Mapping Points to the Grid with Bounded Hausdorff Distance

Maarten Löffler and Jérôme Urhausen

March 29, 2024

Mapping Points to the Grid with Bounded Hausdorff Distance

Maarten Löffler[*] Jérôme Urhausen[*]

Abstract

We consider the problem of representing a set of m points using disjoint pixels on a grid with bounded Hausdorff distance. We prove that optimizing the problem is NP-complete. Additionally, we present a constant factor approximation algorithm with running time in $O(m^2 \log \delta^* / \log m)$, where δ^* is the Hausdorff distance in an optimal solution, as well as a slower algorithm with a constant additive error.

1 Introduction

The field of *digital geometry* concerns itself with the representation of geometric objects using pixels on a grid while preserving geometric properties. Examples are mapping convex regions to a similar-looking orthoconvex set of pixels or mapping lines to chains of pixels that still only intersect at most once. Digital geometry finds application in image processing and storage. For a survey, see Klette and Rosenfeld [15, 16].

More recently, error bounds under the Hausdorff distance have been studied. Chun et al. [7] investigate the problem of digitizing rays originating in the origin to digital rays such that certain properties are satisfied. They show that rays can be represented on the $n \times n$ grid in a consistent manner with Hausdorff distance $O(\log n)$. This bound is tight in the worst case. By ignoring one of the consistency conditions, the distance bound improves to $O(1)$. Their research is extended by Christ et al. [5] to line segments (not necessarily starting in the origin), who obtain the logarithmic distance bound in this case as well. A possible extension to curved rays was developed by Chun et al. [6]. Other results with a digital geometry flavor within the algorithms community are those on snap rounding [8, 11, 14], integer hulls [1, 13], and discrete schematization [17].

The present submission is inspired by two recent papers: *Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance*, by Bouts, Kostitsyna, van Kreveld, Meulemans, Sonke and Verbeek [3] and *Mapping Multiple Regions to the Grid with Bounded Hausdorff Distance* by van der Hoog, van de Kerkhof, van Kreveld, Löffler, Staals, Urhausen and Vermeulen [20]. Intuitively the problem discussed in these papers is: for a given set of regions find a set of pixels from the unit grid that best represents the input (see Figure 1).

[*]Department of Information and Computing Sciences, Utrecht University, {m.loffler|j.e.urhausen}@uu.nl. This work is partially supported by the Dutch Research Council (NWO) under project number 612.001.651, 614.001.504 and 628.011.005.

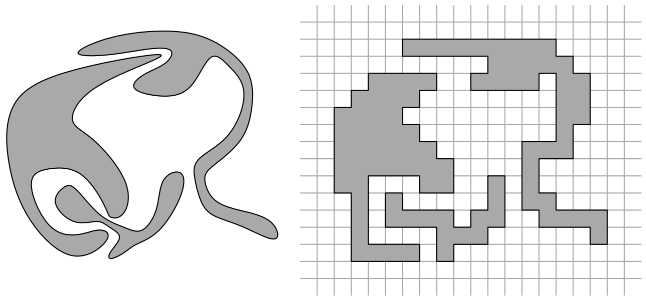


Figure 1: An example of a valid mapping of regions to grid polygons.

In the following we use the notation $H'(R, P)$ for the maximum between the Hausdorff distance between the sets R and P and the Hausdorff distance between their boundaries ∂R and ∂P. Amongst others, Bouts et al. [3] show that for a given connected region R, one can find a simply connected grid polygon P in the unit grid such that the Hausdorff distance $H'(R, P)$ is at most a constant. Note that this result holds, no matter the resolution of R. On the other hand they show that, given a region R, it is NP-hard to find the grid polygon P that minimizes the Hausdorff distance $H'(R, P)$.

Van der Hoog et al. [20] extend this concept to multiple regions. Whereas the result from [3] was extended to two regions, for three or more regions there is no constant upper bound on the Hausdorff distance between the regions and any simply connected grid polygons. Nonetheless, they show that, if the regions are m β-fat convex regions, one can construct a set of disjoint grid polygons within Hausdorff distance H' at most $O(\sqrt{m})$, for β constant. This is tight in the worst case. Note that points are β-fat convex regions, for any β. Their last result is that for m convex regions, one can construct a set of orthoconvex disjoint grid polygons within Hausdorff distance of $O(m)$, which is again tight in the worst case.

Computation. Previous work focuses on the *existence* of solutions with bounded Hausdorff distance, but not on their *efficient computation*. When considering the question of efficiency, we are faced with some additional modeling questions. The two main ones are:

1. How do we locate input features (vertices or edges) on the grid?

2. How do we compactly represent sets of pixels that correspond to output regions?

Regarding question (1), we note that traditionally, geometric algorithms are analysed in the *Real RAM* computation model. In this model, one may work with arbitrary real numbers, but certain natural operations are not available; in particular the *floor* operation is known to be problematic [2]. In the context of digital geometry, where we have a natural underlying grid, and the whole objective is to map objects to a grid, such a restriction seems not entirely reasonable. In this work, we will move away from the Real RAM model and assume that the input coordinates are all polynomial in the input size—in other words, the *bit complexity* is logarithmic—and that the floor function is available. (Note that under our bit complexity assumption, if desired, the floor function can also be implemented in logarithmic time on a Real RAM.)

Regarding question (2), we note that when mapping large regions (in size, not in description complexity) to a grid, an explicit representation of the output listing precisely which pixels are and which are not part of a set would necessarily be large as well, and may be unrelated to the input complexity. Alternatively, one could compactly represent output regions by providing only the coordinates of *vertices* and interpolating boundary edges onto the grid. Given the complexity of mapping lines to the grid, however, it is not entirely clear how to do this in a consistent way. In this work, we make a first step towards understanding the computational aspect of the question by focusing on *point* regions. For a single point region and constant Hausdorff distance, the output is necessarily a set of only a constant number of pixels, and thus we avoid the issue.

