Journal of Mathematical Imaging and Vision (2020) 62:693–703
https://doi.org/10.1007/s10851-020-00957-6

Efficiently Testing Digital Convexity and Recognizing Digital Convex Polygons

Loïc Crombez[1] · Guilherme D. da Fonseca[2] · Yan Gerard[1]

Received: 8 July 2019 / Accepted: 9 April 2020 / Published online: 11 May 2020

Abstract
A set $S \subset \mathbb{Z}^2$ of integer points is *digital convex* if $\text{conv}(S) \cap \mathbb{Z}^2 = S$, where $\text{conv}(S)$ denotes the convex hull of S. In this paper, we consider the following two problems. The first one is to test whether a given set S of n lattice points is digital convex. If the answer to the first problem is positive, then the second problem is to find a polygon $P \subset \mathbb{Z}^2$ with minimum number of edges and whose intersection with the lattice $P \cap \mathbb{Z}^2$ is exactly S. We provide linear-time algorithms for these two problems. The algorithm is based on the well-known quickhull algorithm. The time to solve both problems is $O(n + h \log r) = O(n + n^{1/3} \log r)$, where $h = \min(|\text{conv}(S)|, n^{1/3})$ and r is the diameter of S.

Keywords Digital geometry · Digital convexity · Convex · Digital polyhedron · Digital polyhedron recognition · Quickhull · Polygonal separation

1 Introduction

Digital geometry is the field of mathematics that studies the geometry of points with integer coordinates, also known as *lattice points* [1]. Although the subsets of $\mathbb{Z}^d$ are not convex in the usual meaning of the term, a simple notion of convexity is induced by the convexity of $\mathbb{R}^d$ [2]. A set of lattice points $S \subset \mathbb{Z}^d$ is *digital convex* if $\text{conv}(S) \cap \mathbb{Z}^d = S$, where $\text{conv}(S)$ denotes the convex hull of S in $\mathbb{R}^d$. In other words, S is digital convex if it is the intersection of a convex subset of $\mathbb{R}^d$ with the lattice $\mathbb{Z}^d$ (Fig. 1). Digital convex lattice sets are then directly related to the lattice polytopes investigated in geometry of numbers since the works of Minkowski [3]. Digital convexity is preserved by homeomorphisms of $\mathbb{Z}^d$.

Let us remark that a digital convex lattice set S is not necessarily connected while the convex sets of $\mathbb{R}^d$ are arc-connected or simply connected. In $\mathbb{Z}^2$ and $\mathbb{Z}^3$, the lack of connectivity has led to the introduction of some alternative definitions of digital convexity that we will not consider [4–8].

✉ Loïc Crombez
lcrombez@isima.fr

Guilherme D. da Fonseca
guilherme.fonseca@lis-lab.fr

Yan Gerard
yan.gerard@uca.fr

1 LIMOS, Université Clermont Auvergne, Aubière, France

2 LIS, Aix-Marseille Université, Marseille, France

Herein, we consider the following two problems in the plane.

1.1 Testing Convexity

The first problem is to determine whether a given finite lattice set S is convex.

Problem `TestConvexity`
Input: Set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates.
Output: Determine whether S is digital convex.

The input of `TestConvexity` is an unstructured finite lattice set (without repeating elements). Related work considered more structured data, in which S is assumed to be connected. The *contour* of a connected set S of lattice points is the ordered list of the points of S having a grid neighbor (a lattice point whose Chebyshev distance to the point is one) not belonging to S. When S is connected, it is possible to represent S by its contour, either directly as in [9] or encoded by its Freeman chains code [10]. The algorithms presented in [9,10] test digital convexity in linear time on the respective input representations.

Our work, however, does not make any assumption on S being connected, or any particular ordering of the input. In this setting, a naive approach to test the digital convexity is:

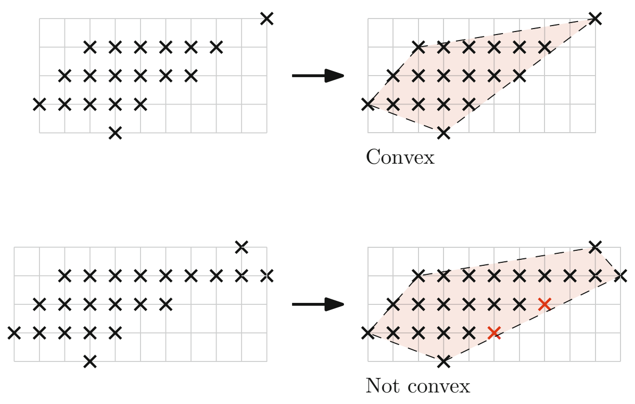

Fig. 1 Digital convexity. The first set is digital convex, while the second set is not because of the red lattice points that are inside the convex hull of the set but not in the set itself (Color figure online)

1. Compute the convex hull conv(S) of the n lattice points of S.
2. Compute the number n' of lattice points inside the convex hull of S.
3. If $n = n'$, then S is convex. Otherwise, it is not.

Step 1 consists in computing the convex hull of n points. The field of computational geometry provides a plethora of algorithms for computing the convex hull of a finite set $S \subset \mathbb{R}^2$ of n points [11]. The fastest algorithms take $O(n \log n)$ time [12], which matches the lower bound in the algebraic decision tree model of computation [13]. If we also take into consideration the output size h, i.e., the number of vertices of the convex hull, then the fastest algorithms take $O(n \log h)$ time [14,15].

Step 2 consists in computing the number of lattice points inside a convex polygon (represented by its vertices), which is a well studied problem. In dimension 2, it can be solved using Pick's formula [16]. In higher dimension, the question has been widely investigated in the framework of the geometry of numbers, from Ehrhart theory [17] to Barvinok's algorithm [18]. The currently best-known algorithms have a complexity of $O(n^{O(d)})$ for fixed dimension d [19]. Overall, the time complexity of this naive approach is at least that of the computation of the convex hull.

1.2 Digital Convex Polygon Minimization

In the case where the set S is convex, the second problem is to determine a convex polygon P having as few edges as possible and whose intersection $P \cap \mathbb{Z}^d$ with the lattice is exactly S (Fig. 2).

Problem `Minimization`

Input: Set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates.

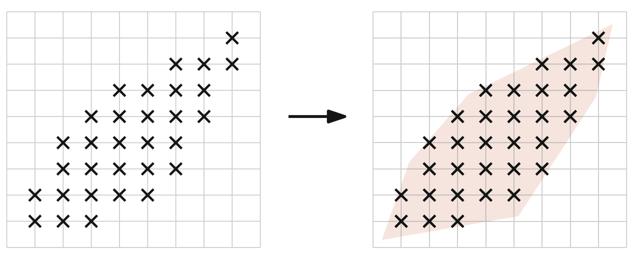

Fig. 2 Recognition. The input of the `Minimization` problem is a finite lattice set. The question is to find a convex polygon P with the smallest number of edges $P \cap \mathbb{Z}^2 = S$ and whose intersection with $\mathbb{Z}^2$ is exactly S

Output: Find a convex polygon P with minimum number q of edges verifying $P \cap \mathbb{Z}^2 = S$.

The problem `Minimization` has been mentioned in a survey of open questions in Digital geometry [20]. Contrary to what is claimed, a minimal decomposition of the boundary of the set in digital straight segments (`Min-DSS`) as in [21] does not yield a solution. We disprove the reduction of `Minimization` to `Min-DSS` with the counter-example provided in Fig. 3. A weaker form of the problem, assuming 8-connectivity of the input set, has been studied in [22].

