*, *contour tracing*, or *contour following*, which is a well-known method from digital image processing to find the frontier of a digital object.

Boundary tracing for 4-contours is described in detail in [28], for general mentions see textbooks as [4,12, 6]. Identifying the discrete plane $\mathbb{Z}^2$ with a quadratic mosaic (each pixel $p \in \mathbb{Z}^2$ with the unit square tile centered at p), boundary tracing to find the 4-contour of any finite 4-connected set of pixels can easily be reformulated to find the boundary (being a set of tiles) of the corresponding square complex. The method finds an ordered sequence of boundary tiles, forming a special type of boundary chain. This latter rely on the fact that adjacency for tiles in a rectangular mosaic is the same as 4-adjacency for square tiles, equivalently, for pixels in $\mathbb{Z}^2$. In consequence, boundary tracing can be generalized to rectangular mosaics in a straightforward manner, as follows.

Consider a rectangular mosaic $\mathcal{M}$, a regular complex $\mathcal{C} \subset \mathcal{M}$, and a finite set $\mathcal{N}$ with $\mathcal{C} \subset \mathcal{N} \subset \mathcal{M}$ and

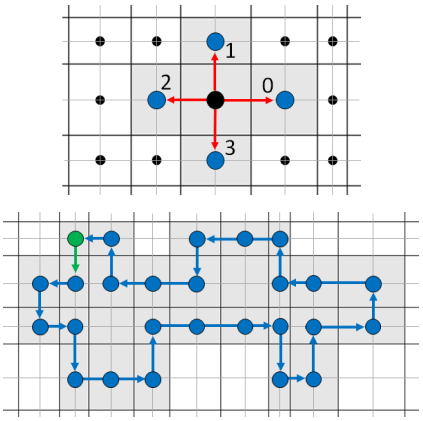

Fig. 19 Freeman code for edge-adjacency in a rectangular mosaic. Below: a complex (shaded grey) with its generic boundary chain, obtained by the boundary tracing algorithm. The boundary chain starting at the leftmost tile of the top row, has the Freeman chain code (3, 2, 3, 0, 3, 0, 0, 1, 0, 0, 0, 3, 0, 1, 0, 1, 2, 2, 1, 2, 2, 3, 2, 2, 1, 2).

such that the *background tiles* from $(\mathcal{N} \setminus \mathcal{C})$ fully surround the complex $\mathcal{C}$. The edge-adjacency graph $G(\mathcal{M})$ (see Section 4) has the finite induced subgraphs $G(\mathcal{N})$ and $G(\mathcal{C})$ where each node has a valence at most four. We apply the ***Freeman chain code*** known for 4-neighborhood graphs to represent any edge-adjacency-path in $\mathcal{N}$, see Figure 19: for any fixed tile $T \in G(\mathcal{N})$, the direction of passing from T to any of its edge-neighbors T', is coded by a number $f(T,T') \in \{0,1,2,3\}$. Whereas $f(T,T')$ describes the direction of passing from T to T', the reverse direction results as coded by $f(T',T) = (f(T,T')+2) \bmod 4$.

All tiles of a boundary chain $\beta(\mathcal{C})$ belong to the uniquely defined boundary $\mathcal{B}(\mathcal{C})$, they meet the frontier $fr(|\mathcal{C}|)$. To find such a chain $\beta(\mathcal{C})$, boundary tracing starts with determining a first tile of $\mathcal{B}(\mathcal{C})$. Due to the nature of a rectangular mosaic, this can easily be done by scanning $\mathcal{N}$ (or $G(\mathcal{N})$, interpreting each node as the centre point of the corresponding tile), which begins at some tile in the background $(\mathcal{N} \setminus \mathcal{C})$, until a first tile T_0 of $\mathcal{C}$ is found. Evidently $T_0 \in \mathcal{B}(\mathcal{C})$. The *boundary tracing algorithm* presented in Algorithm 5, uses one of the options, where T_0 is found such that its edge-neighbor on the left, belongs to the background, that is, $T_0 \in \mathcal{C}$ but if $f(T_0,q) = 2$ then $q \notin \mathcal{C}$. It determines a boundary chain given as ordered cyclic sequence of tiles $\beta(\mathcal{C}) = (T_0, T_1, T_2, \cdots, T_t)$ such that, when boundary tracing would be continued, the next tiles found would be, again, $T_{t+1} = T_0, T_{t+2} = T_1$.

Algorithm 5 finds a boundary chain $\beta(\mathcal{C}) = (T_0, T_1, \cdots, T_n)$ of special type. Step 2 will found T_0, again, when boundary tracing is finished, but it may happen also if the boundary chain touches itself at T_0, the end condition test distinguishes between both situations. The resulting boundary chain corresponds to a closed path in the edge-adjacency graph $G(\mathcal{N})$, $\beta(\mathcal{C})$ is an ordered cyclic sequence of tiles which can be described by its Freeman chain code $(f(T_0,T_1), f(T_1,T_2), f(T_2,T_3), \cdots, f(T_{n-1},T_n), f(T_n,T_0))$, Figure 19 shows an example. The following properties are evident.

Algorithm 5 Boundary tracing for an edge-adjacency-connected complex $\mathcal{C}$ in a rectangular mosaic.

Input: A finite subset $\mathcal{N}$ of a rectangular mosaic, a complex $\mathcal{C} \subset \mathcal{N}$ which is fully surrounded by tiles from the background $(\mathcal{N} \setminus \mathcal{C})$.
Output: List $\beta(\mathcal{C})$, a boundary chain of $\mathcal{C}$.

Step 0: Find a boundary tile $T_0 \in \mathcal{C}$ whose left edge-adjacency neighbor is in $(\mathcal{N} \setminus \mathcal{C})$, then go to Step 1.
Step 1: Initialize list $\beta(\mathcal{C}) := (T_0)$ and set $d := 2$. Then perform iteratively $d := (d+1) \bmod 4$. For each such d, analyse whether the tile T' that satisfies $f(T_0,T') = d$, belongs to $\mathcal{C}$. While this is false, d is augmented to continue searching for a tile of $\mathcal{C}$. If $T' \in \mathcal{C}$, add T' to $\beta(\mathcal{C})$, then go to Step 2.
Step 2: Set $d := (f(T_{n-1},T_n)+2) \bmod 4$ for the current list $\beta(\mathcal{C}) = (T_0, T_1, T_2, \cdots, T_n)$. Then perform iteratively $d := (d+1) \bmod 4$. For each such d analyse whether the tile T' with $f(T_n,T') = d$ belongs to $\mathcal{C}$. While $T' \notin \mathcal{C}$, d is increased to continue searching for a tile of $\mathcal{C}$. If $T' \in \mathcal{C}$, proceed to the End Condition Test.
End Condition Test:
if $T' \neq T_0$ **then** add T' to $\beta(\mathcal{C})$, then go to Step 2.
else (now $T' = T_0$) add T' to $\beta(\mathcal{C})$. Then run Step 2 with the current list $\beta(\mathcal{C})$ only to obtain a new next boundary tile $T' \in \mathcal{C}$. Then,
  if $T' = T_1$ **then** remove the last tile from the list $\beta(\mathcal{C})$, then STOP.
  else (now $T' \neq T_1$) add T' to $\beta(\mathcal{C})$, then go to Step 2.
  end if
end if

Lemma 2 *Let $\mathcal{C}$ be an edge-adjacency connected complex in a rectangular mosaic $\mathcal{M}$, $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ the boundary chain generated by Algorithm 5, denote by b_i the centre point of the tile β_i, $i = 0, 1, \cdots, t$. Then $\beta(\mathcal{C})$ satisfies the following:*

(1) The cyclic sequence $\beta(\mathcal{C})$ is unique.

(2) The polygonal curve $\gamma = \overline{b_0b_1} \cup \overline{b_1b_2} \cup \cdots \cup \overline{b_{t-1}b_t} \cup \overline{b_tb_0}$ is weakly simple and traced in counterclockwise sense. During this tracing, γ leaves all centre points of tiles of $\mathcal{C}$ on its left side (or on itself), but it leaves the centre points of tiles of $(\mathcal{M} \setminus \mathcal{C})$ strictly on the right.

(3) There do not exist tiles β_i, β_{i+1} in $\beta(\mathcal{C})$ such that both points b_i, b_{i+1} are concave vertices of the curve γ from (2).

(4) If $\mathcal{C}$ is regular then $\beta(\mathcal{C})$ is regular, that is, $\beta_{i-1} \neq \beta_{i+1}$ for all $0 \leq i \leq t$.

(5) If $\mathcal{C}$ is regular, when visiting the tiles β_i in the order as they appear in $\beta(\mathcal{C})$, the Jordan curve $fr(|\mathcal{C}|)$ is

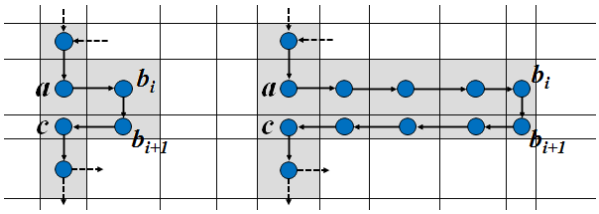

Fig. 20 A generic boundary chain does not contain two immediately consecutive concave turns, the boundary tracing algorithm finds a followed by its edge-neighbor c.

traced in counterclockwise sense, where it leaves $|\mathcal{C}|$ on the left and $(\mathbb{R}^2 \setminus |\mathcal{C}|)$ strictly on the right.

Proof (4) is true since $\mathcal{C}$ has no end tile. The remaining affirmations are evident from the Freeman code as defined in Figure 19 and the performance of Algorithm 5 which follows the boundary $\mathcal{B}(\mathcal{C})$ such that the chain $\beta(\mathcal{C})$ leaves $\mathcal{C}$ on the left, and the background $(\mathcal{M} \setminus \mathcal{C})$ strictly on the right. For (3), suppose tiles β_i, β_{i+1} in $\beta(\mathcal{C})$ such that their centre points b_i, b_{i+1} both are concave vertices of γ. This corresponds to a situation as those in Figure 20 where γ contains the subsequence (a, b_i, b_{i+1}, c), or, a longer one, with some linear points after a and before c. This is not possible for $\beta(\mathcal{C})$ since Algorithm 5 finds a followed by c.

□

Definition 5 For any edge-adjacency connected complex $\mathcal{C} \subset \mathcal{M}$ in a rectangular mosaic $\mathcal{M}$, its boundary chain determined by the boundary tracing Algorithm 5, is called the ***generic boundary chain*** of $\mathcal{C}$.

Algorithm 5 can be applied to any edge-adjacency connected complex $\mathcal{C}$. If $\mathcal{C}$ has end tiles, Algorithm 5 returns a well-defined (non-regular) boundary chain whose corresponding edge-adjacency subgraph has nodes of valence 1. Even if $|\mathcal{C}|$ has holes, Algorithm 5 can be adapted to find the "exterior boundary chain" and all other chains of boundary tiles which circumscribe the holes. The latter is common practice in digital image processing for 4-contours.

8 A new MPP algorithm for regular complexes in rectangular mosaics

8.1 Strategy, pseudocode, and examples

The new MPP algorithm presented in Algorithm 6, and equivalently in Algorithm 7, takes advantage from our analysis of the algorithms in the previous sections, it joints all correct ideas from the previous algorithms but avoids their errors. Algorithm 6 is inspired by the partially corrected Sklansky-Kibler algorithm presented in Algorithm 4 which essentially contains the correct strategy. Nevertheless, Algorithm 6 does not use a base line or reference vector and achieves an optimization by ignoring tiles which provide no MPP vertex candidates.

Algorithm 6 New algorithm to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic.

Input: Generic boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ (counterclockwise tracing) with its cyclic sequence of tile centre points $(b_0, b_1, \cdots, b_t)$.
Output: List *MPP* of MPP vertices.

1: Determine a convex MPP vertex $m = m_1$ as the lower right corner of the leftmost tile T_0 in the top row of $\mathcal{C}$. Perform cyclic shifting on $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ until β_0 becomes the tile T_0.
2: Add m_1 to the list *MPP*, set $i := 0$, $n := 1$.
3: **if** $i > t$ **then** STOP **endif**
4: Determine the end points x_i (right) and y_i (left) of the edge $e_i = \beta_i \cap \beta_{i+1}$.
5: **if** ($m \neq x_i$ and $m \neq y_i$ and b_{i-1}, b_i, b_{i+1} are not collinear) **then**
6: $p := x_i$ (concave, initial right cone border $\overrightarrow{mp}$)
7: $q := y_i$ (convex, initial left cone border $\overrightarrow{mq}$)
8: $k := i$
9: **while** (m, y_k, x_k form a right turn or are collinear, and x_k, y_k are not both on the same side strictly outside the cone) **do**
10: Update last concave candidate and right border: **if** (m, p, x_k form a left turn or are collinear) **then** ($p := x_k$ and $z_p := k$) **endif**
11: Update last convex candidate and left border: **if** (m, q, y_k form a right turn or are collinear) **then** ($q := y_k$ and $z_q := k$) **endif**
12: $k := k + 1$ (continue with the same cone)
13: **if** $k > t$ **then** STOP, **endif**
14: **if** b_{k-1}, b_k, b_{k+1} are collinear **then** go to Line 12 **endif** (ignore β_k)
15: Determine the end points x_k, y_k of $e_k = \beta_k \cap \beta_{k+1}$.
16: **end while**
17: (A new MPP vertex is found:)
18: **if** (m, p, x_k) forms a right turn, or are collinear, **then** ($m := p$ and $z := z_p$) **endif**
19: **if** (m, p, x_k) forms a left turn **then** ($m := q$ and $z := z_q$) **endif**
20: $n := n + 1$, add the point $m_n = m$ to the list *MPP*.
21: $i := z + 1$ (continue searching with β_{z+1})
22: Go to Line 3.
23: **end if**
24: $i := i + 1$
25: Go to Line 3.

Algorithm 6 requires as input data the generic boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ of a regular complex $\mathcal{C}$ in a rectangular mosaic, and generates the ordered list of MPP vertices. We suppose that the input chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ includes data about the vertices and the centre point b_i of each tile β_i. Then $\beta(\mathcal{C})$ corresponds to the polygonal curve γ represented by the cyclic sequence $(b_0, b_1, \cdots, b_{t-1}, b_t)$ of curve points,

and, it is easy to calculate the end points x_i, y_i of each edge $e_i = \beta_i \cap \beta_{i+1}$, where $x_i \in fr(|\mathcal{C}|)$.