To summarize, in the present submission, we make the following assumptions:

- The input is a set $\mathcal{R}$ of m points in $\mathbb{R}^2$, with a polynomial upper bound on the coordinate sizes; that is, for every point $R \in \mathcal{R}$ we have $-f(m) < x_R < f(m)$ and $-f(m) < y_R < f(m)$ for some polynomial function f.

- We have access to a *floor* operation, which can provide us with the integer part of any real number in the range $[-f(m), f(m)]$.

Related Work. Testing whether there exists a set of pixels within Hausdorff distance δ from $\mathcal{R}$ can be seen as a problem where we are given a set of squares of (L_1)-radius δ, and are asked to place a set of unit squares such that each square touches an input square. The problem of placing unit squares in the neighbourhood of points can be viewed as dual to the problem of placing points in squares: if we shrink the (L_1)-radius of the squares-to-be-placed by $\frac{1}{2}$ and we grow the δ-neighbourhoods of the points also by $\frac{1}{2}$, a valid solution where points are placed at integer coordinates at distance at least 2 from each other corresponds exactly to a valid solution to our problem.

The problem of placing points in squares such that the points are not too close to each other has been introduced under the name of *distant representatives* [10] and was later also studied in the context of data *imprecision* [18]. Fiala et al. [10] prove that the problem is NP-hard (both for disk and square regions), and Cabello [4] proposes a constant-factor approximation algorithm.

We note that our problem is essentially different, since for us, valid placements are restricted to a *discrete* set of points (the unit grid). Neither the hardness proof nor the algorithmic result carry over directly to this discrete setting.

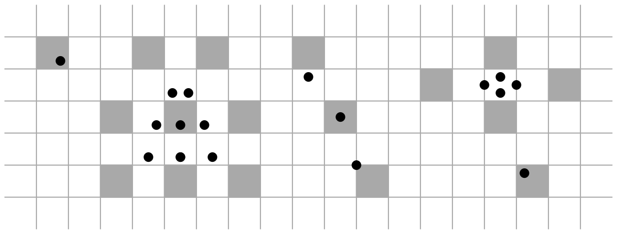


Figure 2: An example of a valid mapping of points to pixels on the grid.

Contribution. In the present submission, we study the computational question of mapping point sets to disjoint pixels on a unit grid with small Hausdorff distance, as visualized in Figure 2. Van der Hoog et al. [20] observed that in general the solution constructed with their algorithms might yield a "visually unfortunate" output. Formally, the algorithm might yield a solution with high Hausdorff distance, even in the case where the optimal solution has constant Hausdorff distance. In Section 2, we show that finding the solution with minimal Hausdorff distance to a given set of regions is NP-complete, even if the regions are just points. Then, in Section 3, we present an approximation algorithm for points that produces a solution with Hausdorff distance at most $2\sqrt{2}(\lceil\delta^*\rceil + 1) \leq 6\sqrt{2}\delta^*$ and has a running time of $O(m^2 \log \delta^* / \log m)$, where δ^* is the maximal Hausdorff distance in an optimal solution. Finally, we present a second algorithm which produces a solution with Hausdorff distance at most $\lceil\delta^*\rceil + \sqrt{2}$ in $O(\delta^{*4} m^2 / \log m)$ time.

Notation and Definitions. We denote by Γ the (infinite) unit grid in two dimensions, whose unit squares are referred to as *pixels*. The *(symmetric) Hausdorff distance* between two sets $A, B \subset \mathbb{R}^2$ is defined as $H(A,B) = \max\{\max_{a \in A}(\min_{b \in B}(|ab|)), \max_{b \in B}(\min_{a \in A}(|ab|))\}$, where $|ab|$ is the distance between the points a and b.

Let $\mathcal{R} = \{R_1, R_2, \ldots R_m\}$ be a set of m points in the plane. In this paper, we treat the problem on how to assign a pixel $P_i \in \Gamma$ to each point $R_i \in \mathcal{R}$ such that different pixels do not meet in any edge or vertex of the grid. This is consistent with the problem definition from [20]. We call the set $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ of such pixels a *valid mapping* for $\mathcal{R}$. Our goal is to find a valid mapping $\mathcal{P}$ that minimizes $\max_{i \in \{1,\ldots,m\}}\{H(R_i, P_i)\}$. See Figure 2 for an example.

Note that in contrast to [3] and [20], we disregard the Hausdorff distance between the boundaries $H(\partial R_i, \partial P_i)$ because we have $H(\partial R_i, \partial P_i) = H(R_i, P_i)$, for convex R_i and P_i.

2 NP-completeness

Bouts et al. [3] proved that for a single simply connected region R it is NP-complete to test if there is a grid-polygon within Hausdorff distance 1/2. We extend this result to multiple point-regions. Formally, we show that for a set of points $\mathcal{R}$ it is NP-complete to test if there is a valid mapping within Hausdorff distance $\sqrt{2}$. Our proof is inspired both by the construction of Bouts et al. [3] and the proof by Fiala et al. [10] for a similar problem in a continuous setting.

We first prove containment in NP. For each point there are at most 9 options to place the corresponding pixel. An oracle can guess the correct placement and then just has to test that no two pixels share a common grid vertex.

We now show that the problem is NP-hard. We reduce from the NP-complete problem monotone rectilinear planar 3-Sat [9].