The problem `Minimization` is a fundamental problem in geometry, related for instance to combinatorial optimization. In this field, solutions are often characterized by an exponential number of linear constraints, and the reduction of the number of linear inequalities characterizing them is a major concern. It is related to the question that we address but in our framework the set of integer points is explicitly given through the list of the coordinates of the points. Even with the assumption that the lattice set is given, the state of the art about `Minimization` is restricted to a few results. The problem `Minimization` is decidable in dimensions $d = 2$ and 3. In arbitrary dimensions, the problem is only known to be decidable if S is a non-hollow convex polytope (non-hollow means that there is some lattice points in the interior of its convex hull) [23–25]. These questions of decidability have been investigated without focus on the efficiency of the algorithms. Providing algorithms of low complexity for solving `Minimization` remained a fully open question, which we solve in this paper in dimension 2.

1.3 Our Results

In Sect. 2, we consider the problem of testing the digital convexity of a given lattice set S. We recall the linear time solution already presented in the conference version [26]. Our main result is an algorithm to solve `TestConvexity` in $O(n + h \log r) = O(n + n^{1/3} \log r)$ time, where h is the number of edges of the convex hull and r is the diameter of S. Furthermore, when the set S is digital convex, the algorithm returns the convex hull of S.

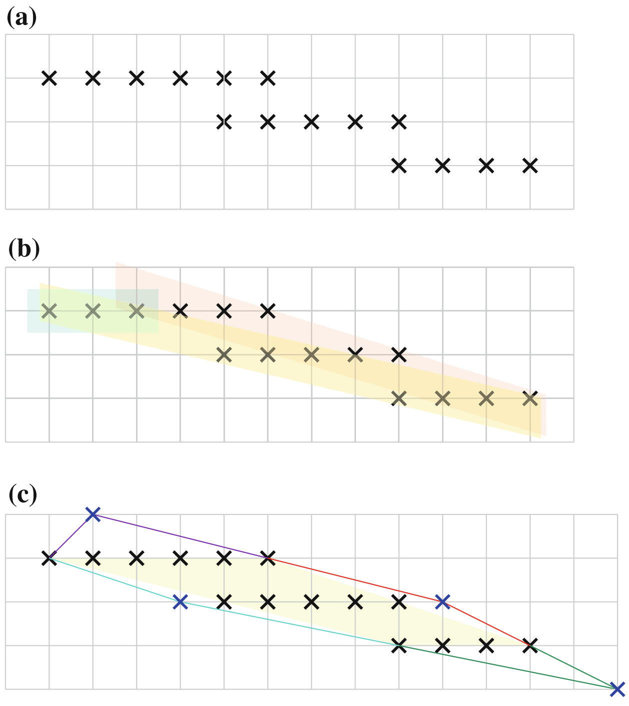

Fig. 3 Min-DSS fails to solve `Minimization`. According to [20], the problem `Minimization` can be solved by a technique decomposing the contour of a 4-connected shape in a minimal number of digital segments. The above example disproves it. The min-DSS decomposition of the boundary of the lattice set is done with only 3 digital segments while the set cannot be separated from its complement in $\mathbb{Z}^2$ by a triangle. It can be seen by considering the four exterior colored points in **c**. No pair of colored points can be separated from the lattice set by a line, which proves that the problem `Minimization` admits only a solution with at least $q = 4$ edges and not 3 as in Min-DSS (Color figure online)

In Sect. 3, we consider the problem `Minimization`. We present the first linear-time algorithm to recognize a digital convex polygon. This algorithm uses the convex hull of S computed in `TestConvexity` for digital convex sets and then solves `Minimization` in $O(h \log r)$ time.

For a given number of edges h, the number of points n and the diameter r are not bounded (consider for instance large triangles). Similarly, for a given n, the diameter r is not bounded (consider the pair of points $(0, 0)$ and $(1, r)$), however the number of vertices or edges h is bounded by $O(n^{1/3})$ [27]. At last, given the diameter r, the number of points n is clearly at most $O(r^2)$ and h is at most $O(r^{2/3})$ [28]. Expressed only with n and r, our algorithm for solving `TestConvexity` takes $O(n + n^{1/3} \log r)$ time while a post-processing step in $O(n^{1/3} \log r)$ time allows us to solve `Minimization`.

2 Digital Convexity

The purpose of this section is to provide an algorithm to test the convexity of a finite lattice $S \subset \mathbb{Z}^2$ in linear time in n.

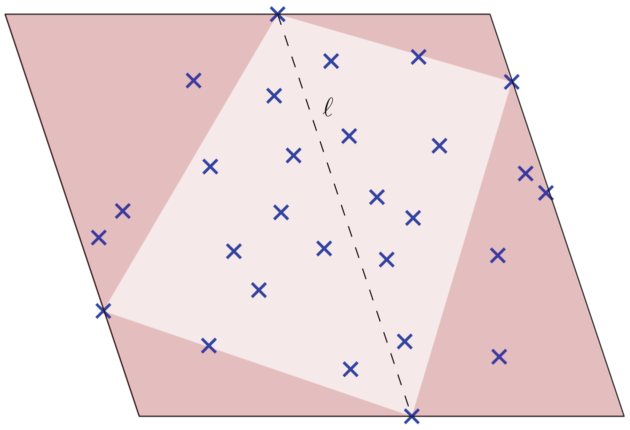

Fig. 4 Quickhull initialization. Points inside the partial hull (light brown) are discarded. The remaining points are potentially part of the hull (Color figure online)

To achieve this goal, we first show that the convex hull of a digital convex set S can be computed in linear time using the well-known quickhull algorithm [29].

Quickhull is one of the many early algorithms to compute the convex hull in dimension 2. Its worst-case time is $O(n^2)$. However, for some inputs and variations of the algorithm, the average time complexity is reduced to $O(n \log n)$ or $O(n)$ [11,30].

The quickhull algorithm starts by initializing a convex polygon in the following manner. First it computes the top-most and bottom-most points of the set. Let ℓ be the line defined by these two points. Then, the algorithm computes the farthest point from ℓ, on each side of ℓ. The (at most) four points we computed describe a convex polygon that we call a *partial hull*, which is a subset of the vertices of the convex hull of S. All points contained in the interior of the partial hull are discarded from S. Furthermore, horizontal lines and lines parallel to the top-most to bottom-most line passing through these points define an outlying bounding box containing the convex hull (Fig. 4).

After the initial step, the algorithm adds vertices one by one to the partial hull until it obtains the entire convex hull. For each edge of the partial hull, we apply the following steps. Let v denote the edge's outwards normal vector. The algorithm searches for the extreme point in direction v. If this point distance from the edge is 0, then the edge is part of the convex hull. Otherwise, we add to the convex hull the farthest point found, discarding the points that are inside the new partial hull. Throughout this paper, we call a *step* of the quickhull algorithm the computation of the farthest point of every edge for a given partial hull. When adding new vertices to the partial hull, the region inside the partial hull expands. Points inside that expansion are discarded by quickhull and herein we call this region *discarded region*.

The points not belonging to the partial hull are preserved, and are the elements of the *preserved region* (Fig. 5).

We show that quickhull steps take linear time for any digital convex lattice set and that, in this case, at each step half of the remaining input points are discarded. Therefore, the total running time remains linear, as in standard decimation algorithms (see for example [31]). In Sect. 2.2, we explain how to use this algorithm to test the digital convexity of any lattice set in linear time in n.

Theorem 1 *If the input is a digital convex set of n points, then quickhull has $O(n)$ time and space complexities.*

2.1 Proof of Theorem 1

We prove Theorem 1 as follows.

Proof During quickhull algorithm, we discard from S the points that become useless for the next computation and add some of them as vertices of the partial hull. The algorithm discards all the points that are in the interior or on the boundary of the current partial hull. The theorem is a consequence of the following two propositions, which we prove next: (i) At each step, the running time is linear in the number of points remaining in S. (ii) At least half of the remaining points are discarded at each iteration. We start by proving proposition (ii).