Algorithm 6 determines in Line 1 a first special convex MPP vertex as proposed in the Sklansky-Chazin-Hansen algorithm (Algorithm 2) from [20].

Only concave vertices x_i of $fr(|\mathcal{C}|)$ and points y_i corresponding to convex vertices of γ are candidates for MPP vertices. Algorithm 6 in Lines 5 and 14 ignores each tile β_i where b_{i-1}, b_i, b_{i+1} are collinear (straight passing of the boundary), since such β_i provides no MPP vertex candidate. Algorithm 1 also ignores such tiles, but Algorithms 2, 3, 4 include them for angle calculus and the iterative restriction of the cone.

The while-condition $[\theta_k, \theta'_k] \cap [\theta, \theta'] \neq \emptyset$ in Line 8 of Algorithm 4 is equivalent to that the cone $[\theta_k, \theta'_k]$ is not empty and intersects the current cone $[\theta, \theta']$. The latter means that x_k, y_k do not lie both on the same side strictly outside $[\theta, \theta']$. Clearly $[\theta_k, \theta'_k] \neq \emptyset$ if and only if x_k lies on the right of, or on, $\overrightarrow{my_k}$, that is, m, y_k, x_k form a right turn or are collinear. Hence the while-conditions of Algorithm 4 and Algorithm 6 (Line 9) are equivalent.

Algorithm 4 (Line 9) updates the cone as follows:
– The right border is restricted or confirmed if x_k lies on the left of, or on, the current right border, which is equivalent to updating $p := x_k$ (last concave candidate) if m, p, x_k form a left turn or are collinear, this is Line 10 of Algorithm 6.
– The left border is restricted or confirmed if y_k lies on the right of, or on, the current left border, which is equivalent to updating $q := y_k$ (last convex candidate) if m, q, y_k form a right turn or are collinear, this is Line 11 of Algorithm 6.

Algorithm 4 (Lines 14,15) updates the candidates p, q but not the cone borders used in the while-condition. Our correction made in Section 6.2 was that for a new tile β_k, its angles first need to be analysed by the while-condition, before updating the cone borders. Our new algorithm does not use the angles, but the candidates themselves determine the cone borders and are updated after successful checking by the while-condition.

A new MPP vertex is found if the while condition in Line 9 is not satisfied, then the boundary performs at β_k an essential movement to the left, or to the right, which makes find a convex or concave new polygon vertex in Lines 18-19, given as the last candidate which defined, resp., the left or right current cone border.

Example 9: We apply Algorithm 6 to the complex studied before in Example 8 where the Sklansky-Kibler algorithm fails to detect the correct MPP vertices after y_{14}. Suppose $m = y_{14} = y_{15}$ to be the last found MPP vertex, see Figure 21. Algorithm 6 starts with $i = 16$, $m \neq x_{16}$, $m \neq y_{16}$, but β_{16} presents straight passing hence it is ignored. For $i = 17$, β_{17} provides an initial

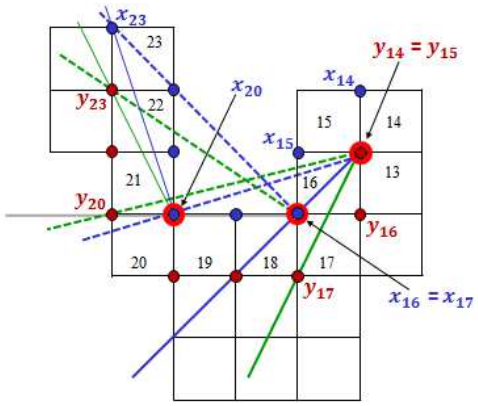

Fig. 21 The complex studied before in Example 8 where the Sklansky-Kibler algorithm fails, considered now in Example 9 working with the new Algorithm 6. Blue lines illustrate right cone borders, left borders are outlined as green. For $m = x_{17}$, the initial cone is degenerated to the ray $m - x_{20} - y_{20}$.

cone with $p = x_{17}$ (right border) and $q = y_{17}$ (left border). For $k = 17$, the while-condition is trivially true, Lines 10,11 do not change p and q. Then $k = 18$, but β_{18} and β_{19} are ignored. For $k = 20$, x_{20} lies on the left of $\overrightarrow{my_{20}}$, making the while-condition not satisfied. As x_{20} lies on the right outside the cone, Line 18 gives the next MPP vertex $m = p = x_{17}$.

Algorithm 6 continues with $m = x_{17}$, β_{18}, β_{19} are ignored. For $i = 20$, β_{20} provides an initial cone with $p = x_{20}$ (right border) and $q = y_{20}$ (left border). That cone is a ray since m, x_{20}, y_{20} are collinear, see the grey line in Figure 21. Then $k = 21$, but β_{21}, β_{22} are ignored. For $k = 23$, x_{23} lies on the right of $\overrightarrow{my_{23}}$, but x_{23}, y_{23} both lie on the right outside the cone hence the while-condition is not fulfilled, and Line 18 provides the next MPP vertex $m = p = x_{20}$.

Proceeding with $m = x_{20}$, β_{21} and β_{22} are ignored, a cone is initialized by $p = x_{23}$ (right border) and $q = y_{23}$ (left border). Augmenting k in the next step leads to $k = 24 > t$, the algorithm stops and returns a list which ends with the correct MPP vertices y_{14}, x_{17}, x_{20}.

Example 10: We apply Algorithm 6 to the complex boundary studied before in Example 1. We are interested only in the part between the MPP vertices m (called V_L in Example 1) and s, therefore, let the boundary tiles β_i be numbered as shown in Figure 22. The point g is the unique MPP vertex between m and s, this cannot be obtained by Algorithm 1.

Algorithm 6 starts from m with β_1 ($i = 1$), $x_1 \neq m$, $y_1 \neq m$ but β_1, β_2, β_3 are ignored since these tiles present straight passing of the boundary. The initial cone is determined from $i = 4$ by $p = x_4$ (right border $\overrightarrow{mp}$) and $q = y_4$ (left border $\overrightarrow{mq}$). Advancing with $k = 5$, $p = x_5 = x_4$ confirms the right border, y_5 is ignored since it lies outside the cone on the left. Tiles β_6, β_7,

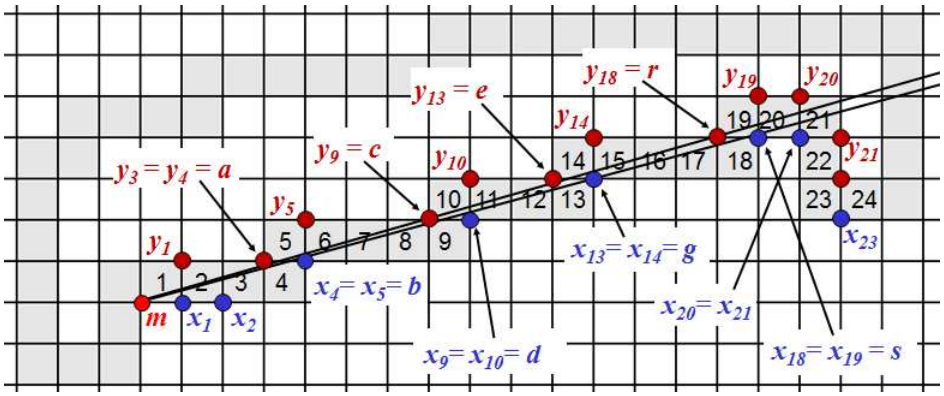

Fig. 22 Part of the 4-contour from Figure 2, to be processed by Algorithm 6 in Example 10. Thin black lines illustrate the cone with left border determined by c and r and right border defined by g, the points m, g, s are consecutive MPP vertices.

β_8 are ignored (straight passing). For $k = 9$, $p = x_9$ confirms the right border, y_9 lies inside the cone, $q = y_9 = c$ restricts the cone by defining a new left border (slope 2/7). For $k = 10$, $p = x_{10} = x_9$ confirms the right border, y_{10} is ignored since it lies outside the cone on the left. Tiles β_{11}, β_{12} are ignored. For $k = 13$, $p = x_{13} = g$ restricts the cone by a new right border (slope 3/11), but $y_{13} = e$ is ignored since it lies on the left outside the cone. Summarizing the next steps, $p = x_{14}$ confirms the right border, y_{14}, β_{15}, β_{16}, β_{17}, x_{18} all are ignored, $q = y_{18}$ confirms the left border $\overrightarrow{my_9} = \overrightarrow{mc}$. For $k = 19$, both x_{19} and y_{19} are ignored, the same for x_{20}, y_{20}. For $k = 21$, x_{21} lies strictly on the left of $\overrightarrow{my_{21}}$, the while-condition is not satisfied, Line 17 is reached: since x_{21} lies strictly on the right of the right border given by $p = x_{14} = g$, the next MPP vertex $m = p = g$ is correctly found. It is easy to see that, starting with $m = g$, Algorithm 6 finds the next MPP vertex $s = x_{19}$.

Algorithm 6 corrects the error of Example 1 thanks to the correct strategy that concave candidates x_k which lie strictly on the right of the cone, and convex candidates y_k which lie strictly on the left of the cone, are ignored for updating the cone borders. This strategy is also contained in the Sklansky-Kibler algorithm, but, it is not part of the other algorithms studied in the previous sections. The partially corrected Sklansky-Kibler algorithm (Algorithm 4) also determines the correct MPP vertices for this example.

Example 11: We apply Algorithm 6 to the complexes of Figure 3 studied before in Example 2 where Algorithm 1 found erroneous MPP vertices. Consider the smallest complex (d) where the points 1,6,8 are the MPP vertices. Starting at point 1, Algorithm 6 finds a cone with left border defined by $q = 4$ (slope 1) and right border given by $p = 5$ (slope 2/3). For the next tile, $y = 6$ lies in the cone hence a new left border is defined by $q = 6$ (slope 3/4). In distinction to Example 2, $x = 7$ now is ignored since it lies outside the cone on the right. For the next tile, $y = 8$ (outside the cone on the left) and $x = 7$ (outside the cone on the right) both are ignored. Then $y = 8$ is ignored but $x = s$ confirms the right border, see Figure 3, $p = x = s$ is updated. The next tile provides $y = 9$ and $x = n$, x lies on the right of the ray $\overrightarrow{1, y}$ but also outside the cone on the left since $\overrightarrow{1, n}$ has slope 5/6 larger than 3/4. The while-condition of Algorithm 6 is not fulfilled, hence the next MPP vertex $m = q = 6$ is correctly found.

The MPP for the complex (a) has the unique MPP vertex 18 between the MPP vertices 1 and 20, which is detected by Algorithm 6 as follows: a cone with left border determined by $q = 6$ (slope 3/4) and right border given by $p = 5$ (slope 2/3) is obtained, $x = 7$ is ignored. In the next steps, $y = 8$ and $x = 8$ both are ignored, and $x = 9$ confirms the right border with $p = x = 9$. Then, $x = 11$ is ignored, but $y = 10$ lies inside the cone since $\overrightarrow{1, 10}$ has slope $5/7 \in (2/3, 3/4)$, hence $q = y = 10$ defines a new left border with slope 5/7. Continuing, $y = 12$, $x = 11$ both are ignored, $x = 13$ confirms the right border with $p = x = 13$. Then, $x = 15$ is ignored, but $y = 14$ lies inside the cone, $\overrightarrow{1, 14}$ has slope $7/10 \in (2/3, 5/7)$, $q = y = 14$ defines a new left border with slope 7/10. Later, $y = 16$, $x = 15$ both are ignored, $x = 17$ confirms the right border with $p = x = 17$. Then, $x = 19$ is ignored, but $y = 18$ lies inside the cone since $\overrightarrow{1, 18}$ has slope $9/13 \in (2/3, 7/10)$, hence $q = y = 18$ defines a new left border with slope 9/13. As next, $y = 20$ and $x = 19$ are ignored, $x = s$ confirms the right border with $p = x = s$. The next tile provides $y = 21$ and $x = n$, x lies on the right of the ray $\overrightarrow{1, y}$ but also outside the cone on the left since $\overrightarrow{1, n}$ has slope 11/15 larger than 9/13. The while-condition of Algorithm 6 is not satisfied, the next MPP vertex is correctly found as $m = q = 18$.

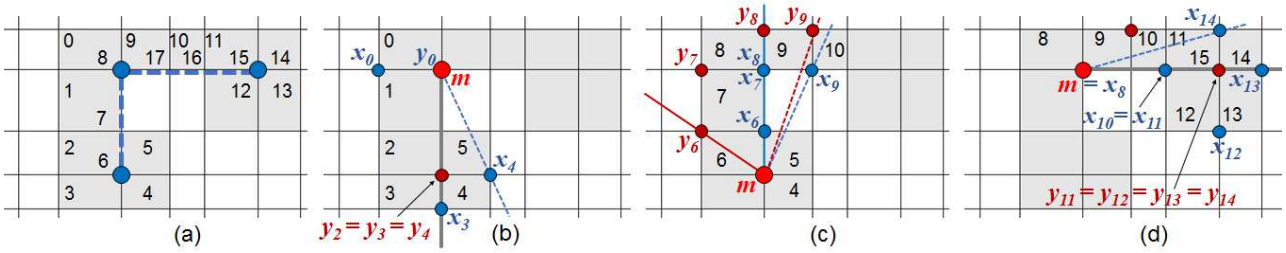

Fig. 23 A complex $\mathcal{C}$ with a thin part near the tile β_0, $\mathcal{C}$ coincides with its boundary. (a) Tiles numbering due to the generic boundary chain, the MPP of $\mathcal{C}$ is a weakly simply polygon with vertex sequence $(y_0, y_3, x_8 = y_0, y_{13})$ drawn with blue broken lines. (b) cone illustration for Algorithm 6 starting with $m = y_0$, for $m = y_3$ in (c), for $m = x_8$ in (d).

Example 12: We apply Algorithm 6 to the complex of rectangles in Figure 23. Starting with $m = y_0$, $i = 1$, β_1 and β_2 are ignored. For $i = 3$, β_3 provides an initial cone with $p = x_3$ defining its right border and $q = y_3$ giving its left border, that cone is a ray drawn as bold grey line in Figure 23(b). For $k = 4$, the while-condition is not true, x_4 lies on the left of $\overrightarrow{my_4}$, and of the cone, the next MPP vertex is found as $m = q = y_3$.