Rectilinear monotone planar 3-SAT. Input: a 3-Sat formula with only all positive or all negated variables per clause, embedded as a graph with rectilinear, non-crossing edges. The set of vertices consists of variable-, split- and clause-vertices; variable-vertices are drawn on a horizontal line that no edge crosses; clause-vertices for positive (negative) clauses are drawn above (below) this line; clauses are connected with the variables they contain with an edge or a path of edges and split-vertices. Output: "Yes" if there exists a satisfying assignment for the variables, "No" otherwise.

Such a 3-Sat formula embedded as a graph is illustrated in Figure 3. Without loss of generality we can assume that the embedded graph has the following additional properties: edges have at most one bend, each variable-vertex v has degree at most 2 and the incident

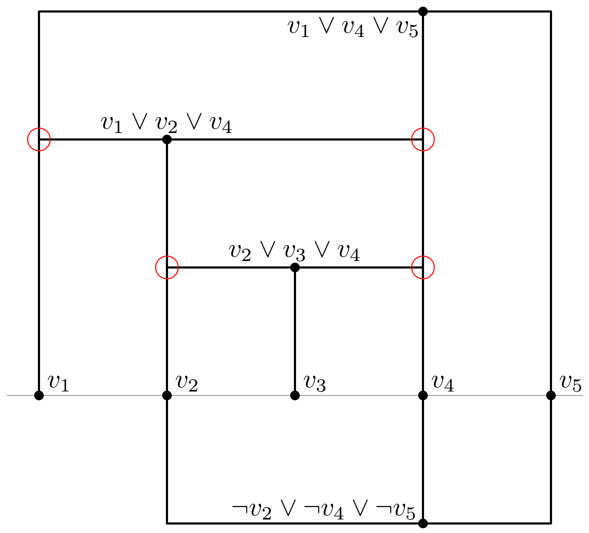


Figure 3: An example of the embedded formula $(v_2 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee v_4) \wedge (v_1 \vee v_4 \vee v_5) \wedge (\neg v_2 \vee \neg v_4 \vee \neg v_5)$. The split vertices are highlighted in red.

edges are vertical at v, split-vertices w have degree 3 and only one incident edge is horizontal at w, and for each variable a, all split-vertices corresponding to a are vertically aligned with the variable-vertex v_a of a.

For a given monotone rectilinear planar 3-Sat instance that is embedded as described above, let $\mathcal{G}$ be a drawing of the embedding. Without loss of generality we assume that $\mathcal{G}$ is drawn on the unit grid and that the horizontal line containing all variables is the x-axis. We scale $\mathcal{G}$ such that each vertex is on an even grid vertex $(2x, 2y)$ and such that the distance between any two vertices, between any vertex and any bend, and between any two non-incident edges is at least 8.

Construction. We create a set of points $\mathcal{R}$. We only place points on grid vertices. In the end, we ask the question whether one can place pixels within Hausdorff distance $\sqrt{2}$ from the points of $\mathcal{R}$, that is, we ask the question if for each point in $\mathcal{R}$, we can choose one of the four adjacent pixels so that no two chosen pixels of different points share a common vertex. We say a point R_i has a *top-left* (*top*, *top-right*, ...) *pixel* if P_i is to the top-left (top, top-right, ...) of R_i.

These following two observations are the main tools for the construction of the gadgets, depicted in Figure 4. When two horizontally aligned points are at distance 1, the leftmost (rightmost) point has a left (right) pixel. For two horizontally aligned points at distance 2, if the leftmost (rightmost) point has a right (left) pixel, the rightmost (leftmost) point has a right (left) pixel, too. This is symmetric for vertically aligned points.

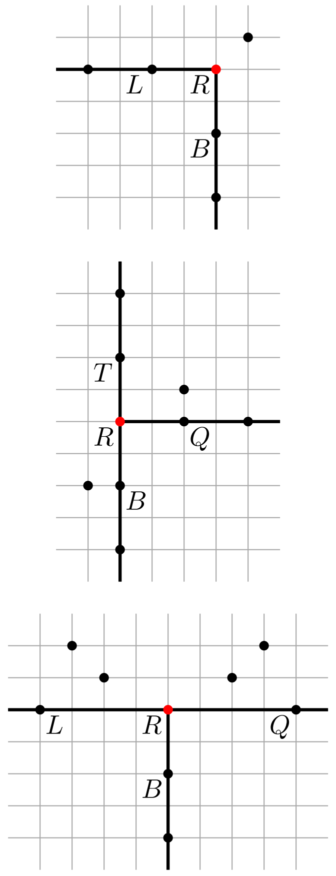


Figure 4: The bend, split and clause gadgets. The point R is highlighted in each gadget.

Variable. We first place a point $R \in \mathcal{R}$ on each even grid vertex that is intersected by the drawing. For each variable a in G, there is a point $R_a = (2x, 0)$ in $\mathcal{R}$ where the variable-vertex v_a is drawn. We call that point R_a the *indicator* of a. Intuitively, if R_a has a bottom (top) pixel, a is true (false). If the point $(2x, 2)$ has a bottom (top) pixel we say it has a pixel *toward the variable* (*away from the variable*). Symmetrically, if the point $(2x, -2)$ has a top (bottom) pixel we say it has a pixel *toward the variable* (*away from the variable*). This concept propagates throughout the points corresponding to a.

Bend. Let $R = (2x, 2y) \in \mathcal{R}$ be a point at the corner of a bend of an edge e. We assume e connects a vertex to the left of R with a vertex below R. The other cases are symmetric. Thus, the points $L = (2x - 2, 2y)$ and $B = (2x, 2y - 2)$ are in $\mathcal{R}$. We add another point $(2x + 1, 2y + 1)$ to $\mathcal{R}$. Now, if L has a right pixel, B has a bottom pixel and if B has a top pixel, L has a left pixel.