Consider one step of the algorithm. Let ab be the edge defining the step. When a was added to the hull, it was as the farthest point in a given direction. Hence, there is no point beyond the line orthogonal to this direction going through a (Fig. 5b). The same holds for b. Let c be the intersection point of these two lines going through a and b. We know that any remaining point of S is in the interior of such a triangle $\triangle abc$ to which it is allocated. We proceed with the remaining points of S in $\triangle abc$ as follows. We are looking for the point that is the farthest from the supporting line of ab in the triangle $\triangle abc$ (Fig. 6). Three cases might occur. If the triangle $\triangle abc$ does not contain any remaining point, then ab is an edge of the partial hull and we stop the computation for this edge in the following steps. If there is a unique remaining point of the triangle $\triangle abc$ which is the farthest from the line ab, then we denote it d. If there are multiple points which are farthest from ab in the interior of the triangle $\triangle abc$, we denote the two extreme points of S on this segment d and d'.

Let us consider the case where the point d is the unique farthest point from the line ab. Let e and f be the intersections between the line parallel to ab going through d, and respectively, ac and bc. The point d is the unique remaining point in the triangle $\triangle cef$. Adding d to the partial hull creates two other edges to be further processed: One is ad and the other is bd. Then, we insert the vertex d in the partial hull and remove from S all the points which are neither in the interior of the triangles $\triangle ade$ nor $\triangle bdf$. The points of

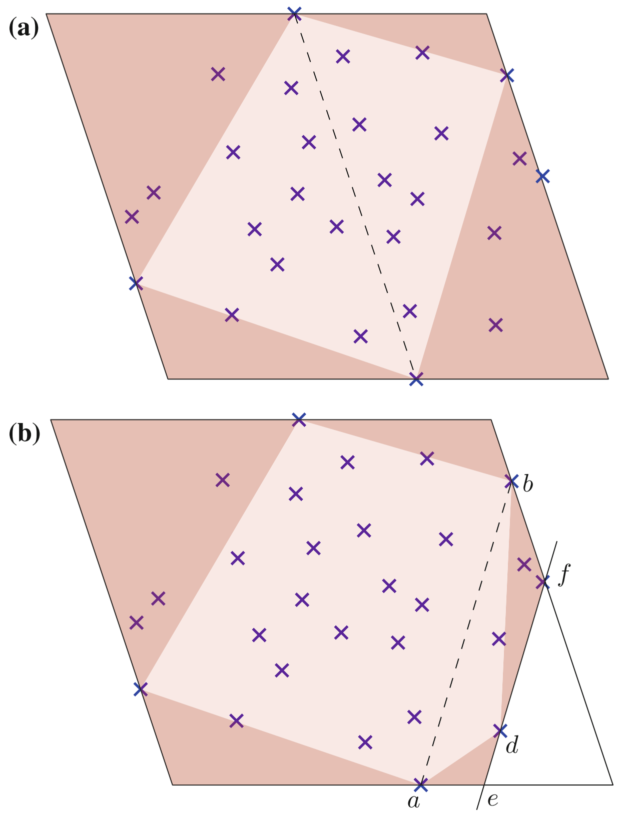

Fig. 5 Quickhull regions. The preserved region (region in which we look for the next vertex to be added to the partial hull) is a triangle. This stays true when adding new vertices to the hull (as shown here in the bottom right corner). The partial hull (whose interior is shown in light brown) grows at each vertex insertion to the partial hull. The points in or on the boundary of the new region of the partial hull are discarded (Color figure online)

S in the interior of the triangle $\triangle abc$ that we do not discard are allocated either to $\triangle ade$ or to $\triangle bdf$ according to their positions.

We denote, respectively, c_1 and c_2 the midpoints of ad and bd. All the lattice points in the interior of the triangles $\triangle ade$ and $\triangle dbf$ have different symmetric lattice points toward c_1 and c_2 in the interior the triangle $\triangle ade$. Since S is digital convex, those lattice points are in S, they also are discarded due to their positions. (Fig. 6a). In other words, at this step, for each remaining points of S, one point of S is discarded. It proves (ii). This proposition also holds in the case where there are two extreme points d and d' from S on the line ef. In this case, we insert the two vertices d and d' in the partial hull. We discard from S all the points of the triangle $\triangle abc$ which are not in the interiors of the triangles $\triangle ade$ and $\triangle d'bf$. As previously, any of the remaining points has a different symmetric point which is discarded (Fig. 6b). It proves (ii) in this case. In both cases, our initial assumption is preserved: All the remaining points are in the interior of the triangle to which they are allocated. At last, we can easily

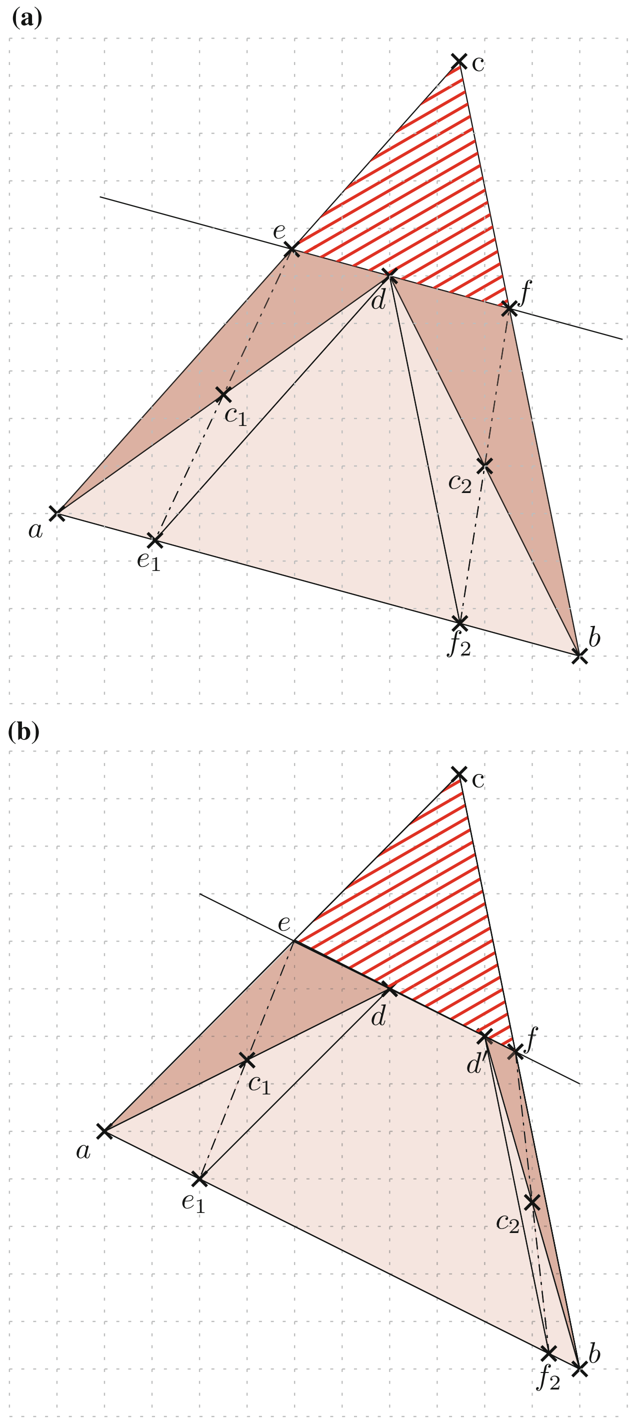

Fig. 6 Symmetrical regions. At each step, we discard from S all the points of the triangle $\triangle abc$ which are not in the interior of $\triangle ade$ or of $\triangle dbf$ ($\triangle d'bf$ in **b**). By considering the symmetries through c_1 and c_2, any of these remaining points has a symmetric lattice point in the interior of $\triangle abd$ which is discarded

provide an initialization of the partial hull and of the set of remaining points satisfying this condition.