Continuing with $m = y_3$, see Figure 23(c), $y_4 = y_5 = m$ hence $i = 6$ provides an initial cone with $p = x_6$ (right border) and $q = y_6$ (left border). For $k = 7$, β_7 is ignored, $k = 8$ gives $p = x_8$, $q = y_8$, the cone has become a ray. For $k = 9$, x_9, y_9 both lie on the right outside the cone, the while-condition is not fulfilled, Line 18 provides the next MPP vertex $m = p = x_8$.

Proceeding with $m = x_8$, β_9, β_{10} are ignored, an initial cone is found from $i = 11$, $p = x_{11}$, $q = y_{11}$ (left border), the cone is a ray, see Figure 23(d). For $k = 12$, x_{12} is ignored since it lies on the right outside the cone, $q = y_{12}$ confirms the left border. For $k = 13$, $p = x_{13}$ and $q = y_{13}$ both confirm the cone. For $k = 14$, x_{14} lies on the left of $\overrightarrow{my_{14}}$ and of the cone, the while-condition is not true, we find the next MPP vertex as $m = q = y_{13}$. Finally, for $m = y_{13}$, β_{14} ($m = y_{14}$), β_{15}, β_{16}, β_{17} all are ignored. We get $i = 18$ which exceeds t (Line 3), hence Algorithm 6 stops and delivers the correct sequence (y_0, y_3, x_8, y_{13}) of MPP vertices.

The Sklansky-Kibler algorithm correctly handles the starting situation of this example, but later on, wrong angle calculus causes errors: from $m = y_3$, since $m = y_4 = y_5$ we reach $k = i = 6$ where x_6 defines a new base line (and right cone border), y_6 defines the left border. For $k = 7$, $p = x_7$ lies on the cone border, $q = y_7$ restricts the cone. For $k = 8$, $p = x_8 = x_7$ confirms the right border, $q = y_8$ restricts the left border, the cone has become a ray, $\theta' = \theta$. For $k = 9$, although the cone $[\theta_9, \theta'_9]$ is well-defined, it does no intersect the cone $[\theta, \theta']$, the while-condition is not satisfied. Since $\theta_9 > 0 = \theta$ (which does not correctly describe that x_9 lies on the right outside the cone), Algorithm 4 gets $m = q = y_8$ as next MPP vertex which is false.

We will see in Section 8.4 that the Sklansky-Chazin-Hansen algorithm cannot process the (generic boundary chain of the regular) complex from Figure 23 since the sets u_1, w_1 are not well-defined.

8.2 An MPP algorithm equivalent to Algorithm 6

Recall from Section 2 that in any right-hand Cartesian coordinate system, the sign of the determinant $D(p_1, p_2, p_3)$ indicates the orientation of the triple of points: $D(p_1, p_2, p_3) < 0 \iff (p_1, p_2, p_3)$ forms a right turn, $D(p_1, p_2, p_3) > 0 \iff (p_1, p_2, p_3)$ forms a left turn, $D(p_1, p_2, p_3) = 0 \iff p_1, p_2, p_3$ are collinear.

Using this, Algorithm 6 can be rewritten as Algorithm 7: the definition of that determinant implies that the corresponding Lines 5, 10, 11, and 14, of both algorithms are equivalent. In the while condition of Algorithm 6 (Line 9), m, y_k, x_k form a right turn or are collinear if and only if $D(m, y_k, x_k) \leq 0$. Supposing that this is satisfied,

– x_k, y_k both lie strictly outside the cone on its left side if and only if $D(m, q, x_k) > 0$;

– x_k, y_k both lie strictly outside the cone on its right side if and only if $D(m, p, y_k) < 0$.

The while condition negates these facts, it yields that $(D(m, q, x_k) > 0$ or $D(m, p, y_k) < 0)$ is false if and only if $(D(m, q, x_k) \leq 0$ and $D(m, p, y_k) \geq 0)$ is true, hence Lines 9 of Algorithms 6 and 7 are equivalent.

8.3 Correctness proof of Algorithm 6

Definition 6 Let $\mathcal{C}$ be a regular complex in a rectangular mosaic and $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ a boundary chain of $\mathcal{C}$. For $i \in \{0, 1, \cdots, t\}$, denote by b_i the centre point of the tile β_i, and let γ be the curve $\gamma = \overline{b_0 b_1} \cup \overline{b_1 b_2} \cup \cdots \cup \overline{b_{t-1} b_t} \cup \overline{b_t b_0}$. For any transversal edge $e_i = \beta_i \cap \beta_{i+1}$ (including $e_t = \beta_t \cap \beta_0$), denote by x_i the right (with respect to $\overrightarrow{b_i b_{i+1}}$) end point of e_i, and by y_i its left end point. The ***candidate list*** *Cand(C)* $= (p_1, p_2, \cdots, p_s)$ is constructed as follows:

Algorithm 7 Algorithm to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic, equivalent to Algorithm 6, using the determinant describing the orientation of a point triple.

Input: Generic boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ (counterclockwise tracing) with its cyclic sequence of tile centre points $(b_0, b_1, \cdots, b_t)$.
Output: List *MPP* of MPP vertices.

1: Determine a convex MPP vertex $m = m_1$ as the lower right corner of the leftmost tile T_0 in the top row of $\mathcal{C}$. Perform cyclic shifting on $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ until β_0 becomes the tile T_0.
2: Add m_1 to the list *MPP*, set $i := 0$, $n := 1$.
3: **if** $i > t$ **then** STOP **end if**
4: Determine the end points x_i (right) and y_i (left) of the edge $e_i = \beta_i \cap \beta_{i+1}$.
5: **if** ($m \neq x_i$ and $m \neq y_i$ and $D(b_{i-1}, b_i, b_{i+1}) \neq 0$) **then**
6: $p := x_i$ (concave, initial right cone border $\overrightarrow{mp}$)
7: $q := y_i$ (convex, initial left cone border $\overrightarrow{mq}$)
8: $k := i$
9: **while** ($D(m, y_k, x_k) \leq 0$ and $D(m, q, x_k) \leq 0$ and $D(m, p, y_k) \geq 0$) **do**
10: Update last concave candidate and right border: **if** $D(m, p, x_k) \geq 0$ **then** ($p := x_k, z_p := k$) **endif**
11: Update last convex candidate and left border: **if** $D(m, q, y_k) \leq 0$ **then** ($q := y_k, z_q := k$) **endif**
12: $k := k + 1$ (continue with the same cone)
13: **if** $k > t$ **then** STOP **endif**
14: **if** $D(b_{k-1}, b_k, b_{k+1}) = 0$ **then** go to Line 12 **endif**
15: Determine the end points x_k, y_k of $e_k = \beta_k \cap \beta_{k+1}$.
16: **end while**
17: (A new MPP vertex is found:)
18: **if** $D(m, p, x_k) \leq 0$ **then** ($m := p$, $z := z_p$) **endif**
19: **if** $D(m, p, x_k) > 0$ **then** ($m := q$, $z := z_q$) **endif**
20: $n := n + 1$, add the point $m_n = m$ to the list *MPP*.
21: $i := z + 1$ (continue searching with β_{z+1})
22: Go to Line 3.
23: **end if**
24: $i := i + 1$
25: Go to Line 3.

For each $i \in \{0, 1, \cdots, t\}$,

– if b_i is a concave vertex of γ then add the point x_i to the list *Cand(C)*.

– if b_i is a convex vertex of γ then add the point y_i to the list *Cand(C)*.

Each point p_j of *Cand(C)* is called ***candidate***, denote by $ind(p_j)$ the index i in the list $\beta(\mathcal{C})$, of the tile β_i which provided p_j.

The list *Cand(C)* does not contain points x_i, y_i provided by boundary tiles of straight passing, hence *Cand(C)* may result shorter than $\beta(\mathcal{C})$, and the index j of a point p_j in *Cand(C)* may be smaller than the index $i = ind(p_j)$ of the tile that provided p_j, in $\beta(\mathcal{C})$.

The following two lemmas join properties of the generic boundary chain determined by Algorithm 5, which is used as input list in Algorithm 6.

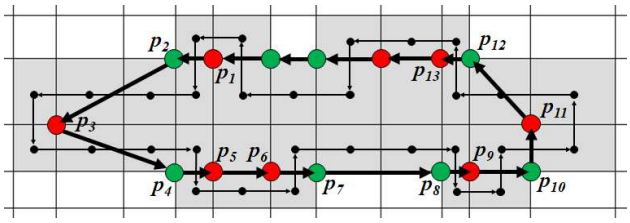

Fig. 24 A complex $\mathcal{C}$ (shaded grey) with its generic boundary chain depicted by small black discs and thin black arrows. The list *Cand(C)* is visualized by thick black arrows, convex candidates (red discs) are convex vertices of the core, concave candidates (green discs) are concave vertices of $fr(\mathcal{C})$. In this example, the MPP of $\mathcal{C}$ is given by the vertex sequence $(p_1, p_3, p_5, p_9, p_{11}, p_{13})$.

Lemma 3 *Let $\mathcal{C}$ be a regular complex in a rectangular mosaic, $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ its generic boundary chain, and Cand(C)= $(p_1, p_2, \cdots, p_s)$ its candidate list.*
(1) If β_i, β_{i+k}, $k \geq 1$, generate successive candidates p_j, p_{j+1} in Cand(C) then $\overline{p_j p_{j+1}} \subset \beta_i \cup \beta_{i+1} \cup \cdots \cup \beta_{i+k}$.
(2) The curve $\delta = \overline{p_1 p_2} \cup \overline{p_2 p_3} \cup \cdots \cup \overline{p_{s-1} p_s} \cup \overline{p_s p_1}$ is traced in counterclockwise sense and goes through all tiles of $\beta(\mathcal{C})$ due to their order in that list, δ is contained in the point set union of the boundary $\mathcal{B}(\mathcal{C})$.
(3) The curve δ is weakly simple, it represents the frontier of a weakly simple polygon P which is a preimage of $\mathcal{C}$ and satisfies that core($\mathcal{C}$) $\subset P \subset |\mathcal{C}|$. Meanwhile P is circumscribed, P lies on the left of (or on) δ, and $(\mathbb{R}^2 \setminus P)$ lies strictly on the right of δ.
(4) Let $= (q_1, q_2, \cdots, q_n)$ be a subsequence of Cand(C), $n \leq s$, that is, for each $j \in \{1, 2, \cdots n\}$, there exists $i \in \{1, 2, \cdots, s\}$ and $k \geq 1$ such that $q_j = p_i, q_{j+1} = p_{i+k}$. If each segment $\overline{q_j q_{j+1}}$ is contained in $\beta_{ind(p_i)} \cup \beta_{ind(p_i)+1} \cup \beta_{ind(p_i)+2} \cup \cdots \cup \beta_{ind(p_{i+k})-1} \cup \beta_{ind(p_{i+k})}$ then the curve $\lambda = \overline{q_1 q_2} \cup \overline{q_2 q_3} \cup \cdots \cup \overline{q_{n-1} q_n} \cup \overline{q_n q_1}$ has the same properties (2),(3) as δ.

Proof All properties are consequences of the performance of Algorithm 5 and of Lemma 2.
(1): suppose that b_i, b_{i+k}, $k \geq 1$, generate successive candidates p_j, p_{j+1}. For $k = 1$, $\overline{p_j p_{j+1}}$ consists in a rectangle side belonging to $\beta_i \cup \beta_{i+1}$, for example, as $\overline{p_1 p_2}$ and $\overline{p_4 p_5}$ in Figure 24.

Assume now $k \geq 2$, then $b_i, b_{i+1}, \cdots, b_{i+k-1}, b_{i+k}$ are collinear. If b_i, b_{i+k} both are concave, or, both are convex, $\overline{p_j p_{j+1}}$ is parallel to a coordinate axis since it is the union of collinear tile sides, for example, as $\overline{p_7 p_8}$ (concave) and $\overline{p_5 p_6}$ (convex) in Figure 24). If one of the points b_i, b_{i+k} is convex and the other one is concave, $\overline{p_j p_{j+1}}$ is the diagonal of a rectangle which is the union of consecutive tiles, for example, as $\overline{p_{11} p_{12}}$ and $\overline{p_2 p_3}$ in Figure 24. In any case, $\overline{p_j p_{j+1}}$ lies in $\beta_i \cup \beta_{i+1} \cup \cdots \cup \beta_{i+k}$.

(2),(3): As consequence of (1), δ is contained in the point set union of $\mathcal{B}(\mathcal{C})$ since for all $j = 1, \cdots, s-1$,

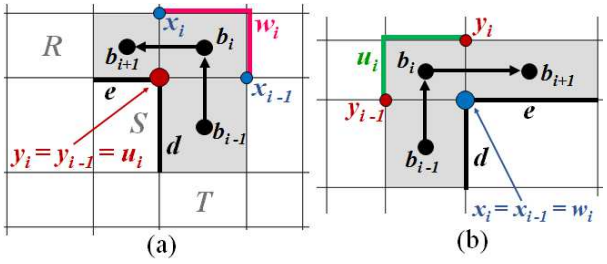

Fig. 25 Convex turn (a) and concave turn (b) of a boundary chain, considered in the proof of Lemma 4.

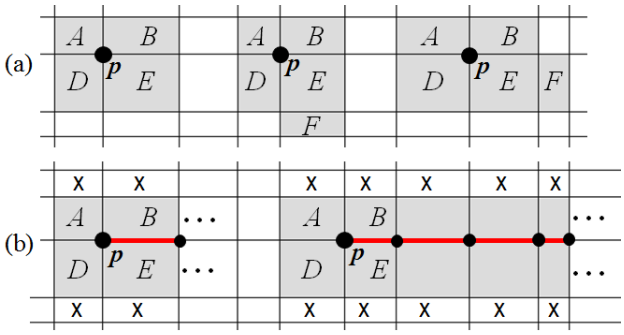

Fig. 26 (a) In a complex containing the tiles A, B, D, E, p is an isolated point of the core, considered in the proof of Lemma 4. (b) In a complex containing the tiles shaded grey, p is the end point of a straight line segment belonging to $fr(core(\mathcal{C}))$ drawn red, that line may contain two o more vertices of tiles.