Split. Let $R = (2x, 2y) \in \mathcal{R}$ be a point at a split-vertex. We assume that the horizontal edge incident to R is to its right and that the split vertex is above the x-axis and therefore its corresponding variable-vertex. Thus, the points $T = (2x, 2y + 2)$, $Q = (2x + 2, 2y)$ and $B = (2x, 2y - 2)$ are in $\mathcal{R}$. The other cases are symmetric. We add the points $(2x + 2, 2y + 1)$ and $(2x - 1, 2y - 2)$ to $\mathcal{R}$. If T has a bottom pixel or if Q has a left pixel, B has a bottom pixel. That is, if a point on the top or right edges has a pixel toward the variable, the points on the bottom edge also have pixels toward the variable.

Clause. Let $R = (2x, 2y) \in \mathcal{R}$ be a point at a clause-vertex. We call R the *clause-point*. We assume the three edges connect to the left, right and bottom of R respectively, else the situation is symmetric. As the distance between two gadgets is least 8, the points $(2x - 2, 2y)$, $L = (2x - 4, 2y)$, $(2x + 2, 2y)$, $Q = (2x + 4, 2y)$ and $B = (2x, 2y - 2)$ are in $\mathcal{R}$. We move the points $(2x - 2, 2y)$ and $(2x + 2, 2y)$ to $(2x - 2, 2y + 1)$ and $(2x + 2, 2y + 1)$ and add two points $(2x - 3, 2y + 2)$ and $(2x + 3, 2y + 2)$ to $\mathcal{R}$. It follows that if the clause-point R has a top-left (top-right, bottom) pixel, L (Q, B) has a left (right, bottom) pixel. That means that the points on at least one of the incident edges have pixels toward the variable.

Put together the points $\mathcal{R}$ and a valid mapping $\mathcal{P}$ are shown in Figure 5.

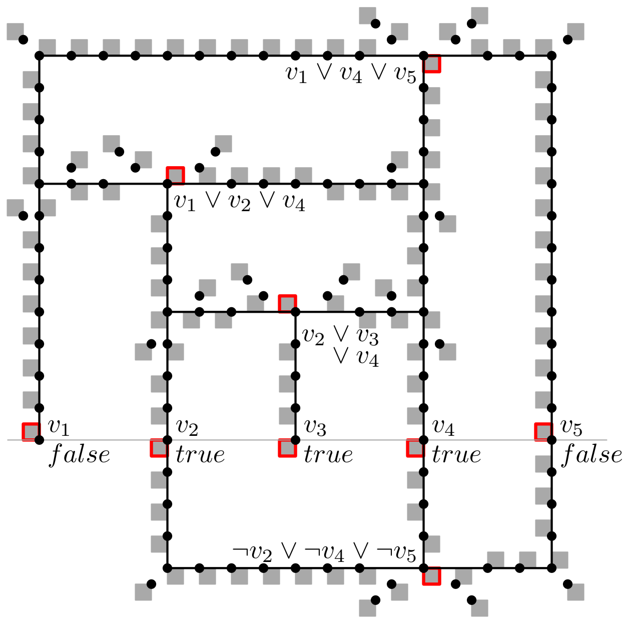


Figure 5: The complete construction of the NP-hardness reduction. The pixels corresponding to indicators or to clause-points are highlighted.

Proof of correctness. Let A be an assignment of variables to $\{true, false\}$ such that the 3-SAT formula is satisfied. For each variable a, if a is true, first, we give the indicator R_a a bottom pixel. Second, we give each point $R \in \mathcal{R}$ that is on an edge corresponding to a a pixel toward (away from) the variable, if p is above (below) the x-axis. For each variable assigned the value false we do the inverse. As in each positive (negative) clause there is a variable that is assigned to true (false), each clause-point can place its pixel in one of the four adjacent spots. So overall there is a valid mapping such that the Hausdorff distance between a point and its corresponding pixel is at most $\sqrt{2}$.

Inversely, let there be a set of pixels such that the Hausdorff distance between any point and its corresponding pixel is at most $\sqrt{2}$. For each variable a, if the indicator R_a has a bottom (top) pixel, we set a to true (false). We now prove that in each positive (negative) clause there is a variable that is assigned to true (false). Let c be a positive clause such that the incident edges are on the left, right and bottom of the clause point R of c. The other cases are symmetric. If R has a top-left (top-right, bottom) pixel, we know that the points on the left (right, bottom) edge have pixels toward the variable. Let e be an edge incident to R whose points have pixels toward the variable. If e connects to a split-vertex w, the points on the vertical edge e' incident at the bottom at w also have pixels toward the variable. We then set $e = e'$. This repeats until we have an edge e that connects to the variable-vertex of a. It follows that the indicator R_a has a bottom pixel and a has been assigned the value true. The theorem follows.

Theorem 1 *If $\mathcal{R}$ is a set of m points, it is NP-complete to decide whether there exists a valid mapping such that for each point $R_i \in \mathcal{R}$ with corresponding pixel P_i, we have $H(R_i, P_i) \leq \sqrt{2}$.*

3 Approximation Algorithms

We now turn our attention to approximation. We start by making some observations. Clearly, the optimal Hausdorff distance δ^* for any instance is at least $\frac{1}{2}\sqrt{2}$, since the distance is taken between a point and (at least one) unit square. Therefore, a constant *additive* approximation in this case automatically translates to a constant *multiplicative* approximation. We also note that, due to the discrete nature of the output, we cannot hope to do better than a constant factor approximation.