For proving (i), the computation of the farthest point from the line ab among the remaining points of S in the triangle $\triangle abc$ takes linear time. For all points in the triangle, we test if they are in the interior of either the triangles $\triangle ade$ or $\triangle dbf$ (or $\triangle d'bf$ in the second case). We allocate them to their containing triangle or discard them. The operation takes a constant time per point. In the second case, where we have two extreme points d and d', these two points are also computed in linear time; This proves (i). Consequently, the number of operations is proportional to $n\sum_{i=0}^{\infty}(\frac{1}{2})^i = 2n$ and quickhull takes linear time for digital convex sets. □

2.2 Testing Digital Convexity

By running quickhull on any given set S, and stopping the computation if any step of the algorithm discards less than half of the remaining points, we ensure both that the running time is linear, and that if S is digital convex, quickhull finishes and returns the convex hull of S. If the computation finishes for S, we still need to test its digital convexity. To do so, we use the previously computed convex hull and compute $|\operatorname{conv}(S) \cap \mathbb{Z}^2|$ using Pick's formula [16]. The set S is digital convex if $|\operatorname{conv}(S) \cap \mathbb{Z}^2| = |S|$. Hence, the resulting Algorithm 1.

Algorithm 1 isDigitalConvex(S)

```
Require: $S$ a set of points
Ensure: true if $S$ is digital convex, false if not.
1: while $S$ is not empty do
2:   Run one step of the quickhull algorithm on $S$
3:   if quickhull discarded less than half the remaining points of $S$ then
4:     return false
5: Compute $|\operatorname{conv}(S) \cap \mathbb{Z}^2|$
6: if $|\operatorname{conv}(S) \cap \mathbb{Z}^2| > |S|$ then
7:   return false
8: return true
```

Theorem 2 *Algorithm 1 tests digital convexity of S in $O(n + h\log r) = O(n + n^{1/3}\log r)$ time, where $h = \min(|\operatorname{conv}(S)|, n^{1/3})$ and r is the diameter of S.*

Proof As Algorithm 1 runs quickhull, but stops as soon as less than half the remaining points have been removed, the running time of the quickhull part is bounded by the series $n\sum_{i=0}^{\infty}(\frac{1}{2})^i = 2n$, and is hence linear. Thanks to Theorem 1 we know that the computation of quickhull will be completed for any digital convex set. Computing $|\operatorname{conv}(S) \cap \mathbb{Z}^2|$ using Pick's formula requires the computation of both the area of $\operatorname{conv}(S)$ in $O(h)$ and the number of lattice points on its boundary, which requires the computation of a greatest common divisor. Hence, this takes $O(h \log r)$ time where h is the number of edges of $\operatorname{conv}(S)$ and r is the diameter of S. As S is digital convex if and only if $|S| = |\operatorname{conv}(S) \cap \mathbb{Z}^2|$, Algorithm 1 effectively tests digital convexity in $O(n + h \log r)$ time. □

3 Minimum Digital Convex Polygon

In this section, we consider a fundamental question of pattern recognition: the recognition of digital convex polygons, namely problem `Minimization` from Sect. 1.2. In this problem, we are given a set $S \subset \mathbb{Z}^2$ of n points and the goal is to find a convex polygon P with minimum number of edges such that $P \cap \mathbb{Z}^2 = S$. Notice that the vertices of P are not necessarily lattice points. We prove the following theorem:

Theorem 3 *Given a finite lattice set $S \subset \mathbb{Z}^2$ of n points, the Algorithm 3 solves the problem* `Minimization`*. The running time is $O(n + h \log r) = O(n + n^{1/3} \log r)$, where $h = \min(|\operatorname{conv}(S)|, n^{1/3})$ and r is the diameter of S.*

3.1 Strategy

The problem `Minimization` can be rephrased as the following polygonal separation problem with the set $IN = S$ and its complement $OUT = \mathbb{Z}^2 \setminus S$.

Problem: `Polygonal Separation`

Input: A set $IN \subset \mathbb{Z}^2$ of inliers and a set $OUT \subset \mathbb{Z}^2$ of outliers.

Output: A convex polygon $P \subset \mathbb{R}^2$ with as few edges as possible and such that all points of IN and none of OUT are inside P.

Polygonal separability has been widely investigated in the literature. An optimal algorithm for `Polygonal Separation` that takes $O((|IN| + |OUT|) \log(|IN| + |OUT|))$ time is presented in [32]. However, it cannot be applied to `Minimization` since the set of outliers $OUT = \mathbb{Z}^2 \setminus S$ is not finite.

The strategy to solve `Minimization` is as follows: We start by testing the digital convexity of S in linear time using Theorem 2. If S is not digital convex, then there is no solution. Otherwise, the algorithm quickhull computes the convex hull of S in linear time and we can proceed to the second step.

The second step of the algorithm is to reduce the set of outliers $OUT = \mathbb{Z}^2 \setminus S$ to a finite subset $OUT' \subseteq OUT$ of $O(n)$ points. In fact, we do not explicitly compute OUT'. Instead, we compute an implicit description of OUT' of size $O(h)$ in $O(h \log r)$ time, where h is the number of edges of $\operatorname{conv}(S)$.

The third step is to separate OUT' from S using the smallest number of edges. We could use the polygonal separability algorithm from [32], but that would lead to a running time of $O(n \log r + n \log n) = O(n \log r)$. Instead, we provide an algorithm that takes benefit of the lattice structure to achieve a running time of $O(h \log r)$ after the convex hull computation and digital convexity tests of the first step, that takes $O(n + h \log r)$ time.

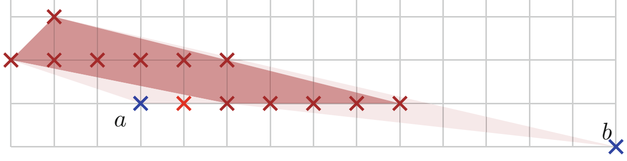

Fig. 7 Jewel's hull. In black, the set S, its convex hull is in dark red. The point a is not a jewel because of the red point, any convex polygon that includes both S and a also includes the red point. The point b is a jewel because its union $S \cap \{b\}$ with S is still convex. In other words, the convex hull of the union $S \cap \{b\}$ does not contain any other lattice points (Color figure online)

As the first step, i.e., testing the digital convexity, is already addressed in the previous section, we present the second and third steps of the algorithm in the two following sections.

3.2 Reduction

In this section, we assume that the set S is digital convex and show how to reduce the set of outliers $OUT = \mathbb{Z}^2 \setminus S$ to a finite set of $O(n)$ points. To do this, we use the notion of jewels introduced in [23,33] for testing digital circularity and recognizing digital polyhedra. We say that a point $p \in \mathbb{Z}^2 \setminus S$ is a *jewel* of S if $\operatorname{conv}(S \cup p) \cap \mathbb{Z}^2 = S \cup p$ (Fig. 7). The set of all the jewels of S is denoted $\operatorname{Jewel}(S)$ and it has the property that a convex set separates S from $\mathbb{Z}^2 \setminus S$ if and only if it separates separates S from $\operatorname{Jewel}(S)$ [23]. Hence, the infinite set of the outliers of our problem of separability can be reduced from $OUT = \mathbb{Z}^2 \setminus S$ to $OUT' = \operatorname{Jewel}(S)$.