$\overline{p_j p_{j+1}}$, and also $\overline{p_s p_1}$, belong to $\beta_i \cup \beta_{i+1} \cup \cdots \cup \beta_{i+k}$ for $ind(p_j) = i$ and $ind(p_{j+1}) = i + k$. Moreover, the generic boundary chain circumscribes the whole complex, always having the tiles of $\mathcal{C}$ on its left side, or on itself, and leaving the complement of $\mathcal{C}$ strictly on the right. Hence δ goes through all boundary tiles, visiting them in the order due to the list $\beta(\mathcal{C})$, and encloses a weakly simple polygon P which contains all other tiles of $\mathcal{C}$. In consequence, P is a preimage of $\mathcal{C}$, contains $core(\mathcal{C})$, and is contained in $|\mathcal{C}|$.
(4): the explicit supposition ensures the same properties of δ, for the curve λ.

□

Lemma 4 *Let $\mathcal{C}$ be a regular complex in a rectangular mosaic and $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ its generic boundary chain, Cand(C) its candidate list, b_i the centre point of β_i, and $\gamma = \overline{b_0 b_1} \cup \overline{b_1 b_2} \cup \cdots \cup \overline{b_{t-1} b_t} \cup \overline{b_t b_0}$. Let x_i be the right (with respect to $\overrightarrow{b_i b_{i+1}}$) end point of $e_i = \beta_i \cap \beta_{i+1}$, and y_i its left end point.*
(1) If b_i is a concave vertex of γ then $x_i = x_{i-1}$ is a concave vertex of $fr(|C|)$.
(2) If $\beta(\mathcal{C})$ is non-repeating and b_i is a convex vertex of γ then $y_i = y_{i-1}$ is a convex vertex of core($\mathcal{C}$).
(3) Each vertex of the MPP of $\mathcal{C}$ is a point $x_i \in e_i$ where b_i is a concave vertex of γ, or, a point $y_i \in e_i$ where b_i is a convex vertex of γ.
(4) All vertices of the MPP of $\mathcal{C}$ are contained in the candidate list Cand(C).

Proof .
(1) If b_i is a concave vertex of γ, it belongs to a situation as in Figure 25(b). The edges d and e lie on $fr(|\mathcal{C}|)$ since $\beta_{i-1}, \beta_i, \beta_{i+1}$ belong to a generic boundary chain (which leaves the background on the right), hence $x_i = x_{i-1}$ is a concave vertex of $fr(|\mathcal{C}|)$.

(2) The hypothesis means that $\mathcal{C}$ has no thin parts where the boundary chain passes twice. If b_i is a convex vertex of γ, it belongs to a situation as in Figure 25(a), where $\beta_{i-1}, \beta_i, \beta_{i+1} \in \beta(\mathcal{C})$. Since $\beta(\mathcal{C})$ is a closed edge-adjacency path, from β_{i+1}, it continues until reaching β_{i-1}, again. That path could not start at β_{i+1} and then curve clockwise up to β_{i-1}, since then the curve $fr(|\mathcal{C}|)$ would enclose a hole of $|\mathcal{C}|$, which contradicts that $\mathcal{C}$ is regular and hence $fr(|\mathcal{C}|)$ is a Jordan curve. In consequence, $\beta(\mathcal{C})$ goes from β_{i+1}, curving counterclockwise until reaching β_{i-1}. Since $\beta(\mathcal{C})$ does not repeat tiles and does not circumscribe a hole of $|\mathcal{C}|$, the tile S belongs to $\mathcal{C}$. It is easy to see that then, in all cases for the tiles R, T, to belong to $\mathcal{C}$, or not, y_i is an isolated point of $core(\mathcal{C})$, or, is the end point of one of the edges d, e which then is a thin part of $core(\mathcal{C})$. In each situation, y_i is a convex vertex of $core(\mathcal{C})$.

(4) is an immediate consequence of (3). To see (3), first, let p be a convex MPP vertex. By Lemma 1 (known from [20,21,24]), then p is a convex vertex of $fr(core(\mathcal{C}))$. There are three possible situations:

– If p is an isolated point of $core(\mathcal{C})$, each one of the four edges starting at p does not fully belong to $fr(core(\mathcal{C}))$, hence its other end point lies in $fr(|\mathcal{C}|)$, see Figure 26(a). Then the tiles A, B, D, E belong to the boundary of $\mathcal{C}$. If $\mathcal{C}$ does not contain more tiles, the cyclic sequence $\beta(\mathcal{C}) = (D, E, B, A)$ is the generic boundary chain. Then p may be detected as y_i due to (2) taking $\beta_i = B$ since B corresponds to a left turn of γ. If $\mathcal{C}$ contains some other tile F, F is an edge-neighbour of one of the tiles A, B, D, E, without lost of generality, assume of E, see Figure 26(a). Then $\beta(\mathcal{C})$ contains the subsequence (F, E, B, A, D), p may be detected due to (2) from $\beta_i = B$, for example.

– If p is the end point of a straight line segment being a thin part of $core(\mathcal{C})$, the situation is as in Figure 26(b), note that the tiles marked X do not belong to $\mathcal{C}$. In any case, $\beta(\mathcal{C})$ contains the subsequence (B, A, D, E) which permits to detect p due to (2), taking A or D as β_i. The situation near p is the same if p is the end point of a curve segment being a thin part of $core(\mathcal{C})$, since such

a curve is made up from rectangle sides and ends at p with a straight line segment.
– If p is neither an isolated point of the core, nor the end point of some thin part of the core, then p is a convex vertex of $fr(core(\mathcal{C}))$ as the point y_i in Figure 25(a) where two edges starting at p belong to the core, but the other two edges reach $fr(|\mathcal{C}|)$. Hence the point p bounds tiles $\beta_{i-1}, \beta_i, \beta_{i+1}$ as in the figure, and p may be detected due to (2) from β_i via the convex vertex b_i.

If p is a concave MPP vertex then p is a concave vertex of $fr(|\mathcal{C}|)$, as known from [20,21,24]. In consequence, the situation near p is as in Figure 25(b) where p bounds tiles $\beta_{i-1}, \beta_i, \beta_{i+1}$ such that p may be detected due to (1) from β_i via the concave vertex b_i of γ. This completes to prove (3).

□

Algorithm 6 works with the visibility cones, the ***current cone*** given by (m, p, q) is rooted at the last found polygon vertex m, $\overrightarrow{mp}$ is its right border, $\overrightarrow{mq}$ the left border. It is firstly generated as the ***initial cone*** in Lines 6-7: $p = x_i, q = y_i$. Then, each β_k taken from the input list, provides a ***k-th cone*** given by (m, x_k, y_k) (right border $\overrightarrow{mx_k}$, left border $\overrightarrow{my_k}$). The condition in Line 9 requires that the k-th cone is well-defined, which means that m, x_k, y_k forms a left turn, or, are collinear. If all conditions of Line 9 are satisfied, the k-th cone is compared with the current cone, as result, the cone borders eventually are updated in Lines 10-11.

Lemma 5 *For any regular complex $\mathcal{C}$ in a rectangular mosaic, and its generic boundary chain of $\mathcal{C}$ as input list, the visibility cones used in Algorithm 6 satisfy the following:*
(1) Each initial cone given by (m, x_i, y_i) defined in Lines 6-7, is well-defined. The angle given as $\alpha = \sphericalangle(\overrightarrow{mx_i}, \overrightarrow{my_i})$ satisfies $0° \leq \alpha < 90°$.
(2) Each current cone given by (m, p, q), obtained by eventual updating in Lines 10-11, is well-defined, and, is a subset of the initial cone.
(3) Situating m at the origin of the plane, the initial cone belongs to exactly one quadrant of the plane, or, is a ray lying on the x- or y-axis.
(4) Suppose that Algorithm 6 in Line 9 finds a k-th cone which is not well-defined, that is, (m, x_k, y_k) forms a right turn. Then, both points x_k, y_k lie on the same side strictly outside the current cone, or, x_k lies strictly outside the current cone on the left and y_k lies on the ray $\overrightarrow{mq}$, or, y_k lies strictly outside the current cone on the right and x_k lies on the ray $\overrightarrow{mp}$.
(5) If $\mathcal{C}$ belongs to a quadratic mosaic, the initial cone forms an angle α such that $0° \leq \alpha \leq 45°$.

Proof Inspecting all possible initial cones generated by Algorithm 6 in Figures 27 and 28, confirms the facts stated in (1) and (3). The angle formed by the initial cone depends on whether there are collinear tiles starting with β_{z+1} or not, but also on the proportion between width and height of the appearing tiles. The figures make evident that the largest possible cone angle is of 45° if all tiles are squares, which confirms (5). The initial cone, which is the first current cone, is well-defined by construction, x_i determines the right border and y_i the left border. Each current cone resulting from updating in Lines 10-11, is well-defined because updating only is performed if the while condition of Line 9 is satisfied. That condition also guarantees that updating confirms or restricts the current cone, so, each time, the current cone remains the same, or, becomes a smaller subset, this confirms (2).

To prove (4), suppose that Algorithm 6 has determined a current cone given by (m, p, q), which had been initialized by x_i, y_i from $\beta_i \cap \beta_{i+1}$. Now, in Line 9, k was found where (m, x_k, y_k) forms a right turn but for all $t \in \{i, i+1, \cdots, k-1\}$, the t-th cone is well-defined ((m, x_t, y_t) is a left turn, or they are collinear).

Assuming m as situated at the origin of the plane, by (3), the current cone lies in exactly one quadrant of the plane, or, is a ray lying on some coordinate axis. Up to rotation by multiples of 90°, all possible initial cones are shown in Figures 27, 28. Since $\mathcal{C}$ belongs to a rectangular mosaic, the movement from β_k to β_{k+1} may be only in directions up, to the right, down, and to the left, and the edge $\beta_k \cap \beta_{k+1}$ is parallel to a coordinate axis. The current cone goes through all tiles from the tile which provided the point m, until β_k.

First consider a current cone belonging exactly to the first quadrant, the discussion for other quadrants would be similar. Then the cone looks like those of Figure 29 where (a)(b) show that any movement from β_k to β_{k+1} in directions up, or, to the right, generates a well-defined k-th cone, although x_k or y_k may lie outside the current cone. In contrast, any movement from β_k to β_{k+1} to the left, or down, causes that (m, x_k, y_k) forms a right turn. Moreover, the initial cone started with movements to the right and up (or, vice versa), see Figures 27(b)(g)(j) and 28(b)(c)(e). Therefore, a movement to the left is only possible if β_k presents a left turn (β_{k-1} lies below β_k), and a movement down needs β_k to present a right turn (β_{k-1} lies on the left of β_k), where β_{k-1} also intersects the current cone, see Figure 29(c)(d). By the updating rules in Lines 10-11, when Algorithm 6 reaches β_k, y_{k-1} defines the left cone border, or lies on the left outside the cone. Likewise, x_{k-1} defines the right cone border, or lies on the right outside the cone. For β_k being a left turn, see Figure 29(c),

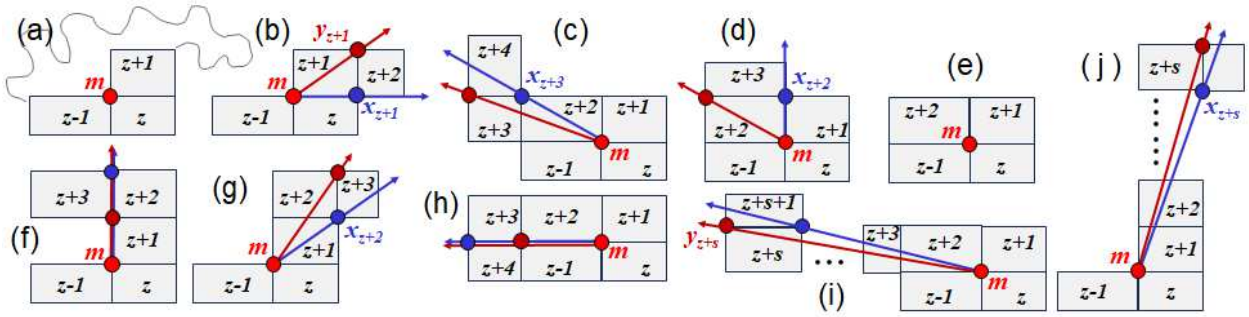

Fig. 27 All possible initial cones generated by Algorithm 6 if the last found polygon vertex m is convex, up to rotations by multiples of 90°. Notations as z within a tile mean β_z. The algorithm found m as vertex of β_z, m is a vertex of $core(\mathcal{C})$, the curve drawn in (a) illustrates the boundary of $\mathcal{C}$. From β_{z+1}, the boundary chain may continue only to the right, up, or, to the left, since $\mathcal{C}$ has no end tiles. If β_{z+2} lies on the right of β_{z+1}, see (b), β_{z+1} is a candidate (right turn), hence x_{z+1}, y_{z+1} determine the initial cone. If β_{z+2} lies above β_{z+1}, β_{z+1} presents straight passing and hence is ignored. If then β_{z+2} presents a turn which may be to the left (f), or to the right (g), x_{z+2}, y_{z+2} define the initial cone. But β_{z+2} may present straight passing as well as other tiles being all collinear, until some β_{z+s} presents a right (j) or left turn. The latter case is not drawn, the initial cone would be degenerated to a ray. If β_{z+2} lies on the left of β_{z+1}, β_{z+1} is ignored since $y_{z+1} = m$, β_{z+2} offers possible continuation to the left (c), up (d), or, down (e). In (e), $\beta_{z+3} = \beta_{z-1}$, no initial cone is generated, the algorithm ends returning m as unique polygon vertex, which is correct for this complex of four tiles.