To illustrate the complexity of the problem, in Section 3.1 we first discuss some natural ideas which do *not* lead to a working approximation. Then, in Section 3.2, we then present an algorithm that achieves a Hausdorff distance of at most $2\sqrt{2}\lceil\delta^*\rceil + 2\sqrt{2}$. In Section 3.3 we show how to improve the approximation to $\lceil\delta^*\rceil + \sqrt{2}$, at the cost of a slower runtime.

3.1 A First Attempt

As discussed in the introduction, Cabello [4] presents a constant factor approximation algorithm for placing n points into respective discs or squares. A first approach could be to dualize our problem and directly run their algorithm to place a set of points—however, we would have no guarantee the points are placed at integer coordinates. We would have to snap the points to the grid before translating them back to squares. The Hausdorff distance itself would only increase by at most $\frac{1}{2}\sqrt{2}$ in this way, which would still result in a constant-factor approximation, albeit with a slightly higher constant. However, the snapping procedure could also result in touching or even overlapping pixels, so the solution would not necessarily be valid.

Another approach would be to find a subdivision of the grid into cells such that for each cell all the points contained in it can be assigned separate pixels in that cell. If the minimal size of the cells depends only on the Hausdorff distance between the points and an optimal valid mapping, this approach could lead to an approximation algorithm. Formally, let Γ_k be a coarsening of the grid Γ whose cells have $k \times k$ pixels. We call these cells *superpixels*. The following lemma proves that this approach does not work either.

Lemma 2 *There is a set of points $\mathcal{R}$ and a point $R \in \mathcal{R}$, such that (1) there is a valid mapping $\mathcal{P}$ within Hausdorff distance at most 3; (2) for any superpixel with side length s that contains R and at most $\lfloor\frac{s}{2}\rfloor^2$ points from $\mathcal{R}$, we have $s \in \Omega(|\mathcal{R}|)$.*

Proof. A pixel is a set $[i, i+1] \times [j, j+1]$, for integers i, j. We define the set of points $\mathcal{R} = \left\{R = \left(\frac{1}{4}, \frac{1}{4}\right)\right\} \cup \left\{\left(2i + \frac{1}{2}, 2j + \frac{1}{2}\right), \left(2i + \frac{1}{2}, -2j - \frac{1}{2}\right), \left(-2i - \frac{1}{2}, -2j - \frac{1}{2}\right), \left(-2i - \frac{1}{2}, 2j + \frac{1}{2}\right) \mid i, j \in \{0, \ldots, n\}\right\}$, as shown in Figure 6. Let the four endpoints of a superpixel S containing R be (a, b), $(-c, b)$, $(-c, -d)$ and $(a, -d)$, with $a, b \geq 1; c, d \geq 0$. The side length of the superpixel is $s = a + c = b + d$. If $s \leq n$, the superpixel S contains $1 + \lceil a/2\rceil\lceil b/2\rceil + \lceil c/2\rceil\lceil b/2\rceil + \lceil c/2\rceil\lceil d/2\rceil + \lceil a/2\rceil\lceil d/2\rceil > \lfloor s/2\rfloor^2$ points. □

3.2 A Constant Factor Approximation Algorithm

We present an algorithm that, for a given set of points $\mathcal{R}$ in $\mathbb{R}^2$ determines a valid mapping $\mathcal{P}$, such that the Hausdorff distance between a point and its corresponding pixel is at most $2\sqrt{2}(\lceil\delta^*\rceil + 1)$, for δ^* being the minimal possible Hausdorff distance between $\mathcal{R}$ and any valid mapping $\mathcal{P}$.

For a coarsening Γ_k of the grid Γ, let $\mathcal{S}_k$ be the set of superpixels that either contain a point in $\mathcal{R}$ or are adjacent to a superpixel that does.

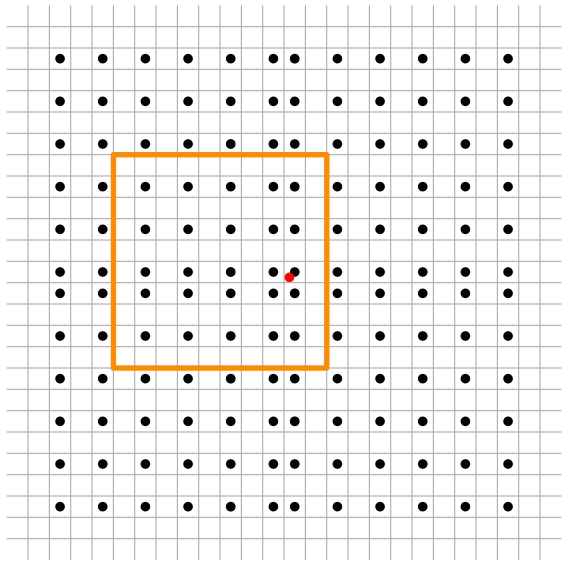


Figure 6: Any superpixel with side length i containing the red point R contains more than $\lfloor i/2 \rfloor^2$ points. For example, the yellow superpixel has side length 10 and contains 26 points.