The number of jewels is infinite if and only if S is the intersection of a line segment and $\mathbb{Z}^2$ [23]. In this case, it is easy to see that the set S forms a digital triangle. A simple way to establish bounds on the number of jewels has been discovered by French high school students during the national contest TFJM2017. They presented the following structure of the set of jewels: The jewels of the lattice set S are the lattice points that lie on the edges of a polygon J surrounding the convex hull of S. This surrounding polygon $J \supset \operatorname{conv}(S)$ is the arithmetic dilation of $\operatorname{conv}(S)$ obtained by moving the support lines of the edges of the $\operatorname{conv}(S)$ to the next Diophantine lines toward the exterior (Fig. 8). We define J as the *jewel hull* of S (Fig. 8) and define it more formally as follows.

Given S, let $E = \{e_1, e_2, \ldots e_h\}$ be the edges of $\operatorname{conv}(S)$. For each i, let $HP_i : a_i x + b_i y + c_i \leq 0$ (a_i and b_i co-prime integers) be the closed supporting halfplane associated with e_i such that $S \subset HP_i$. Notice that $\operatorname{conv}(S) = \bigcap_i HP_i$. Consider the open halfplanes $HP_i' : a_i x + b_i y + c_i < 1$. Notice that there is no integer point in $HP_i' \setminus HP_i$. The *jewel hull* of S is the closure of the intersection of the half-planes HP_i' (Fig. 8).

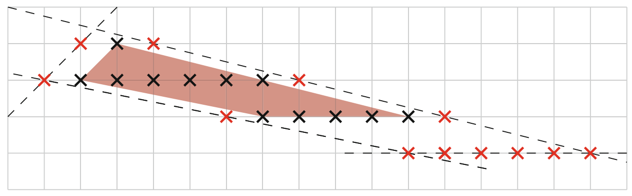

Fig. 8 Jewels.In black, the set S, its convex hull is in dark red. The halfplanes H_i' are delimited by the dashed lines, and form the *jewel hull* that surrounds the convex hull of S. The jewel hull has three properties: Its edges are parallel to the ones of the convex hull of S, there are no point between the convex hull and the jewel hull and all the jewels (drawn in red are) on its boundary (Color figure online)

The jewel hull J of S has three main properties. (i) By construction, its edges are parallel to the edges of conv(S). (ii) It is easy to prove that there is no integer point between conv(S) and the jewel hull J. Finally, (iii) the jewels of S are a subset of J. This last property is in fact a corollary of the first Lemma of [34] which is reformulated in the next lemma.

Lemma 1 *For any three lattice points p_1, p_2, p_3 such that p_1, p_2 lie on the line $ax+by+c=0$ (coefficients a and b are coprime) and p_3 does not, we have that the triangle $p_1p_2p_3$ either contains a lattice point on the line $ax+by+c+1=0$ or on the line $ax+by+c-1=0$.*

Proof Up to a lattice preserving affine isomorphism, we can assume $p_1 = (0, 0)$ and $p_2 = (0, u)$ while the images of the two lines are $x = -1$ and $x = 1$. We assume p_3 lies on the right of p_1p_2 (the other case is identical by symmetry). Hence, there exists three integers u, v, w with $u, v > 0$ such that $p_1 = (0, 0)$, $p_2 = (0, u)$, and $p_3 = (v, w)$ and we want to prove that the triangle $p_1p_2p_3$ contains an integer point on the line $x = 1$. The lower and upper points of the triangle in the line $x = 1$ are the two intersection points of $x = 1$ and each of the two segments p_1p_3 and p_2p_3. Their coordinates are, respectively, $(1, \frac{w}{v})$ and $(1, u + \frac{w-u}{v})$. Then, the intersection of the line $x = 1$ and the triangle $p_1p_2p_3$ contains an integer point if and only if the interval $[\frac{w}{v}, \frac{uv+w-u}{v}]$ contains an integer namely if the interval $[w, w + u(v - 1)]$ contains a multiple of v, which is trivially true since there is necessarily a multiple of v in any interval $[w, w + v[$ and then in $[w, w + v - 1] \subset [w, w + u(v - 1)]$ as $u \geq 1$. □

The area of the jewel hull of S is finite unless all the points of S are colinear. This case is easy to detect, and it is easy to see that in this case there exists a triangle with vertices in $\mathbb{R}^2$ that separates S from $\mathbb{Z}^2 \setminus S$.

The jewel hull consists of the intersection of a set of h halfplanes. Computing the vertices of the intersection of halfplanes is the dual [11, Chapter 8] of the computation of the convex hull of a given points set. In the general case, computing the intersection of h halfplanes takes $O(h \log h)$ time [11, Chapter 4]. However, since we already have the h halfplanes sorted by slope, we can use Graham Scan [11, Chapter 1] to compute the jewel hull in $O(h)$ time. Notice that not all h halfplanes appear on the boundary of the jewel hull, which is the dual of the fact that some points may be in the interior of the convex hull.

3.3 Jewel Separation

The jewel separation is the final step to solve the `Minimization` problem. The jewel hull J has been computed and the problem is the polygonal separation of $IN = S$ and the jewel set $OUT' = Jewel(S)$. The previous step does not provide the set of jewels but the ordered list of edges of the jewel hull J as a sequence of linear equalities ℓ_i : $a_ix+b_iy+c_i = 1$ with coprime integers a_i and b_i. An initial lattice point d_i of each given Diophantine straight lines ℓ_i can be computed with the extended Euclid algorithm in $O(\log r)$ time. We can go from this first point to the other integer points of the line ℓ_i through translations of vectors $k\overrightarrow{(-b_i, a_i)}$ where $k \in \mathbb{Z}$. Nevertheless, J is a rational polytope. Its vertices are the intersection point of consecutive Diophantine lines ℓ_i but they are not necessarily integer points. It is even possible that some edges of the jewel hull do not contain any integer point. By computing the vertices of each edge e_i we can count all the jewels on ℓ_i and obtain a generating formula for them in $O(1)$ time and space for each edge. The jewels on ℓ_i are: $\bigcup_k d_i + k(-b_i, a_i)$. The computation of an integer point d_i per line ℓ_i for each one of our at most h Diophantine lines takes $O(h \log r)$. The computation of the vertices of J takes $O(h)$ time, and hence the computation of the formulas generating the jewels takes $O(h \log r)$ time and $O(h)$ space.

The jewels are determined in counterclockwise order according to their order of appearance in the jewel hull. Their cyclic index i goes from 0 to $|jewel(S)| - 1$. Furthermore, any pair of indices i, j with $i < j$ defines two intervals of indices, the interval $I_{i\to j}$ containing the indices of the successors of i until j and the interval $I_{j\to i}$ containing the indices of the successors of j until i. We introduce now the precise meaning of *separation*. We say that a real line ℓ *separates* some jewels from S if S lies entirely on one side of ℓ while the jewels lie strictly on the other side. The fact that all jewels lie on the boundary of a convex polygon leads to the following simple lemma:

Lemma 2 *If ℓ is a line separating the jewels of indices i and j from S, then the line ℓ separates S from either the jewels with indices in $I_{i\to j}$ or the jewels with indices in $I_{j\to i}$.*

A naive approach to solve the polygonal separation problem of the sorted set of jewels from S is the following: Choose a starting jewel of index i_0. Search for the index j_0 such that the jewels with indices in the interval $I_{i_0\to j_0}$ can be separated from S and $|I_{i_0\to j_0}|$ is maximized. The method used to compute j_0 in constant time using our representation of the jewels