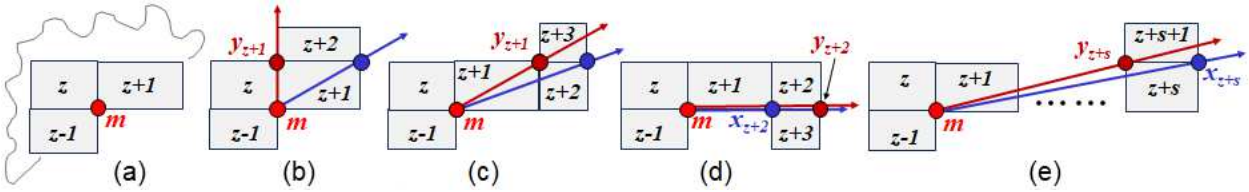

Fig. 28 All possible initial cones generated by Algorithm 6 if the last found polygon vertex m is concave, up to rotations by multiples of 90°. Notations as z within a tile mean β_z. Now m is a vertex of $fr(|\mathcal{C}|)$, the curve drawn in (a) gives an idea of the boundary of $\mathcal{C}$. From β_{z+1}, the boundary chain may continue only up, or, to the right. If β_{z+2} lies above β_{z+1}, see (b), β_{z+1} is a candidate (left turn), hence x_{z+1}, y_{z+1} determine the initial cone. If β_{z+2} lies on the right of β_{z+1}, β_{z+1} presents straight passing and is ignored, hence x_{z+2}, y_{z+2} are considered. If β_{z+2} presents a turn which may be to the left (c) or to the right (d), x_{z+2}, y_{z+2} define the initial cone. There may exist more collinear tiles, until some β_{z+s} presents a turn as in (e). The case of β_{z+s} being a right turn is not drawn, the initial cone would be degenerated to a ray. Note that it is not possible for β_{z+2} to lie below β_{z+1}, since the frontier point m could not be completely surrounded by complex tiles.

if $y_k = y_{k-1}$ lies outside the cone then both x_k, y_k are outside the cone on the left side, but if $y_k = y_{k-1}$ defines the left cone border then y_k lies on $\overrightarrow{mq}$ and x_k on the left outside the cone. For β_k being a right turn, Figure 29(d) shows when $x_k = x_{k-1}$ lies outside the cone, then both x_k, y_k are outside the cone on the right side, but $x_k = x_{k-1}$ also could define the right cone border, then x_k would lie on $\overrightarrow{mp}$ and y_k on the right outside the cone. This proves (4) for a cone in the first quadrant.

Let now the current cone be a ray $\overrightarrow{c}$ lying on a coordinate axis. We treat the positive part of the x-axis, the arguments for other cases would be similar. Such a ray is result of updating from an initial cone as those in Figures 27(b,d,f,h) and 28(d). Since all tiles $\beta_i, \beta_{i+1}, \cdots, \beta_{k-1}, \beta_k$ touch the ray $\overrightarrow{c}$ and belong to a rectangular mosaic, each tile shares its bottom or top side with $\overrightarrow{c}$. Figure 30(a-f) shows that for any movement from β_k up or down, or to the right, to β_{k+1}, the k-th cone is well-defined. A right turn (m, x_k, y_k) only appears in the situation of Figure 30(g): $\beta(\mathcal{C})$ moving from β_k to the left to β_{k+1} is only possible if β_{k-1}, β_{k-2} are situated as shown in the figure, then y_k lies on $\overrightarrow{mq}$ and x_k stays outside the cone on the left. This completes to prove (4).

□

Algorithm 6 determines the end points x_k, y_k of each boundary edge $e_k = \beta_k \cap \beta_{k+1}$, these are ignored if β_k performs straight passing of $\beta(\mathcal{C})$ (Lines 5, 14). For each β_k which performs a left or right turn, both points x_k, y_k are used to eventually update the cone borders by $p = x_k$, $q = y_k$. In Lines 18-19, one of the points p and q is considered as next polygon vertex. If β_k presents a left turn, y_k is a candidate but x_k is not. Similarly, for β_k presenting a right turn, x_k is a candidate but y_k is not. Nevertheless, the next lemma ensures that all polygon vertices found by Algorithm 6 are candidates.

Lemma 6 *For any regular complex $\mathcal{C}$ in a rectangular mosaic, and its generic boundary chain of $\mathcal{C}$ as input*

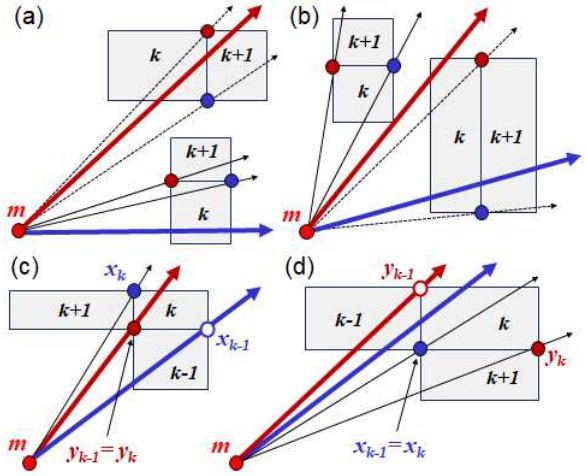

Fig. 29 Current cones in the first quadrant may result, for example, from initial cones as in Figures 27(b)(g)(j) and 28(b)(c)(e), right borders are drawn blue, left borders brown, the notation k within a tile means β_k. In (a)(b), a movement from β_k up or to the right, to β_{k+1}, always generates a well-defined k-th cone given by (m, x_k, y_k). In (c)(d), the k-th cone is not well-defined: (m, x_k, y_k) is a right turn. In (c) β_k presents a left turn, x_{k-1} and y_{k-1} determine the current cone. In (d) β_{k+1} lies below β_k, y_{k-1} defines the left current cone border but x_{k-1} lies on the right outside the cone.

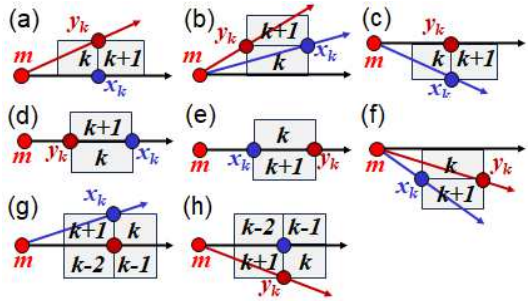

Fig. 30 All possible movements from β_k to β_{k+1} studied in the proof of Lemma 5(4) in relation to a current cone degenerated to a ray lying on the positive part of the x-axis, when m is situated at the plane origin. Note that situation (h) is not possible since then the central point $x_k \in fr(|\mathcal{C}|)$ would be surrounded by complex tiles.

list, all polygon vertices found by Algorithm 6 are candidates, that is, they belong to the list Cand($\mathcal{C}$)).

Proof First, consider a convex turn of the boundary chain where the end point x_k of $\beta_k \cap \beta_{k+1}$ restricts or confirms the right border of the cone rooted at the last found polygon vertex m. Any such situation, up to rotations by multiples of 90°, is as in Figure 31. Algorithm 6 sets $p = x_k$ which is a non-candidate point. The polygon vertex m was provided by some tile β_z, the cone is constructed such that it goes through all tiles $\beta_z, \beta_{z+1}, \beta_{z+2}, \cdots, \beta_k$. The candidate y_k may update the left cone border, then $q = y_k$, but y_k also may lie outside the cone on the left as in Figure 31(b), then $q = y_j$ remains determining the left cone border for some y_j found before, $z < j < k$. The situation guarantees that m belongs to the plane quadrant situated below the line determined by the upper side of β_{k-1}, and on the left of the right side of β_{k-1}, see Figure 31(a). Since $\mathcal{C}$ has no end tiles, the next tile β_{k+2} in the list $\beta(\mathcal{C})$ can only be one of the tiles R, S, T.

If $\beta_{k+2} = R$, $y_{k+1} = y_k$, x_{k+1} is the upper right corner of R. Then x_{k+1} lies strictly on the left of $\overrightarrow{my_{k+1}}$, see Figure 31(a), the condition of Line 9 is not satisfied. Hence, $q = y_k$ (or $q = y_j$) is found as next polygon vertex, x_k has become irrelevant and non-selectable as polygon vertex.

If $\beta_{k+2} = S$, the cone rooted at m with left border given by y_{k+1} (or y_j) and right border by x_{k+1}, is well-defined, see Figures 31(b),(c). Dependently of the position of m and the sizes of β_{k+1}, β_{k+2}, x_{k+1} may lie inside our outside the cone. If x_{k+1} lies inside the cone (or, on its borders) as in Figure 31(b), Algorithm 6 resets $p = x_{k+1}$ which restricts the cone defining a new right border, this makes x_k irrelevant. If x_{k+1} lies strictly outside the cone on the left, see Figure 31(c)), both x_{k+1} and y_{k+1} lie on the left strictly outside the cone, invalidating the condition of Line 9. Hence $q = y_k$ (or $q = y_j$) is considered as next polygon vertex, x_k has become irrelevant.

If $\beta_{k+2} = T$, x_{k+1} is a candidate which confirms the right cone border, and $x_{k+1} = x_k$, x_{k+1} may later result as polygon vertex, but x_k itself has become irrelevant.

Summarizing, the non-candidate x_k which temporarily determines the right cone border, would not result by itself as next polygon vertex. Algorithm 6 could determine a candidate found later which coincides with x_k, as polygon vertex.

For the second case of a concave turn of the boundary chain, a similar discussion would show that a non-candidate end point y_k temporarily may determine the left cone border, but, could not be selected (by itself) as next polygon vertex by Algorithm 6.

□

Theorem 1 *For any regular complex $\mathcal{C}$ in a rectangular mosaic, using one complete tracing (in counterclockwise sense) of the generic boundary chain of $\mathcal{C}$ as input list, Algorithm 6, equivalently, Algorithm 7, determines the ordered sequence of vertices of a weakly simple polygonal curve which represents the frontier of the MPP of $\mathcal{C}$.*

Proof .
(1) The equivalence between Algorithms 7 and 6 was explained in Section 8.2. Algorithm 6 already discussed in Section 8.1, clearly needs only one complete tracing of the input boundary chain.

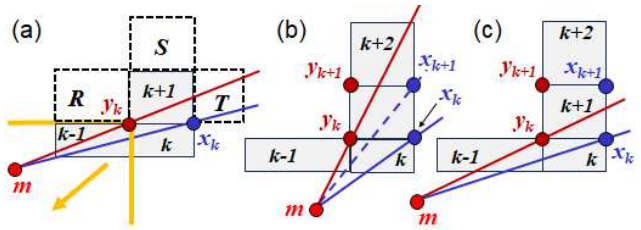

Fig. 31 A convex turn of the generic boundary chain of a rectangular complex, considered in the proof of Lemma 6. The notation k inside a tile means β_k. The right border of the cone rooted at the last found polygon vertex m is determined by the non-candidate point x_k. The candidate y_k may update the left cone border as in (a) and (c), or, could lie outside the cone on the left as in (b), m lies in the plane quadrant limited by the orange lines and indicated by the orange arrow in (a).

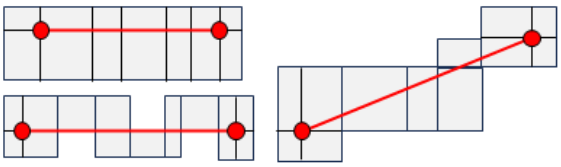

Fig. 32 Examples of regular rectangular complexes (shaded grey) whose MPP is a straight line segment (drawn red).

(2) If the MPP of $\mathcal{C}$ has exactly one vertex m, $\mathcal{C}$ is a four tiles rectangular block around the tile vertex m, see Figure 27(e), $\mathcal{C}$ coincides with its boundary. Algorithm 6 finds $m_1 = m$ from β_0, the other three tiles also have m as vertex, hence the condition of Line 5 never is satisfied, we reach Line 3 with $i = 4 > t = 3$, the algorithms ends and returns the correct MPP vertex list (m_1). If the MPP of $\mathcal{C}$ has exactly two vertices m_1, m_2, the MPP is the line segment $\overline{m_1 m_2}$. Figure 32 shows examples of such complex which coincides with its boundary. It is easy to see that Algorithm 6 correctly finds the MPP vertex list (m_1, m_2). For the following suppose that the MPP of $\mathcal{C}$ has at least three vertices.

(3) Algorithm 6 determines in Line 1 a special polygon vertex m_1 which is a convex MPP vertex as proved in [20]. From each polygon vertex $m = m_n$, $n \geq 1$, provided by some $\beta_z \in \beta(\mathcal{C})$, the next polygon vertex is found with the help of a cone of visibility through the forthcoming tiles. The cone rooted at m is initialized in Lines 6-7 by its right border $\overrightarrow{mp}$ and left border $\overrightarrow{mq}$ with $p = x_i$, $q = y_i$, being the right and left end points of $e_i = \beta_i \cap \beta_{i+1}$ where i is the first index larger than z such that b_{i-1}, b_i, b_{i+1} are not collinear. In Lines 8-16, the cone is updated after inspecting each β_k: $p = x_k$ is updated if x_k restricts or confirms the right border, $q = y_k$ is updated if y_k restricts or confirms the left border. The condition of Line 9 requires that m, x_k, y_k form a left turn, or, are collinear, and that x_k, y_k do not lie on the same side strictly outside the cone. It is important that in Line 10, x_k is ignored if it lies strictly outside the cone on the right, because then, x_k must not be used to update the right cone border, since the straight line $\overline{mx_k}$ leaves the boundary. The same in Line 11: y_k is ignored if it lies strictly outside the cone on the left, y_k must not be used to update the left border, since $\overline{my_k}$ leaves the boundary. Therefore, Lines 9-11 guarantee that (m, p, q) well-defines the cone, and that $\overline{mp}$ and $\overline{mq}$ always belong to the union of corresponding tiles, from β_z up to the tile which provides p or q.

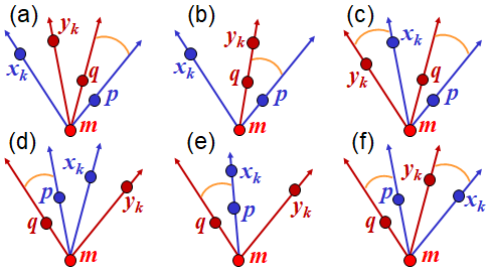

Fig. 33 All possible cases where Algorithm 6 finds the condition of Line 9 not satisfied for β_k, being the current cone given by p, q. Each yellow arc joins rays which may coincide.

As result, since p or q defines the next polygon vertex m_{n+1} (Lines 18-19) provided by some tile β_j, $\overline{m_n m_{n+1}}$ belongs to $\beta_z \cup \beta_{z+1} \cup \cdots \cup \beta_j$. By Lemma 6, all polygon vertices are candidates. In consequence, the vertex list of the polygonal curve constructed by Algorithm 6 is a subsequence of *Cand(C)* which satisfies the suppositions of Lemma 3. Hence Algorithm 6 generates the ordered vertex list of a weakly simple polygon P which is a preimage of $\mathcal{C}$, and where $core(\mathcal{C}) \subset P \subset |\mathcal{C}|$.

(4) Figure 33 shows all possible situations for the condition of Line 9 being not satisfied. A global boundary movement to the left makes Algorithm 6 find a convex polygon vertex given as q which defines the current left cone border, see Figure 33(a)(b)(c). A boundary movement to the right makes necessary to find a concave polygon vertex given as p which determines the current right border, see Figure 33(d)(e)(f). Taking into account that the current cone given by (m, p, q), and any well-defined cone given by (m, x_k, y_k) (Figure 33(c)(f)), may be degenerated to a ray, Lines 18-19 of Algorithm 6 use just one of the possible manners to distinguish between the convex and concave cases.