Observation 1 *Let $x \geq \delta^*$ and $x \in \mathbb{N}$ even. Then we know that for each point $R_i \in \mathcal{R}$ in a superpixel $S \in \mathcal{S}_x$, the pixel P_i is in S or in one of the 8 superpixels adjacent to S. Additionally, each superpixel contains at most $(x/2)^2$ pixels that are disjoint.*

Building on that idea, we create the following test $f(\cdot)$. For an even number $i \in \mathbb{N}$, we want $f(i) =$ true if for each coarsening Γ_i, there exists an assignment $g : \mathcal{R} \to \mathcal{S}_i$ of points to superpixels such that:

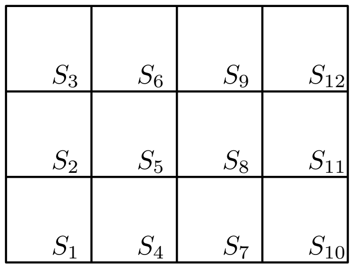
1. $\forall R \in \mathcal{R}$, $g(R)$ is the superpixel containing R or one of the eight adjacent ones;

2. $\forall S \in \mathcal{S}_i$, there are at most $(i/2)^2$ points R with $g(R) = S$.

We call such an assignment g a *correct assignment*. Otherwise, if for each coarsening Γ_i no correct assignment can be found, we want $f(i) =$ false. If a correct assignment g can only be found for some coarsenings, $f(i)$ can either be true or false. We define $f(0) =$ false.

Binary Search. We use exponential and binary search to find an even number $i \in \mathbb{N}$ with $f(i) =$ true and $f(i-2) =$ false. We start at $i = 2$. We iterate calculating $f(i)$: if $f(i) =$ false, we double i and continue, else we stop. Then we binary search normally. Note that from Observation 1 we get $f(i) = true$, for $i \geq \delta^*$. Therefore, this binary search algorithm results in a number $I \leq \lceil \delta^* \rceil + 1$ and has a running time of $O(F \times \log \delta^*)$, where F the the time to run the test $f(\cdot)$.

Test. The calculation of $f(i)$ proceeds as follows. We use a flow algorithm [12] to determine if a correct assignment g exists. We choose a coarsening Γ_i and create a directed acyclic graph $G = (V, E)$ as illustrated in Figure 7. We set $V = \{s, t\} \cup \{S_{in}, S_{out} \mid S \in \mathcal{S}_i\}$ as the set of vertices. We define $(a, b, c) \in E$ as the edge between the vertices $a \in V$ and $b \in V$ with capacity $c \in \mathbb{N} \cup \{\infty\}$. We set $E = \{(s, S_{in}, |S \cap \mathcal{R}|) \mid S \in \mathcal{S}_i\} \cup \{(S_{in}, S'_{out}, \infty) \mid S, S' \in \mathcal{S}_i \wedge (S = S' \vee S \text{ adjacent to } S')\} \cup \{(S_{out}, t, (i/2)^2) \mid S \in \mathcal{S}_i\}$ as the set of edges.



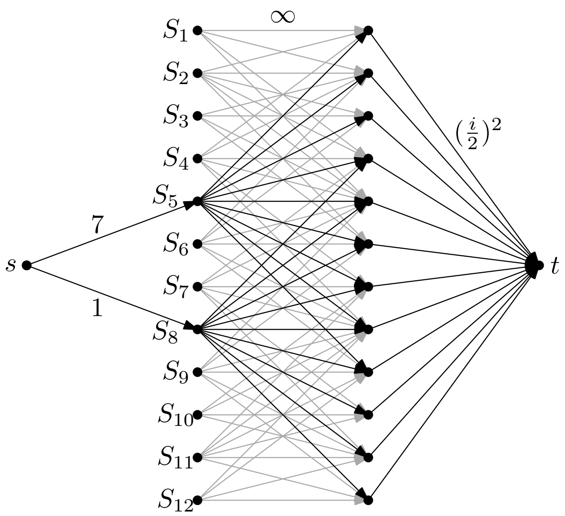


Figure 7: A set $\mathcal{R}$ of 8 points and the corresponding graph G produced by the algorithm in Section 3.2. The edges are annotated with their respective capacities. Edges with capacity 0 are omitted; edges starting at $(S_i)_{in}$ are grayed out if S_i is empty.

We then calculate a maximal flow from s to t. We can assume that the flow in each edge is a natural number. As $|E| \in O(|V|)$, the flow algorithm runs in $O(m^2/\log m)$ time [19] because $|V| \in O(m = |\mathcal{R}|)$. If G admits a flow from s to t with flow rate $|\mathcal{R}|$, the flow induces a correct assignment g as follows: we repeat the following for every superpixel $S \in \mathcal{S}_i$. Let $\{S_1, \ldots, S_9\} = \{S' \mid S' \in \mathcal{S}_i \wedge (S = S' \vee S \text{ adjacent to } S')\}$ be the set containing S and the superpixels adjacent to S. Let $U_1 \cup \cdots \cup U_9 = S \cap \mathcal{R}$ be any partition of the points in S, where, for $j, k \in \{1, \dots, 9\}$, we have $j \neq k \implies U_j \cap U_k = \emptyset$ and $|U_j|$ is the flow from S_{in} to $(S_j)_{out}$. For each $j \in \{1, \dots, 9\}$ and each point $R \in U_j$, we set $g(R) = S_j$. This results in a correct assignment g.

Thus, if G admits a flow from s to t with flow rate $|\mathcal{R}|$, we set $f(i) = $ true. Otherwise $f(i) = $ false. This test performs as requested and has a running time of $O(m^2/\log m)$.

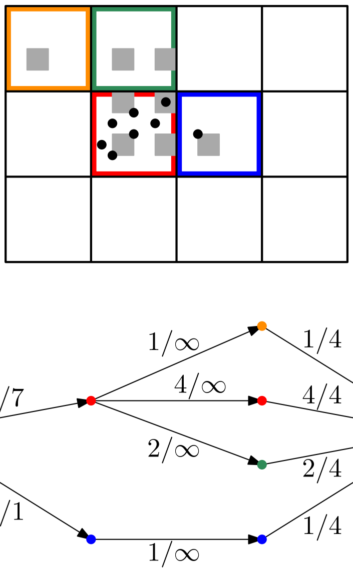


Figure 8: A set $\mathcal{R}$ of 8 points and the graph G with a maximal s-t-flow where edges with flow 0 are omitted, produced by the algorithm in Section 3.2. The superpixels and their respective vertices in G are color-coded. The pixels $\mathcal{P}$ induced by the flow are show in gray.