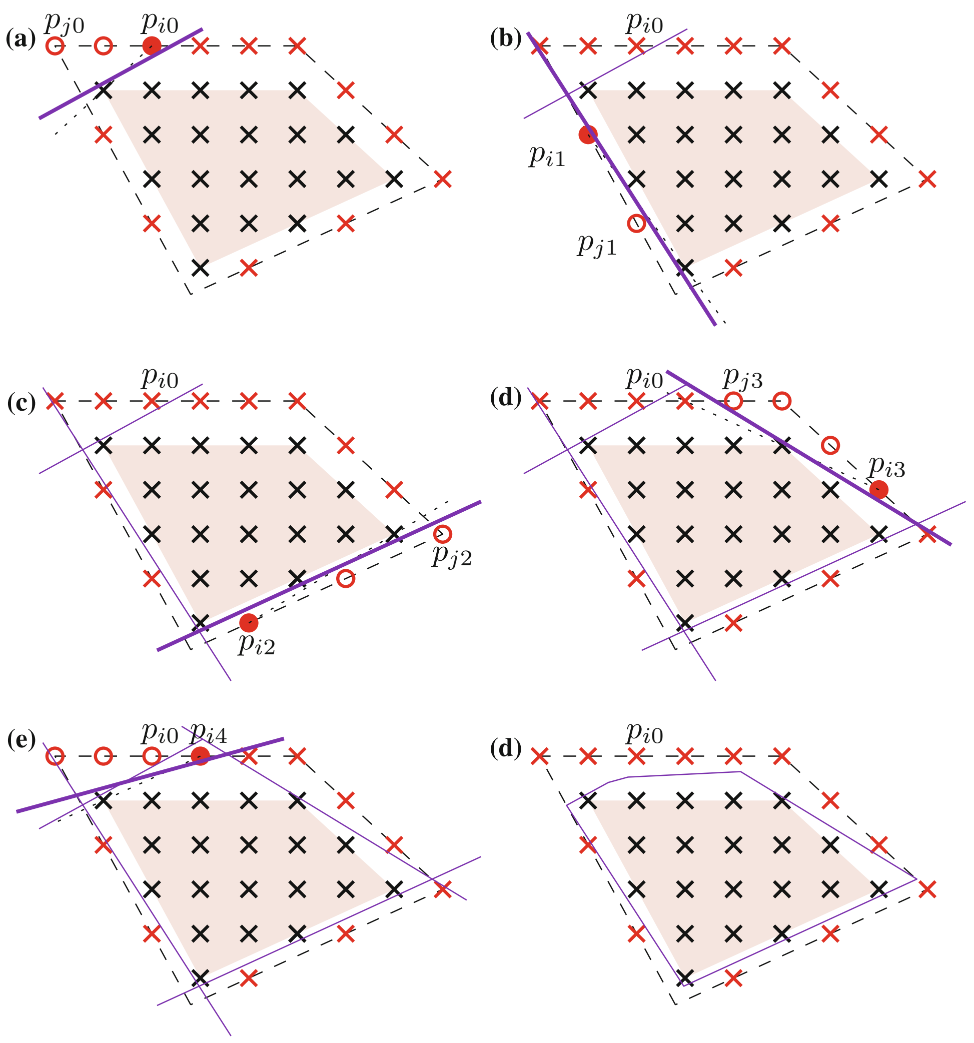

Fig. 9 Turn algorithm. We start from a chosen starting jewel p_{i_0} and search for its last successor p_{j_0} that can be separated from S simultaneously with p_{i_0} by a single line. We then take the successor of p_{j_0} as new starting jewel p_{i_1} and search for the last successor p_{j_1} of p_{i_1} that can be separated with p_{i_1}... We repeat the process until reaching the predecessor of p_{i_0}

will be detailed later. We then define i_1 as the successor of j_0 and repeat the process: search for j_1 such that $I_{i_1 \to j_1}$ can be separated from S and the number of jewels in the interval is maximized. We repeat until we find an interval $I_{i_k \to j_k}$ which contains the predecessor of i_0. The number of lines of the solution is the number $k+1$ of intervals considered. This algorithm is illustrated Fig. 9. We call this greedy algorithm the `turn` routine since the strategy is to turn around the set S from a starting jewel p_{i_0}.

The difficulty of this approach is that different choices of the starting point p_{i_0} may lead to different numbers of separating lines (actually, they may differ by at most 1 line). The strategy to find the minimum number of separating lines is to test several starting jewels. Dynamic programming approaches might be used to find an optimal solution as in [21], but in the framework of our `minimization` problem in the lattice, we are able to obtain a major simplification.

The strategy to simplify the problem is the following. There are two families of jewels: The ones which chosen as starting jewel in the `turn` routine provide a minimal number of lines, their indices are denoted I_{OPT}, and the ones that provide a non optimal number of lines. Notice that if the index i_0 is in I_{OPT}, then all the indices i_k computed during the `turn` routine are also in I_{OPT} since it can be easily seen that they provide also optimal solutions. In the general case of polygonal separability, a large set of starting points has to be investigated until finding one leading to an optimal

Algorithm 2 turn($conv(S)$, Jewel(S),i_0)

Require: the convex hull conv(S), the ordered list of its jewels Jewel(S), and a starting jewel p of index i_0.
Ensure: A separating polygon with S inside and Jewel(S) outside.

```
1: Initialize $i_0$ as the index of the starting jewel, $k = 0$ and $I_{i_{-1} \to j_{-1}}$ as an empty interval
2: while $predecessor(i_0) \notin I_{i_{k-1} \to j_{k-1}}$ do
3:     Compute $j_k$ such that the jewels with indices in the interval $I_{i_k \to j_k}$ can be separated from $S$ and $|I_{i_k \to j_k}|$ is maximized.
4:     $i_{k+1} \leftarrow successor(j_k)$
5:     $k = k + 1$
6: return The polygon obtained from the separating lines
```

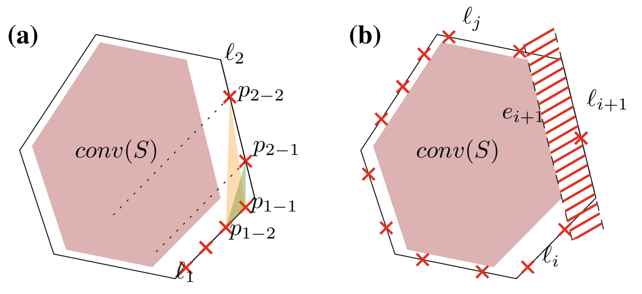

Fig. 10 Jewel separation. **a** If a single line separates both p_{1-2} and p_{2-2}, then the triangle $\triangle p_{1-1}p_{1-2}p_{2-2}$ is larger than $\triangle p_{1-1}p_{1-2}p_{2-1}$ and hence must contain a fourth lattice point, which is impossible. **b** No jewels lie between e_{i+1} and ℓ_{i+1} hence it is impossible to separate simultaneously jewels from ℓ_i and jewels from ℓ_j

solution but in the framework of the separation of $IN = S$ and $OUT' = \text{Jewel}(S)$, we can provide a subset of at most 4 jewels containing at least one in I_{OPT}. It means that testing these four jewels as starting points of the `turn` routine is enough to find the optimal solution. The properties of the set I_{OPT} are presented in the next two lemmas.

The first lemma states that there is no line that simultaneously separates two jewels of a line ℓ_i and two jewels of ℓ_{i+1}.