(5) Algorithm 6 finishes when the input list $\beta(\mathcal{C}) = \{\beta_0, \beta_1, \cdots, \beta_t\}$ was processed, i is increased and Line 3 detects that $i > t$. Suppose $\mathcal{C}$ to have more than four tiles. Recall that β_0 is the leftmost tile of the top row of $\mathcal{C}$, $\mathcal{C}$ has no end tile, and $\beta(\mathcal{C})$ is the generic boundary chain. Hence $\beta(\mathcal{C})$ moves down from β_0 to β_1, and β_t lies on the right of β_0. If β_t was the last tile inspected, it presents a convex turn, then β_{t-1} lies be-

low β_t. Otherwise, Algorithm 6 ignored β_t because $\beta(\mathcal{C})$ passes straight through it, there may be more collinear tiles $\beta_t, \beta_{t-1}, \beta_{t-2}, \cdots$ between β_0 and β_{t-r} which finally presents a convex turn, for some $r \geq 1$. In any case, $\beta(\mathcal{C})$ returns to β_t coming from the right, or from below. This makes evident that, if Algorithm 6 would continue inspecting now β_0, $\beta_1, \cdots$, it would find the convex candidate $y_t = y_0$ as next polygon vertex, but $y_0 = m_1$ was the first vertex found. This confirms the correct ending of Algorithm 6.

(6) Now let m_n be any polygon vertex determined by Algorithm 6. Suppose that m_n is an MPP vertex. Denote by a the next MPP vertex after m_n for counterclockwise tracing of the MPP frontier. We will prove that then, a coincides with the next polygon vertex m_{n+1} found by Algorithm 6. Recall that the MPP frontier curve is the shortest curve which circumscribes $core(\mathcal{C})$ and does not leave $|\mathcal{C}|$, it is polygonal and weakly simple.

(6a) The vertex m_n is a candidate provided by a tile β_z. When the new cone is initialized for some $i \geq z+1$, the condition of Line 9 is trivially satisfied for $k = i$ (Line 8), then k is stepwise augmented (Line 12) and the conditions in Lines 13,14,9, serve to decide whether the current cone continues to pass through the tiles under inspection, or not. The cone is eventually updated during this process, until for some $k > i$, Algorithm 6 reaches Line 9 and the while condition is not satisfied, then the next polygon vertex m_{n+1} is found. The candidate m_{n+1} is provided by a tile β_j with $z+1 \leq i \leq j < k$. All $\beta_{z+1}, \beta_{z+2}, \cdots, \beta_j, \cdots, \beta_{k-1}$ satisfy the condition of Line 9, but β_k does not. By construction, the current cone rooted at m_n goes through all tiles $\beta_k, \beta_{z+1}, \beta_{z+2}, \cdots, \beta_j, \cdots, \beta_{k-1}$.

(6b) Let m_n be situated at the plane origin. Assume first that the current cone rooted at m_n is not a ray lying on some coordinate axis. Then, by Lemma 5, the cone belongs to exactly one quadrant. We will treat the first quadrant, the discussion would be similar for the other quadrants.

The prove of Lemma 5(4) showed that within the first quadrant, any movement from β_l up, or, to the right, to β_{l+1}, causes that m_n, x_l, y_l form a left turn, or, are collinear (well-defined l-th cone), whereas for any movement down, or to the left, (m_n, x_l, y_l) presents a right turn which makes not satisfied the condition of Line 9 for β_l. Since $\beta_{z+1}, \cdots, \beta_j, \cdots, \beta_{k-1}$ satisfy the condition of Line 9, for all $s = i, i+1, \cdots, j, \cdots, k-1$, the s-th cone is well-defined, hence the movement from β_s to β_{s+1} may be only up, or, to the right. Therefore, m_{n+1} is positioned as in Figure 34.

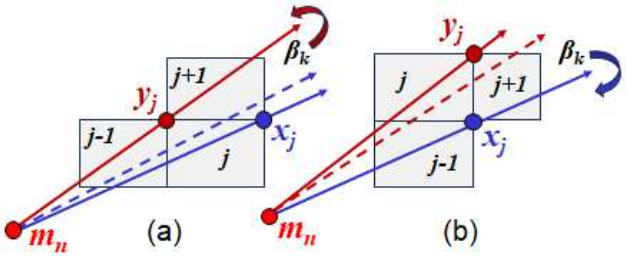

Fig. 34 For a last found polygon vertex m_n and a current cone in the first quadrant, (a): Algorithm 6 detects at β_k a boundary movement to the left which makes it find $m_{n+1} = y_j$ as the next (convex) polygon vertex, (b): a boundary movement to the right detected at β_k makes find $m_{n+1} = x_j$ as the next (concave) polygon vertex.

(6b-a) If m_{n+1} is convex, see Figure 34(a), a boundary movement to the left is detected at β_k. Then $m_{n+1} = y_j$ is the last point which restricts or confirms the left cone border, among all end points y_s of edges $e_s = \beta_s \cap \beta_{s+1}$, $s = i, i+1, \cdots, j, \cdots, k-1$. Hence $y_{j+1}, y_{j+2}, \cdots, y_{k-1}$ all lie strictly on the left of $\overrightarrow{my_j}$. The current right cone border is determined by x_j, or, lies on the left of $\overrightarrow{mx_j}$. Since $\beta_{i+1}, \cdots, \beta_{k-1}$ satisfy the condition of Line 9, $x_{j+1}, x_{j+2}, \cdots, x_{k-1}$ lie on the right of, or on, the left border $\overrightarrow{my_j}$.

The next tile β_{j+2} may lie on the left, on top, or, on the right of β_{j+1}, see Figure 35 where in (a), the $(j+1)-th$ cone is not well-defined which means that $k = j+1$. The MPP frontier part starting at m_n goes through $\beta_i, \beta_{i+1}, \cdots, \beta_j, \beta_{j+1}, \beta_{j+2}, \beta_{j+3}, \cdots$, in that order, it needs the core vertex y_j as convex vertex, because there is no straight way from m_n to reach β_{k+3}. But $\overline{m_n y_j} \subset \beta_i \cup \cdots \cup \beta_{j-1} \cup \beta_j$, which implies $a = y_j$.

In Figure 35(c), $k = j+1$ since both x_{k+1}, y_{k+1} lie outside the cone on the left. The MPP frontier part from m_n until any point of β_{k+2}, passes through y_j as vertex. This implies $a = y_j$ since $\overline{m_n y_j} \subset \beta_i \cup \cdots \cup \beta_j$.

In Figure 35(b)(d), x_{j+1} defines the right border, or, lies outside the cone on the right, whereas y_{j+1} is ignored by Algorithm 6. So, $k \geq j+2$, the algorithm continues to search for β_k. Note that in (b), β_{k+1} could not be such that y_{j+1} lies inside the cone, this would contradict the supposition that y_j is the last point defining the left cone border.

Now we assume $k \geq j+2$. To find m_{n+1} as convex, from β_k, $\beta(\mathcal{C})$ must move up, or, to the left, to β_{k+1}. If it goes from β_k up to β_{k+1} in such a way that both x_k, y_k are outside the cone on the left, $m_{n+1} = y_j$ is found. A movement from β_k to the left to β_{k+1} needs that β_{k-1} lies below β_k, by Lemma 5, then y_k lies on the left cone border and x_k outside the cone on the left. In any such situation, there is no straight line from m_n to reach β_{k+2}. The MPP frontier part from m_n to β_{k+2}

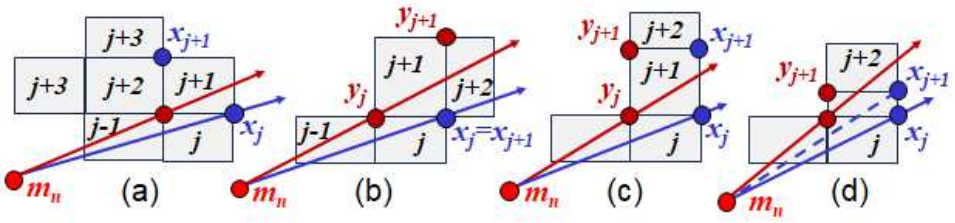

Fig. 35 Special cases for Figure 34(a), discussed in (6b-a) of the proof of Theorem 1. The notation j within a tile means β_j. (a) shows both possible continuations to β_{j+3} if β_{j+2} lies on the left of β_{j+1}, in each case, there is no straight way from m_n to reach the tile β_{k+3}. (b): β_{j+2} lies on the right of β_{j+1}, Algorithm 6 ignores y_{j+1} and continues. (c)(d): if β_{j+2} lies on top of β_{j+1}, it depends on the size of β_{k+1} whether x_{j+1} lies outside or inside the cone.

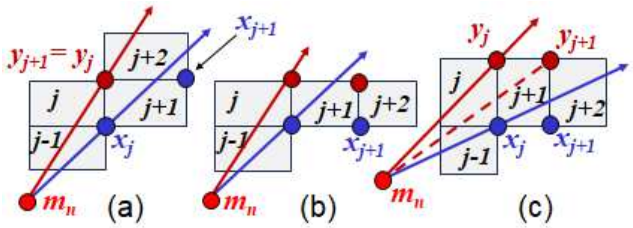

Fig. 36 Special cases for Figure 34(b), discussed in (6b-b) of the proof of Theorem 1. The notation j within a tile means β_{j+1}. (a)(c): Algorithm 6 ignores x_{j+1} and continues. (b): $k = j + 1$, there is no straight way from m_n to reach β_{k+2}.

needs a convex vertex, that is, a convex candidate. Taking into account that the MPP frontier is the shortest curve going through $\beta(\mathcal{C})$, and $\overline{my_j}$ is contained in the boundary, a must be a convex candidate provided by a tile β_t with $t \geq j$. Nevertheless, $y_{j+1}, y_{j+2}, \cdots, y_{k-1}$ lie strictly on the left of $\overrightarrow{my_j}$, making $\overline{my_t}$ to leave the boundary for $t = j{+}1, j{+}2, \cdots$. In consequence, $a = y_j$.

(6b-b) Let m_{n+1} be a concave candidate, see Figure 34(b), a boundary movement to the right is detected at β_k. Now $m_{n+1} = x_j$ is the last point which restricts or confirms the right cone border, among all end points x_s of $e_s = \beta_s \cap \beta_{s+1}$, $s = i, i+1, \cdots, j, \cdots, k-1$. Hence $x_{j+1}, x_{j+2}, \cdots, x_{k-1}$ lie strictly on the right of $\overrightarrow{mx_j}$. The current left cone border is determined by y_j, or, lies on the right of $\overrightarrow{my_j}$. Since $\beta_{i+1}, \cdots, \beta_{k-1}$ satisfy the condition of Line 9, $y_{j+1}, y_{j+2}, \cdots, y_{k-1}$ lie on the left of, or on, the right border $\overrightarrow{mx_j}$.

The next tile β_{j+2} may lie on top, or, on the right of β_{j+1}, see Figure 36. Note that β_{j+2} could not lie below β_{j+1} since $x_j \in fr(|\mathcal{C}|)$ could not be completely surrounded by complex tiles.

In Figure 36(a)(c), y_{j+1} defines the left border, or, lies outside the cone on the left, whereas x_{j+1} is ignored by Algorithm 6. Then $k \geq j + 2$, the algorithm continues. Note that in Figure 36(a), β_{k+1} could not make x_{j+1} lying inside the cone, this would contradict the supposition that x_j is the last point defining the right cone border.

In Figure 36(b), $k = j + 1$ since both x_{k+1}, y_{k+1} lie outside the cone on the right. The MPP frontier part from m_n until reaching β_{k+2}, passes through x_j as vertex. This implies $a = x_j$ because $\overline{m_n x_j} \subset \beta_i \cup \cdots \cup \beta_j$.

Now we assume $k \geq j + 2$. To find m_{n+1} as concave, from β_k, $\beta(\mathcal{C})$ must move down, or, to the left, to β_{k+1}. If it moves down in such a way that both x_k, y_k result outside the cone on the right, $m_{n+1} = x_j$ is found. A movement from β_k to the left to β_{k+1} needs that β_{k-1} lies below β_k. By Lemma 5, then x_k lies on the right cone border, and y_k lies outside the cone on the right. In any such situation, there is no straight line from m_n reaching β_{k+2}. The MPP frontier part from m_n until β_{k+2} needs a concave vertex, which is a concave candidate. Therefore, a must be a concave candidate provided by some β_t with $t \geq j$. But $x_{j+1}, x_{j+2}, \cdots, x_{k-1}$ lie strictly on the right of $\overrightarrow{mx_j}$, hence $\overline{mx_t}$ leaves the boundary for $t = j{+}1, j{+}2, \cdots$. In consequence, $a = x_j$.

(6c) Now assume m_n as situated at the plane origin, and the current cone rooted at m_n being a ray $\overrightarrow{c}$ lying on some coordinate axis. We consider the positive part of the x-axis, the discussion would be similar for other cases. The ray $\overrightarrow{c}$ is the result of eventual updatings from an initial cone which already contained $\overrightarrow{c}$ as a border. Since $\overrightarrow{c}$ is touched by all $\beta_z, \cdots, \beta_i, \cdots, \beta_{j-1}$, β_j, $\beta_{j+1}, \cdots, \beta_{j+2}$, β_k in a rectangular mosaic, each of these tiles shares its top or bottom side with $\overrightarrow{c}$.

If m_{n+1} is convex, see Figure 37(a), $m_{n+1} = y_j$ is the last point which restricts or confirms the left cone border, among all left end points y_s of $\beta_s \cap \beta_{s+1}$, $s = i, i+1, \cdots, j, \cdots, k-1$. Hence all $y_{j+1}, y_{j+2}, \cdots, y_{k-1}$ lie strictly on the left of $\overrightarrow{my_j}$. This is only possible if all tiles $\beta_{j+1}, \beta_{j+2}, \cdots, \beta_{k-1}, \beta_k$ share their bottom sides with $\overrightarrow{c}$. The boundary movement to the left is performed at β_k: β_{k+1} lies on top of β_k, both x_k, y_k are located strictly to the left of $\overrightarrow{c}$. There is no straight path from m_n to reach β_{k+1}, but the shortest path has y_j as unique vertex, hence y_j is the next MPP vertex.