Placing the Pixels. Let I be the even number that is the result from the binary search, that is, $f(I) = $ true and $f(I-2) = $ false. Let now Γ_I be a coarsening and let g be a correct assignment, calculated by the test $f(I)$. We place the pixels $\mathcal{P}$ as shown in Figure 8: for each superpixel $S \in \mathcal{S}_I$, let $\{R_1^S, R_2^S, \dots\} = \{R \in \mathcal{R} \mid g(R) = S\}$ be the points assigned to S. We set the pixel corresponding to R_j^S as $\left(2\left(j \bmod \frac{I}{2}\right), 2\left\lceil \frac{2j}{I} \right\rceil\right)_S$, where $(1,1)_S$ ($(I,1)_S$, $(I,I)_S$) is the pixel on the bottom-left (bottom-right, top-right) of S. That way no two pixels in $\mathcal{P}$ touch and for each point R its corresponding pixel is in the superpixel assigned to R. It follows that the Hausdorff distance between a point and its corresponding pixel is at most $2\sqrt{2}I$. Since I is an even number, $I \leq \lceil \delta^* \rceil + 1$, and $\delta^* \geq \sqrt{(2)}/2$, we get:

Theorem 3 *Given a set of m points $\mathcal{R}$, we can determine a valid mapping $\mathcal{P}$ such that for each point R_i with corresponding pixel P_i, we have $H(R_i, P_i) \leq 2\sqrt{2}(\lceil \delta^* \rceil + 1) \leq 6\sqrt{2}\delta^*$ in $O(m^2 \log \delta^* / \log m)$ time.*

3.3 An Algorithm with Constant Additive Error

Contrary to the constant factor algorithm presented in the previous section, we now present an approximation algorithm with *only* a constant additive error: that is, for a given set $\mathcal{R}$, we determine a valid mapping $\mathcal{P}$, with $H(R_i, P_i) \leq \delta^* + c$ for each $R_i \in \mathcal{R}$, where c is a constant and δ^* is the minimal possible Hausdorff distance between $\mathcal{R}$ and any valid mapping $\mathcal{P}$. The algorithm uses similar ideas to the algorithm in Section 3.2. Let Γ_2 be a coarsening, that is each superpixel only contains 4 pixels forming a square. It follows that when placing a pixel in each superpixel, the pixels are disjoint. We define $H^*(R, S) = \min_{\text{pixel } P \in S} H(R, P)$ as the Hausdorff distance between the point $R \in \mathcal{R}$ and its closest pixel in the superpixel S. For $i \in \mathbb{N}$, let $\mathcal{S}_i$ be the set of superpixels S, such that there is a point $R \in \mathcal{R}$ with $H^*(R, S) \leq i$. Note that here, the size of the set $\mathcal{S}_i$ is dependent on i^2.

Observation 2 *Let $x \geq \delta^*$ and $x \in \mathbb{N}$. For a given valid mapping $\mathcal{P}$ that minimizes the Hausdorff distance, for each point $R_i \in \mathcal{R}$, the pixel $P_i \in \mathcal{P}$ is in a superpixel $S \in \mathcal{S}_x$, with $H^*(R_i, S) \leq x$. Additionally, each superpixel contains at most one pixel.*

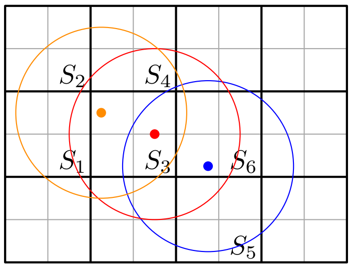
The observation leads us to create the following test $f(\cdot)$: we want $f(i) = $ true, if there exists an assignment $g : \mathcal{R} \to \mathcal{S}_i$ of points to superpixels such that:

1. $\forall R \in \mathcal{R}$, $H^*(R, g(R)) \leq i$;
2. $\forall S \in \mathcal{S}_i$, there is at most one R with $g(R) = S$.

We again call such an assignment g a *correct assignment*. Otherwise, we want $f(i) = $ false. Note that $f(0) = $ false.

Binary Search. Similarly to Section 3.2, we use exponential and binary search to find a number $i \in \mathbb{N}$ with $f(i) = $ true and $f(i-1) = $ false. We start at $i = 1$. We iterate calculating $f(i)$: if $f(i) = $ false, we double i and continue, else we stop. Then we binary search normally. Due to Observation 2, we have $f(i) = $ *true*, for $i \geq \delta^*$. This binary search algorithm results in a number $I \leq \lceil \delta^* \rceil$ and has a running time of $O(F \times \log \delta^*)$, where F the the time to run the test $f(\cdot)$.

Test. The calculation of $f(i)$ proceeds very similarly to Section 3.2. We use a flow algorithm [12] to determine if a correct assignment g exists. We create a directed acyclic graph $G = (V, E)$, as illustrated in Figure 9. We set $V = \{s, t\} \cup \mathcal{R} \cup \mathcal{S}_i$ as the set of vertices. We define $(a, b) \in E$ as the edge between the vertices $a \in V$ and $b \in V$ with capacity 1. We set $E = \{(s, R) \mid R \in \mathcal{R}\} \cup \{(R, S) \mid R \in \mathcal{R}, S \in \mathcal{S}_i \wedge H^*(R, S) \leq i\} \cup \{(S, t) \mid S \in \mathcal{S}_i\}$ as the set of edges.