Lemma 3 *Let ℓ_1 and ℓ_2 be two jewel lines. (i) If $\ell_1 \cap \ell_2 \notin \mathbb{Z}^2$ then there is no line that separates two jewels of ℓ_1 and two jewels of ℓ_2. (ii) If $\ell_1 \cap \ell_2 \in \mathbb{Z}^2$ then there is no line that separates three jewels of ℓ_1 and three jewels of ℓ_2.*

Proof (i) Let order the jewels on ℓ_1: $J_1 = \{p_{1-1}, p_{1-2}, \ldots\}$ according to their distance to ℓ_2, and order the jewels on ℓ_2: $J_2 = \{p_{2-1}, p_{2-2}, \ldots\}$ according to their distance to ℓ_1. Assume that there is a line l such that l separates two jewels of ℓ_1 and two jewels of ℓ_2 from conv(S). Then, l separates $p_{1-1}, p_{1-2}, p_{2-1}$ and p_{2-2} from conv(S). Hence, the triangle $\triangle p_{1-1}p_{1-2}p_{2-2}$ lies inside the jewel hull and outside of conv(S) (Fig. 10a). As the triangle $\triangle p_{1-1}p_{1-2}p_{2-1}$ is not degenerated we have $Area(\triangle p_{1-1}p_{1-2}p_{2-1}) \geq \frac{1}{2}$. Hence, the inequality $Area(\triangle p_{1-1}p_{1-2}p_{2-2}) > Area(\triangle p_{1-1}p_{1-2}p_{2-1})$ leads to $Area(\triangle p_{1-1}p_{1-2}p_{2-2}) > \frac{1}{2}$. Using Pick's theorem we can conclude that $\triangle p_{1-1}p_{1-2}p_{2-2}$ contains at least four lattice points. However, since $p_{1-1}p_{1-2}$ are two consecutive lattice points of ℓ_1, this means that there is a lattice point strictly inside the jewel hull and outside conv(S), which is impossible. Hence, l does not exist. The proof of (ii) is the same, we just have to consider $p_{1-0} = p_{2-0} = \ell_1 \cap \ell_2$. □

We complete Lemma 3 with a lemma about the separation of jewels which are not in consecutive lines ℓ_i and ℓ_{i+1}.

Lemma 4 *If ℓ_i and ℓ_j are two non-consecutive jewels lines: $j \geq i + 2$, then there is no line that separates any jewel that belongs only to ℓ_i and any jewel that belongs only to ℓ_j.*

Proof Consider ℓ_{i+1} and its associated edge on conv(S): e_{i+1}. By construction, there is no lattice points between ℓ_{i+1} and e_{i+1} (Fig. 10b). Assume that there is a line l that separates jewels of both ℓ_i and ℓ_j As all the jewels belonging only to ℓ_i and all the jewels belonging only to ℓ_j lies on the same side s_j of e_{i+1} as S, l has to be in s_j to separate jewels of ℓ_i, then has to leave s_j in order to not intersect conv(S), and finally has to go back in s_j to separate jewels of ℓ_{i+1}. Hence, l intersects e_{i+1} twice which is impossible. □

We now explain how to use Lemmas 3 and 4 to determine at most four jewels such that at least one of them leads to an optimal solution with the `turn` routine. In other words, we provide four indices with the guarantee that at least one of them is in I_{OPT}. For convenience, the successor of the index s is now simply denoted $s + 1$ and so on with the successor of the successor denoted $s + 2$. In the same manner, we also use $s - 1, s - 2, \ldots$ to denote the predecessors of s. When looking for a jewel in I_{OPT}, several cases might occur:

1. The jewel hull J has an edge e_i which does not contain any integer point. If we denote s the index of the first jewel after this edge, then I_{OPT} contains s. It is a corollary of Lemma 4. Considering an optimal solution, the vertex of index s cannot be included in the interval $I_{i_r \to j_r}$ containing $s - 1$ because the interval would contain jewels of the lines ℓ_{i-1} and ℓ_{i+1} which is excluded by Lemma 4. Hence, the index s is a starting index, namely an index of the form i_r of the considered optimal solution. As the indices i_r of the intervals $I_{i_r \to j_r}$ computed from an optimal starting index i_0 are also optimal, s is included in I_{OPT}.
2. The jewel hull J has an edge e_i with only one jewel s, hence I_{OPT} contains either s or $s + 1$. Considering an optimal solution, it follows from Lemma 4 that $s - 1$ and $s + 1$ cannot be in an interval of the form $I_{i_r \to j_r}$ since they are on distant lines ℓ_{i-1} and ℓ_{i+1}. Hence, there exist either an index i_r equal to s or to $s+1$. It proves that one of these two indices s or $s + 1$ is in I_{OPT}.
3. The jewel hull has an edge with only two jewels. Their indices are s and $s + 1$. Considering an optimal solution, according to Lemma 4 the indices $s - 1$ and $s + 2$ cannot be in the same interval $I_{i_r \to j_r}$ because they belong to the distant lines ℓ_{i-1} and ℓ_{i+1}. Hence, there is at least a beginning of interval in s, $s + 1$ or $s + 2$. One of these three indices s, $s + 1$, $s + 2$ is in I_{OPT}
4. The edges of the jewel hull all contain at least three jewels. We choose any edge e_i and denote s, $s + 1$, $s + 2$ the indices of its three firsts jewels. According to Lemma 3, the indices $s + 2$ and $s - 2$ cannot be in the same interval $I_{i_r \to j_r}$. Hence, there is at least a beginning of interval in $s - 1$, s, $s + 1$ or $s + 2$. One of these four indices $s - 1$, s, $s + 1$, $s + 2$ is in I_{OPT}.

In any case, we can determine a set of at most four starting jewels with the guarantee that the `turn` algorithm provides an optimal solution for at least one of them. We now explain how, in the turn algorithm 2, for a given jewel p_i we compute its last successor p_j that can be separated alongside him with a single line. Let p_i be on the jewel line ℓ_i, and let v_i be the end vertex of the edge of the convex hull parallel to ℓ_i. Consider the line $p_i v_i$. S lies on one side of $p_i v_i$, all the jewels that lies strictly on the other side can be separated alongside p_i (Fig. 9). It is clear that all jewels located on ℓ_i can be separated with p_i, and using Lemma 4 we know that the jewels located on ℓ_{i+2} cannot. Hence, all we have to do is determine the last jewel of ℓ_{i+1} that lies on the correct side of $p_i v_i$. This is easily done by computing the intersection point q of $p_i v_i$ and ℓ_{i+1} and expressing q as $d_{i+1} + \lambda(-b_{i+1}, a_{i+1})$ (We remind that the jewels on ℓ_{i+1} are expressed as: $\bigcup_k d_i + k(-b_i, a_i)$). From there a separating line can be computed by rotating slightly $p_i v_i$ around any points in between p_i and v_i.

The time complexity of the `turn` algorithm 2 is hence $O(h) = O(n^{1/3})$. This follows from the fact that h is an upper bound to the number of edges of the solution of the `Minimization` problem and $h = O(n^{1/3})$. Starting from any jewel, the algorithm computes a polygon that has at most one edge more than the optimal solution and each edge is computed in $O(1)$ time.

As the jewel hull is computed $O(h \log r)$ time, the set of $O(1)$ starting jewels can be computed in constant time, and the `turn` algorithm 2 runs in $O(h)$ time. The minimization algorithm, once provided with the convex hull of S runs in $O(h \log r)$ time, which proves Theorem 3.

Algorithm 3 recognition(S)

```
Require: $S$ a set of points.
Ensure: A minimal separating polygon if $S$ is digital convex.
1: Test the digital convexity of $S$ and compute $conv(S)$ using quickhull
2: Compute the jewel hull of $S$ using Graham Scan
3: Compute at most four starting jewels
4: for all starting jewels do
5:     Compute the minimal separating polygon using the given starting jewel using algorithm 2
6: return The minimal separating polygon
```

4 Perspectives

We showed that the convex hull of a digital convex set in dimension 2 can be computed in linear time, and we can determine the minimum digital convex polygon in the same complexity. Can the convex hull of digital convex sets be computed in linear time in dimension 3, or more generally, what is the complexity of convex hull computation of a digital convex set in any fixed dimension? We note that the number of faces of any digital convex set in d dimensions is $O(V^{(d-1)/(d+1)})$, where V is the volume of the polytope [35,36]. Therefore, the lower bound of $\Omega(n^{\lfloor (d-1)/2 \rfloor})$ for the complexity of the convex hull of arbitrary polytopes does not hold for digital convex sets. The decidability of the polygon minimization problem has been proven in dimension 3 [25], but no polynomial-time algorithm have been presented yet. Even the decidability of the problem remains an open problem for dimensions higher than 3.