Figure 37(b) shows the case that m_{n+1} is concave, then $m_{n+1} = x_j$ is the last point which restricts or confirms the right cone border, among all right end points

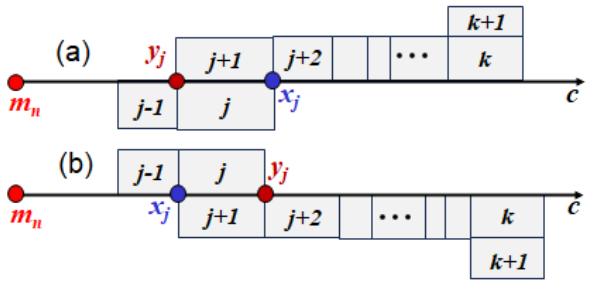

Fig. 37 A current cone rooted at m_n lies on the positive part of the x-axis. The notation k inside a tile means β_k. (a): Algorithm 6 detects at β_k a boundary movement to the left and finds y_j as next (convex) polygon vertex, (b): a boundary movement to the right detected at β_k makes find x_j as next (concave) polygon vertex.

x_s of $\beta_s \cap \beta_{s+1}$, $s = i, i+1, \cdots, j, \cdots, k-1$. Hence all $x_{j+1}, x_{j+2}, \cdots, x_{k-1}$ lie strictly on the right of $\overrightarrow{mx_j}$ which only is possible if $\beta_{j+1}, \beta_{j+2}, \cdots, \beta_{k-1}, \beta_k$ share their top sides with $\overrightarrow{c}$. The boundary movement to the right is performed at β_k: β_{k+1} lies below β_k, both x_k, y_k result as strictly to the right of $\overrightarrow{c}$. There is no straight path from m_n to reach β_{k+1}, but the shortest path has x_j as unique vertex, hence x_j is the next MPP vertex.

(7) As consequence of (6) and (5), since Algorithm 6 first finds the MPP vertex $m = m_1$, and for any found polygon vertex m_n, if it is an MPP vertex, the next MPP vertex coincides with the next found polygon vertex m_{n+1}, Algorithm 6 generates a sequence of polygon vertices which coincides with the sequence of MPP vertices for the complex $\mathcal{C}$.

□

The supposition of a generic boundary chain as input data for Algorithm 6, is important. For arbitrary (even regular) boundary chains, neither Algorithm 6 guarantees to determine the MPP vertices, nor Lemmas 3 and 4 are ensured. To see this, consider the following example: whereas the MPP is correctly obtained from the generic boundary chain in Figure 38, Figure 39 shows the incorrect result for another boundary chain of the same regular complex a square tiles. It is clear that all previous algorithms analysed in the present paper, in general, also would face this same problem, which is never mentioned in the related previous articles.

Example 13: For the complex in Figure 39, Algorithm 6 uses the points x_i, y_i, where the boundary chain performs a right or left turn. The algorithm is not able to realize whether that points really are convex vertices of the core or concave vertices of $fr(|\mathcal{C}|)$, or whether the boundary chain circumscribes the complex "always in the same orientation", since it relies on local analysis. Each y_i is interpreted as convex vertex candidate which may define the left cone border, and each x_i as concave vertex candidate to define the right border. Figures 39(b)(c) show that this interpretation causes confusion when the candidate list corresponds to a curve which transversally crosses itself. For instance, x_{12}, x_{14}, x_{16}, x_{18} are found as "concave MPP vertices" but they are convex vertices of the core. Starting with the MPP vertex y_0, $x_3 = x_4$ defines a right border, y_3 a left border which is restricted by y_4, the cone became a line. Then y_6 pretends to give a left border but lies outside the cone on the right, hence x_4 is the next "MPP vertex". From x_4, we get the borders $\overrightarrow{x_4y_6}$ (left) and $\overrightarrow{x_4x_6}$ (right), y_8 lying on the left outside the cone is ignored, but x_8 lying on the left outside the cone determines the next MPP vertex y_6. From y_6, the cone given by (y_6, x_8, y_8) already is a line, y_{11} is ignored (outside on the left), x_{11} lying outside on the left determines the next MPP vertex y_8. From y_8, x_9 gives a right border later restricted by x_{11}, y_9 a left border restricted by y_{11}. Then $x_{12} = x_{11}$ confirms the right border, y_{12} is ignored as well as x_{14} (outside), y_{14} confirms (lying on the border), x_{16} is ignored but y_{19} (outside on the right) determines the next "MPP vertex" x_{12}. Later on, from x_{18}, y_{19} gives a left border restricted by y_{21}, x_{19} a right border restricted by x_{21}. Then x_{22} is ignored, $y_{22} = y_{21}$ confirms the cone, and the input list is finished (β_{23}, β_{24}, β_{25} are ignored), hence the algorithms ends, $\overline{x_{18}y_0}$ is the last MPP side found. Only by continuation to trace the cyclic input list, the last vertices x_{18}, y_{22}, and y_0, again, may be found. In any case, the result is a curve describing a polygon which is not weakly simple, and, it is not the MPP of the complex.

8.4 Adaptation of Algorithm 6 to correct the Sklansky-Chazin-Hansen algorithm

Let $\mathcal{C}$ be a regular complex in a rectangular mosaic, $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ its generic boundary chain, b_i the centre point of β_i, $\gamma = \overline{b_0b_1} \cup \overline{b_1b_2} \cup \cdots \cup \overline{b_{t-1}b_t} \cup \overline{b_tb_0}$, u_i, w_i the sets from Definition 3 used in Algorithm 2, and x_i, y_i the end points of $e_i = \beta_i \cap \beta_{i+1}$ used in Algorithm 6. Then clearly, $w_i = x_i = x_{i-1}$ for each concave vertex b_i of γ, and $u_i = y_i = y_{i-1}$ for each convex vertex b_i of γ, see Figure 25.

Although the new Algorithms 6,7 may be applied with success to any generic boundary chain of a regular complex, the Sklansky-Chazin-Hansen algorithm could not. There is one more detail which may cause the sets of Definition 3 to be not well-defined. Recall that $\beta_0 = T_0$ is the leftmost tile in the top row of $\mathcal{C}$. Evidently, the authors of [20] have no doubt that $w_0 = \beta_0 \cap fr(|\mathcal{C}|)$ is the union of the top and the left sides of the rectangle T_0. But a regular complex is not

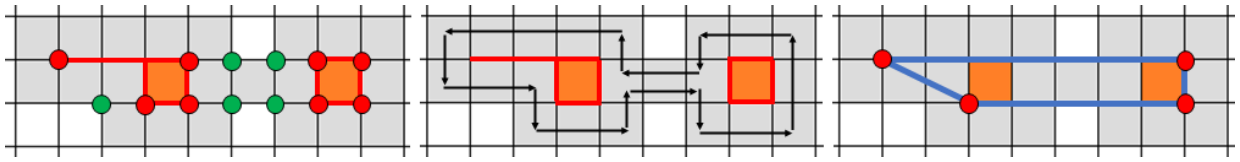

Fig. 38 A complex $\mathcal{C}$ (shaded grey) with core drawn orange and red, convex core vertices as red points, concave vertices of $fr(|\mathcal{C}|)$ as green points, the generic boundary chain by black arrows. The MPP frontier obtained by Algorithm 6 is shown blue.

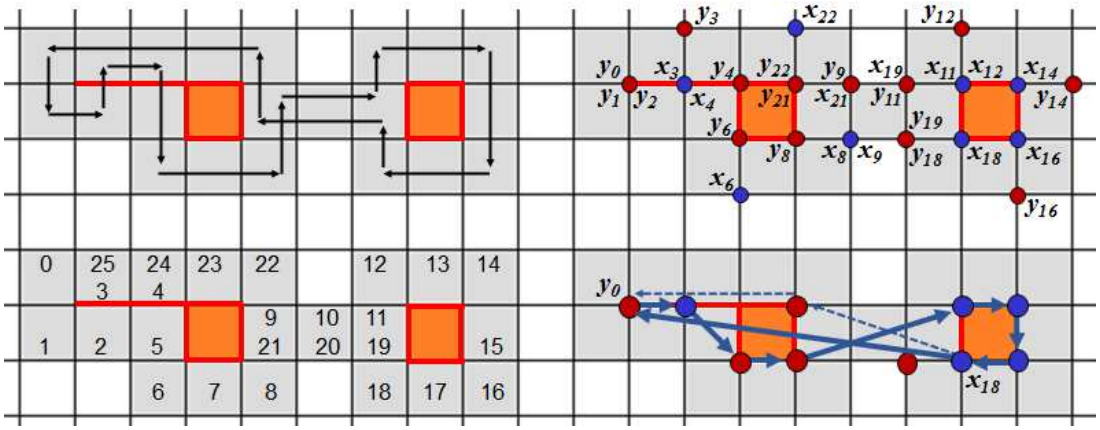

Fig. 39 Another regular boundary chain of the complex from Figure 38, discussed in Example 13, depicted by black arrows and indicated by the tile numbers. Algorithm 6 detects the points x_i, y_i and a polygonal curve drawn by blue arrows, which is not weakly simple, it crosses transversally to itself. That curve does not define the MPP, and, it has the concave vertex x_4 which is not a concave vertex of $fr(|\mathcal{C}|)$.

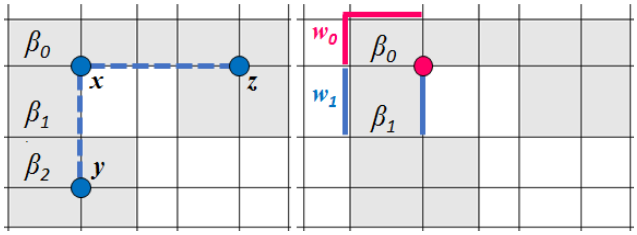

Fig. 40 A regular complex (shaded grey) having a thin part around $\beta_0 = T_0$. Its MPP (drawn by blue broken lines) is a degenerated polygon determined by the cyclic vertex sequence (x, y, x, z), x is a convex and a concave vertex, y and z are convex. Applying Definition 3, w_0 is the disconnected set drawn pink which contains the point x, u_0 results as empty. Then, w_1 is the disconnected set of two opposite sides (drawn blue in the right figure) of the next boundary tile, u_1 is empty. These results are not expected, they make impossible to calculate the angles h_1, H_1 and to continue Algorithm 2 from the first MPP vertex $m_1 = x$.

forbidden to have a thin part containing T_0, such as the complex in Figure 40 which does not satisfy the expectations for the sets w_i, u_i, w_0 is disconnected, hence so is the set w_1 of points which can be connected in $fr(|\mathcal{C}|)$ to (the components of) w_0. As result, u_0, u_1 are empty. This generates a situation useless for detecting the first convex MPP vertex x. In the present article, we simply will forbid such situation for T_0, supposing that the rectangular block of four tiles, whose top left tile is T_0, always belongs to the complex. Under these assumptions, finally, all concepts from Definition 3 are well-defined:

Lemma 7 *Let $\mathcal{C}$ be any regular complex in a rectangular mosaic $\mathcal{M}$ with generic boundary chain $\beta(\mathcal{C}) = (\beta_0 = T_0, \beta_1, \cdots, \beta_t)$ where T_0 is the leftmost tile of the top row of $\mathcal{C}$. Moreover, assume that the rectangular block of four tiles, whose top left tile is T_0, belongs to $\mathcal{C}$. For $0 \leq i \leq t$, let w_i, u_i be the sets due to Definition 3, and denote by b_i the centre point of the tile β_i. Then $\beta(\mathcal{C})$ satisfies the following for each $i \in \{0, 1, \cdots, t\}$:*

(1) w_i is a non-empty subset of the Jordan curve $fr(|\mathcal{C}|)$. That curve coincides with $w_0 \cup w_1 \cup \cdots \cup w_t$ and is traced in counterclockwise sense when visiting $w_0, w_1, \cdots, w_t$ in this order.

(2) u_i is a non-empty set.

(3) β_i corresponds exactly to one of the cases (a),(b),(c) for Definition 3. If b_i is a concave vertex of γ (case (a)) then the point w_i is a concave vertex of $fr(|\mathcal{C}|)$.

(4) If the chain $\beta(\mathcal{C})$ is non-repeating then $u_i \subset core(\mathcal{C})$, and u_i is a convex vertex of $core(\mathcal{C})$ (in generalized sense) in case that b_i is a convex vertex of γ.

Proof Due to Lemma 2, visiting all boundary tiles due to their order in $\beta(\mathcal{C})$ corresponds to counterclockwise tracing the Jordan curve $fr(|\mathcal{C}|)$, where $fr(|\mathcal{C}|)$ leaves $|\mathcal{C}|$ on its left side and $(\mathbb{R}^2 \setminus |\mathcal{C}|)$ strictly on the right.

So, the weakly simple polygonal curve $\gamma = \overrightarrow{b_0b_1} \cup \overrightarrow{b_1b_2} \cup \cdots \cup \overrightarrow{b_{t-1}b_t} \cup \overrightarrow{b_tb_0}$ is traced in counterclockwise sense.

Under the suppositions, w_0 is the union of the top and left sides of the tile T_0, hence u_0 is the lower right corner point of T_0. As proved in [20], u_0 is a convex vertex of (the frontier of) $core(\mathcal{C})$. Clearly b_0 is a convex vertex of γ. By Lemma 2, the regularity of $\mathcal{C}$ and the special position of $\beta_0 = T_0$, β_1 lies below β_0 as shown in Figure 41. The point b_1 may be a concave, convex, or linear point of γ. If b_1 is concave (right turn), β_1 satisfies Case (a) for Definition 3, then w_1 is a single point and concave vertex of $fr(|\mathcal{C}|)$, and u_1 is the union of the two sides of β_1 which are opposite to w_1. By Lemma 2(3), b_2 cannot be concave, hence the side w_2' of β_2 marked by a broken pink line in Figure 41 is a subset of w_2. If b_1 is convex (left turn), β_1 satisfies Case (b) for Definition 3, then w_1 is the union of two sides of β_1 drawn pink in Figure 41, and u_1 is the opposite single point, Figures 41(d-f) show possible continuations for β_2. If b_1 is a linear point of γ, β_1 satisfies Case (c), then w_1 and u_1 are opposite sides of β_1. In both latter cases, the point w_2' drawn as black in Figure 41, belongs to w_2, $w_2' = w_2$ if b_2 is a concave vertex of γ. In all situations, the non-empty set $w_0 \cup w_1$ is a connected part of $fr(|\mathcal{C}|)$, whereas u_0 is an isolated point, or an end point of a thin part, and hence a convex vertex of $core(\mathcal{C})$.

The construction of the sets w_i and u_i is continued while travelling through $\beta(\mathcal{C})$. By the suppositions, w_0, w_1 are well-defined subsets of $fr(|\mathcal{C}|)$, hence so are $w_2, w_3, \cdots$, and $fr(|\mathcal{C}|) = w_0 \cup w_1 \cup \cdots \cup w_t$. This completes to prove (1) and (2).

(3) and (4) are consequences of Lemmas 2 and 4, see Figure 42. If b_i is concave, β_i satisfies Case (a) for Definition 3, then w_i is a single point and concave vertex of $fr(|\mathcal{C}|)$, and u_i is the union of the two sides of β_i opposite to w_i, see also Figure 25(b). By Lemma 2, b_{i-1} and b_{i+1} cannot be concave, hence the sides w_{i-1}' and w_{i+1}' of β_{i-1} and β_{i+1}, resp., drawn by pink broken lines in Figure 42, are subsets of w_{i-1} and w_{i+1}. If b_i is convex, β_i performs a left turn and satisfies Case (b), then w_i is the union of two sides of β_i, and u_i is the opposite single point. If b_i is a linear point of γ, β_i satisfies Case (c) of Definition 3, w_i and u_i are opposite sides of β_i. In both latter cases, the points w_{i-1}' of β_{i-1} and w_{i+1}' of β_{i+1} marked as black in Figure 42 are subsets of w_{i-1} and w_{i+1}, resp.. In any case, w_i and u_i are not empty, w_i may be connected in $fr(|\mathcal{C}|)$ via a path to w_{i-1} and a path to w_{i+1} implying that the union of all w_i, $0 \le i \le t$ gives the whole curve $fr(|\mathcal{C}|)$. Part (4) follows from Lemma 4 since in that case, u_i coincides with the point y_i used in Algorithm 6.

□

Algorithm 8 presents an intuitive formulation of a corrected version of the Sklansky-Chazin-Hansen algorithm. From the analysis in Section 5.5 and the development in Section 8, it is clear that Algorithm 8 is equivalent to the new Algorithm 6, but the new one would always be preferred since it is much simpler and more efficient.

Algorithm 8 Corrected version of the Skansky-Chazin-Hansen algorithm to determine the MPP of a regular complex $\mathcal{C}$ in a rectangular mosaic.

Input: Generic boundary chain $\beta(\mathcal{C}) = (\beta_0, \beta_1, \cdots, \beta_t)$ (counterclockwise tracing), β_0 is the leftmost tile of the top row of $\mathcal{C}$, cyclic sequence $(b_0, b_1, \cdots, b_t)$ of centre points of all β_i. It is assumed that the rectangular block of four tiles, whose top left tile is β_0, belongs to $\mathcal{C}$.

Output: Lists *MPP* (MPP vertices) and L (candidates).

1: Determine a convex MPP vertex $m = m_1$ as the lower right corner of β_0.
2: Add m_1 to the list *MPP* and to the list L.
3: $i := 0$, $n := 1$.
4: **if** $i > t$ **then** STOP **endif**
5: Determine the sets w_i and u_i of β_i, the point a_i of u_i that lies furthest to the right, and the point c_i of w_i which lies furthest to the left.
6: $p := c_i$ (concave, initial right cone border $\overrightarrow{mp}$)
7: $q := a_i$ (convex, initial left cone border $\overrightarrow{mq}$)
8: $k := i$
9: **while** $(m, a_k, c_k$ form a right turn or are collinear, and c_k, a_k are not both on the same side strictly outside the cone) **do**
10: Update last concave candidate and right border: **if** $(m, p, c_k$ form a left turn or are collinear) **then** ($p := c_k$ and $z_p := k$ and add p to list L) **endif**
11: Update last convex candidate and left border: **if** $(m, q, a_k$ form a right turn or are collinear) **then** ($q := a_k$ and $z_q := k$ and add q to list L) **endif**
12: $k := k + 1$ (continue with the same cone)
13: **if** $k > t$ **then** STOP, **endif**
14: Determine the sets w_k and u_k of β_k, the point a_k of u_k which lies furthest to the right, and the point c_k of w_k that lies furthest to the left.
15: **end while**
16: (A new MPP vertex is found:)
17: **if** (m, p, c_k) forms a right turn, or, they are collinear **then** ($m := p$ and $z := z_p$) **endif**
18: **if** (m, p, c_k) forms a left turn **then** ($m := q$ and $z := z_q$) **endif**
19: $n := n + 1$, add the point $m_n = m$ to the list *MPP*.
20: $i := z + 1$ (continue searching with β_{z+1})
21: Go to Line 4.

9 Conclusions

This article studies three MPP algorithms for complexes which are sets of tiles in rectangular mosaics, two classical algorithms from [20,21] (1972, 1976), and an adaptation of them to approximate digital simple

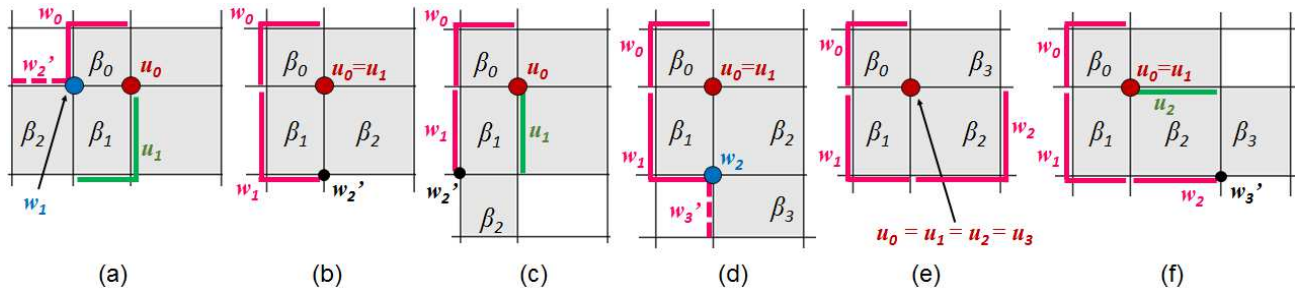

Fig. 41 Starting situations for the boundary chain $\beta(\mathcal{C})$. (a) b_1 concave, (b) b_1 convex, (c) b_1 linear, (d)-(f) possible cases of β_2 for convex b_1.

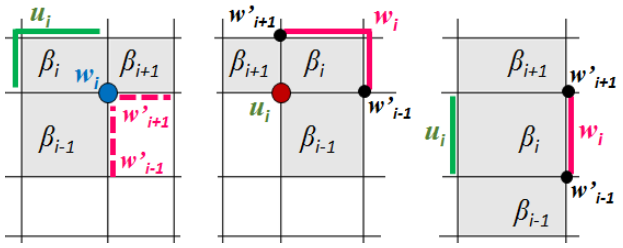

Fig. 42 Possible situations for a boundary tile β_i in the cyclic sequence $\beta(\mathcal{C})$: b_i concave (left image), b_i convex (middle image), b_i linear (right image).

4-contours which is presented and recommended in the modern textbooks [4,6] (2018, 2020) on digital image analysis, and was also presented in similar form in the book [12] (2004). The classical algorithms from [20, 21] are well-known and many cited in the literature, and never have been reported as failing. This paper shows that all these algorithms are erroneous, and that the original articles contain errors in the mathematical foundation, for example, in the definition of concepts, using failing arguments in correctness proofs, and, presenting propositions which are false.

Based on our detailed analysis, we develop a new MPP algorithm for regular complexes in rectangular mosaics, using the correct ideas from the previous algorithms but avoiding their errors. Our algorithm uses the uniquely determined generic boundary chain of the complex as input data. This chain is obtained by a boundary tracing method, also presented in this article, which in a straightforward manner generalizes the boundary tracing of 4-contours well-known from digital image analysis. We illustrate the new MPP algorithm by examples, and, we present its correctness proof. The new algorithm corrects that from [21], we also show how it can be adapted to correct the algorithm from [20].

Our new MPP Algorithm 6 may also be applied to correct the MPP algorithm from the textbooks [4,6, 12] for simple 4-contours in $(c\mathbb{Z})^2$. This will be studied, together with a generalization to complexes which not necessarily are regular, in a forthcoming article.

Acknowledgements ... ...

Conflict of interest

The authors declare to have no conflict of interest.

References

1. Chang, H.C., Erickson, J., Xu, C.: Detecting weakly simple polygons. In: P. Indyk (ed.) Proceedings of 26th Annual ACM-SIAM Symposium on Discrete Algorithms SODA 2015 (USA, 2015), vol. 3, pp. 1657–1672. SIAM (ISBN 978-1-61197-374-7) (2015). DOI 10.1137/1.9781611973730.110
2. Coeurjolly, D., Klette, R.: Comparative evaluation of length estimators of digital curves. IEEE Trans. on Pattern Analysis and Machine Intelligence **26(2)**, 252–258 (2004). DOI 10.1109/TPAMI.2004.1262194
3. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Pearson Education International, USA (2007)
4. Gonzalez, R., Woods, R.: Digital Image Processing (Global Edition), 4rd edn. Pearson Education Limited, USA (2018)
5. Gonzalez, R., Woods, R., Eddins, S.: Digital Image Processing using Matlab, 2nd edn. Gatesmark Publishing LLC, USA (2009)
6. Gonzalez, R., Woods, R., Eddins, S.: Digital Image Processing using Matlab, 3rd edn. Gatesmark Publishing LLC, USA (2020)
7. Gonzalez, R., Woods, R., Eddins, S.: Digital Image Processing using Matlab, 2nd edn. McGraw-Hill Education (India) Private Limited, USA ((20th reprint 2019))
8. Grünbaum, B., Shephard, G.: Tilings and Patterns. W.H. Freeman and Company, USA (1978)
9. Kim, C., Sklansky, J.: Digital and cellular convexity. Pattern Recognition **15 No.5**, 359–367 (1982)
10. Klette, R.: Multigrid convergence of geometric features. In: G. Bertrand, A. Imiya, R. Klette (eds.) Digital and Image Geometry, pp. 318–338. Springer, LNCS 2243, Heidelberg (2002). DOI 10.1007/3-540-45576-0-19
11. Klette, R., Kovalevsky, V., Yip, B.: On the length estimation of digital curves. In: L. Latecki, R. Melter, D. Mount, A. Wu (eds.) SPIE Proc. of Vision Geometry VIII, pp. 117–129. SPIE Proc. Series, USA (1999)
12. Klette, R., Rosenfeld, A.: Digital Geometry - Geometric Methods for Digital Picture Analysis. Morgan Kaufmann Publisher, USA (2004)

13. Klette, R., Yip, B.: The length of digital curves. Machine Graphics and Vision **9(3)**, 673–703 (2000)
14. Kovalevsky, V.: Finite topology as applied to image analysis. Computer Vision, Graphics, and Image Processing **46**, 141–161 (1989)
15. Li, F., Klette, R.: Euclidean Shortest Paths, Exact or Approximate Algorithms. Springer, London (2011). DOI 10.1007/978-1-4471-2256-2
16. Provencal, X., Lachaud, J.: Two linear-time algorithms for computing the minimum length polygon of a digital contour. In: S. Brlek, C. Reutenauer, X. Provencal (eds.) Proc. of DGCI - Int. Conf. on Discrete Geometry for Computer Imagery, pp. 104–117. Springer, LNCS 5810, Berlin Heidelberg (2009). DOI 10.1007/978-3-642-04397-0-10
17. Schulte, E.: Tilings. In: P. Gruber, J. Wills (eds.) Handbook of Convex Geometry (Vol. B), pp. 899–932. Springer, LNCS 2059, Amsterdam (1993)
18. Sklansky, J.: Recognition of convex blobs. Pattern Recognition **2**, 3–10 (1970). DOI 10.1016/0031-3203(70)90037-3
19. Sklansky, J.: Measuring cavity on a rectangular mosaic. IEEE Trans. on Computing **C-21(12)**, 1355—-1364 (1972)
20. Sklansky, J., Chazin, R., Hansen, B.: Minimum perimeter polygons of digitized silhouettes. IEEE Trans. on Computers **21(3)**, 260—268 (1972). DOI 10.1109/TC.1972.5008948
21. Sklansky, J., Kibler, D.: A theory of nonuniformly digitized binary pictures. IEEE Trans. on Systems, Man, and Cybernetics **6(9)**, 637—647 (1976). DOI 10.1109/TSMC.1976.4309569
22. Sloboda, F., Stoer, J.: On piecewise linear approximation of planar jordan curves. Journal of Computational and Applied Mathematics **55**, 369—-383 (1994). DOI doi:10.1016/0377-0427(94)90040-X
23. Sloboda, F., Zatco, B., Klette, R.: Topology of grid continua. In: Proc. of SPIE Conf. Vision Geometry VII, pp. 52–63. SPIE Proc. Series vol. 3454 (1998). DOI 10.1117/12.323274
24. Sloboda, F., Zatco, B., Stoer, J.: On approximation of planar one-dimensional continua. In: R. Klette, A. Rosenfeld, F. Sloboda (eds.) Advances in Digital and Computational Geometry, pp. 113—-160. Springer, Singapore (1998)
25. Tajine, M., Daurat, A.: On local definitions of length of digital curves. In: N. et al. (ed.) Proc. of DGCI - Int. Conf. on Discrete Geometry for Computer Imagery, pp. 114–123. Springer, LNCS 2886, Berlin Heidelberg (2003). DOI 10.1007/978-3-540-39966-7-10
26. Toussaint, G.: Computing geodesic properties inside a simple polygon. Revue d'Intelligence Artificielle **3(2)**, 265–278 (1989)
27. de Vieilleville, F., Lachaud, J.: Digital deformable model simulating active contours. In: S. Brlek et al. (ed.) Proc. of DGCI - Int. Conf. on Discrete Geometry for Computer Imagery, pp. 203–216. Springer LNCS 5810, Berlin Heidelberg (2009)
28. Wiederhold, P., Villafuerte, M.: A polygonal approximation for general 4-contours corresponding to weakly simple curves. Journal of Mathematical Imaging and Vision **64(2)**, 161–193 (2022). DOI 10.1007/s10851-021-01060-0
29. Wiederhold, P., Wilson, R.: The Alexandroff dimension of digital quotients of Euclidean spaces. Discrete and Computational Geometry **27**, 273—286 (2002). DOI 10.1007=s00454-001-0065-4