Acknowledgements This work has been sponsored by the French government research program "Investissements d'Avenir" through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25). A preliminary version of this paper appeared in the 21st International Conference on Discrete Geometry for Computer Imagery (DGCI 2019).

References

1. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Elsevier, Amsterdam (2004)
2. Ronse, C.: A bibliography on digital and computational convexity (1961–1988). IEEE Trans. Pattern Anal. Mach. Intell. **11**, 181–190 (1989)
3. Minkowski, H., Zahlen, Geometrie der.: . No. vol. 2 in Geometrie der Zahlen, B.G. Teubner (1910)
4. Kim, C.E., Rosenfeld, A.: Digital straight lines and convexity of digital regions. IEEE Trans. Pattern Anal. Mach. Intell. **4**(2), 149–153 (1982)
5. Kim, C.E., Rosenfeld, A.: Convex digital solids. IEEE Trans. Pattern Anal. Mach. Intell. **4**(6), 612–618 (1982)
6. Chassery, J.-M.: Discrete convexity: definition, parametrization, and compatibility with continuous convexity. Comput. Vis. Graph. Image Process. **21**(3), 326–344 (1983)
7. Kishimoto, K.: Characterizing digital convexity and straightness in terms of length and total absolute curvature. Comput. Vis. Image Underst. **63**(2), 326–333 (1996)
8. Chaudhuri, B.B., Rosenfeld, A.: On the computation of the digital convex hull and circular hull of a digital region. Pattern Recogn. **31**(12), 2007–2016 (1998)
9. Debled-Rennesson, I., Rémy, J.-L., Rouyer-Degli, J.: Detection of the discrete convexity of polyominoes. Discrete Applied Mathematics, vol. 125, no. 1, pp. 115–133, 2003. In: 9th International Conference on Discrete Geometry for Computer Imagery (DGCI 2000)
10. Brlek, S., Lachaud, J.-O., Provençal, X., Reutenauer, C.: Lyndon + Christoffel = digitally convex. Pattern Recognition, vol. 42, no. 10, pp. 2239–2246, 2009. Selected papers from the 14th IAPR International Conference on Discrete Geometry for Computer Imagery (2008)
11. Berg, M.D., Cheong, O., Kreveld, M.V., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Santa Clara (2008)
12. Yao, A.C.-C.: A lower bound to finding convex hulls. J. ACM **28**, 780–787 (1981)
13. Preparata, F.P., Hong, S.J.: Convex hulls of finite sets of points in two and three dimensions. Commun. ACM **20**, 87–93 (1977)
14. Chan, T.M.: Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete Comput. Geom. **16**, 361–368 (1996)

15. Kirkpatrick, D., Seidel, R.: The ultimate planar convex hull algorithm? SIAM J. Comput. **15**(1), 287–299 (1986)
16. Pick, G.: "Geometrisches zur zahlenlehre," Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereines für Böhmen "Lotos". Prag **47–48**, 1899–1900 (1899)
17. Ehrhart, E.: Sur les polydères rationnels homothétiques à n dimensions. Technical report, académie des sciences, Paris (1962)
18. Barvinok, A.I.: A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. Math. Oper. Res. **19**(4), 769–779 (1994)
19. Barvinok, A.I.: Computing the Ehrhart polynomial of a convex lattice polytope. Discrete Comput. Geom. **12**, 35–48 (1994)
20. Asano, T., Brimkov, V.E., Barneva, R.P.: Some theoretical challenges in digital geometry: a perspective. Discrete Appl. Math. **157**(16), 3362–3371 (2009)
21. Feschet, F., Tougne, L.: On the min DSS problem of closed discrete curves. Electron. Notes Discrete Math. **12**, 325–336 (2003)
22. Kim, C.E.: Digital convexity, straightness, and convex polygons. IEEE Trans. Pattern Anal. Mach. Intell. **4**, 618–626 (1982)
23. Gérard, Y.: Recognition of digital polyhedra with a fixed number of faces. In: Normand, N., Guédon, J., Autrusseau, F. (eds.) Discrete Geometry for Computer Imagery, pp. 415–426. Springer International Publishing, Cham (2016)
24. Gerard, Y.: About the decidability of polyhedral separability in the lattice $\mathbb{Z}^d$. J. Math. Imaging Vis. **59**, 52–68 (2017)
25. Gérard, Y.: Recognition of digital polyhedra with a fixed number of faces is decidable in dimension 3. In: Proceedings of Discrete Geometry for Computer Imagery—20th IAPR International Conference, DGCI 2017, Vienna, Austria, September 19–21, 2017, pp. 279–290 (2017)
26. Crombez, L., da Fonseca, G.D., Gérard, Y.: Efficient algorithms to test digital convexity. In: Couprie, M., Cousty, J., Kenmochi, Y., Mustafa, N. (eds.) Discrete Geometry for Computer Imagery, pp. 409–419. Springer International Publishing, Cham (2019)
27. Colbourn, C.J., Simpson, R.: A note on bounds on the minimum area of convex lattice polygons. Bull. Aust. Math. Soc. **45**(2), 237–240 (1992)
28. Žunić, J.: Notes on optimal convex lattice polygons. Bull. Lond. Math. Soc. **30**(4), 377–385 (1998)
29. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**, 469–483 (1996)
30. Greenfield, J.S.: A proof for a quickhull algorithm. Technical report, Syracuse University (1990)
31. Megiddo, N.: Linear programming in linear time when the dimension is fixed. JACM **31**(1), 114–127 (1984)
32. Edelsbrunner, H., Preparata, F.: Minimum polygonal separation. Inf. Comput. **77**(3), 218–232 (1988)
33. Coeurjolly, D., Gérard, Y., Reveillès, J., Tougne, L.: An elementary algorithm for digital arc segmentation. Discrete Appl. Math. **139**(1–3), 31–50 (2004)
34. Roussillon, T., Lachaud, J.-O.: Delaunay properties of digital straight segments. In: Debled-Rennesson, I., Domenjoud, E., Kerautret, B., Even, P. (eds.) Discrete Geometry for Computer Imagery, pp. 308–319. Springer, Berlin (2011)
35. Andrews, G.E.: A lower bound for the volumes of strictly convex bodies with many boundary points. Trans. Am. Math. Soc. **106**, 270–279 (1963)
36. Bárány, I.: Extremal problems for convex lattice polytopes: a survey. Contemp. Math. **453**, 87–103 (2008)

Loïc Crombez is a PhD student at Université Clermont Auvergne, LIMOS since 2017, advised by Guilherme D. da Fonseca and Yan Gérard. His main work is in computational geometry and digital geometry, more specifically on convex lattice sets. Before starting his PhD he was a Maths teacher from 2015 to 2017, and a software developer from 2012 to 2015.

Guilherme D. da Fonseca since 2019, he is a computer science professor at Aix-Marseille University, doing my teaching at the computer science IUT in Arles and my research at the ACRO team of the LIS lab. From 2015 to 2019, I was at the IUT of Université Clermont Auvergne, doing my research at LIMOS. I got my PhD at the University of Maryland, College Park in 2007, advised by David Mount.

Yan Gerard is a graduate student of the French Ecole Polytechnique. He obtained his PhD in computer science in 1999 after having started to work with his PhD supervisor J-P Reveilles and Maurice Nivat. He is mainly interested in Digital Geometry, a field of computational geometry and mathematics dealing with the finite subsets of Z^d. His work is focused especially on convex lattice sets. He has also contributed to Computer Vision and Discrete Tomography.