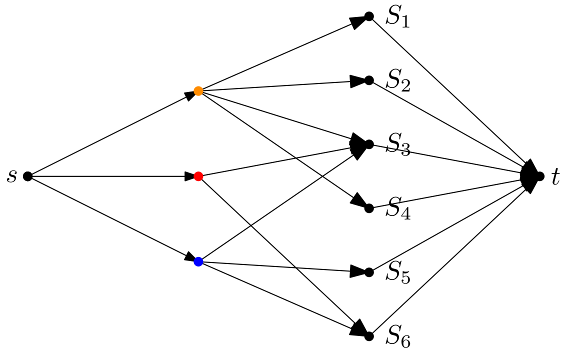


Figure 9: A small set $\mathcal{R}$ of 3 points and the graph $G = (V, E)$ produced by the algorithm in Section 3.3 for $i = 2$. All edges have a capcity of 1. The points in $\mathcal{R}$ and their respective vertices in G are color-coded.

We then calculate a maximal flow from s to t. We can assume that the flow in each edge is a natural number. As $|V|, |E| \in O(i^2 m)$, the flow algorithm runs in $O(i^4 m^2/(\log i \log m))$ time [19]. If G admits a flow from s to t with flow rate $|\mathcal{R}|$, the flow induces a correct assignment g as follows: for each point $R \in \mathcal{R}$, there is exactly one superpixel $S \in \mathcal{S}_i$ where the edge (R, S) has flow 1. We set $g(R) = S$. This results in a correct assignment g.

Thus, if G admits a flow from s to t with flow rate $|\mathcal{R}|$, we set $f(i) = \text{true}$. Otherwise $f(i) = \text{false}$. This test $f(i)$ performs as requested and has a running time of $O(i^4 m^2/(\log i \log m))$.

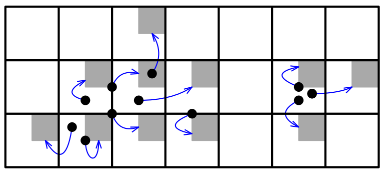


Figure 10: A set $\mathcal{R}$ of 11 points and the valid mapping $\mathcal{P}$ respecting a correct assignment g produced by the algorithm in Section 3.3.

Placing the Pixels. Let I be the number that is the result from the binary search, that is, $f(I) = \text{true}$ and $f(I-1) = \text{false}$. Let g be a correct assignment, calculated by the test $f(I)$. We place the pixels $\mathcal{P}$ as shown in Figure 10: for each superpixel $S \in \mathcal{S}_I$, if there is a point R assigned to S, we place the pixel corresponding to R in the top-right pixel of S. That way no two pixels in $\mathcal{P}$ touch and for each point R its corresponding pixel is in the superpixel assigned to R. It follows that the Hausdorff distance between a point and its corresponding pixel is at most $I + \sqrt{2}$. Since $I \leq \lceil \delta^* \rceil$, we have:

Theorem 4 *Given a set of m points $\mathcal{R}$, we can determine a valid mapping $\mathcal{P}$ such that for each point R_i with corresponding pixel P_i, we have $H(R_i, P_i) \leq \lceil \delta^* \rceil + \sqrt{2}$ in $O({\delta^*}^4 m^2/\log m)$ time.*

4 Conclusion

Overall we extended the previously known results on mapping regions to the grid with bounded Hausdorff distance. Where Bouts et al. [3] showed that, for a given set of regions, it is NP-hard to find the set of grid polygons that minimizes the Hausdorff distance H', even if for just one region, we show that this is hard, even if all regions are points. On the other hand, where van der Hoog [20] focused on worst-case tight algorithms, we present the first approximation algorithm for mapping regions to the grid.

An interesting open question is whether the concepts presented in this paper can be extended to an approximation algorithm that maps convex regions to the grid.

References

[1] E. Althaus, F. Eisenbrand, S. Funke, and K. Mehlhorn. Point containment in the integer hull of a polyhedron. In *Proceedings 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 929–933, 2004.

[2] L. Babai, B. Just, and F. Meyer auf der Heide. On the limits of computations with the floor function. *Inform. Comput.*, 78(2):99–108, 1988.

[3] Q. W. Bouts, I. Kostitsyna, M. van Kreveld, W. Meulemans, W. Sonke, and K. Verbeek. Mapping polygons to the grid with small Hausdorff and Fréchet distance. In *Proceedings 24th Annual European Symposium on Algorithms*, pages 22:1–22:16, 2016.

[4] S. Cabello. Approximation algorithms for spreading points. *Journal of Algorithms*, 62(2):49–73, 2007.

[5] T. Christ, D. Pálvölgyi, and M. Stojaković. Consistent digital line segments. *Discrete & Computational Geometry*, 47(4):691–710, 2012.

[6] J. Chun, K. Kikuchi, and T. Tokuyama. Consistent digital curved rays. In *Abstracts 34th European Workshop on Computational Geometry*, 2019.

[7] J. Chun, M. Korman, M. Nöllenburg, and T. Tokuyama. Consistent digital rays. *Discrete & Computational Geometry*, 42(3):359–378, 2009.

[8] M. de Berg, D. Halperin, and M. Overmars. An intersection-sensitive algorithm for snap rounding. *Computational Geometry*, 36(3):159–165, 2007.

[9] M. De Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications*, 22(03):187–205, 2012.

[10] J. Fiala, J. Kratochvil, and A. Proskurowski. Systems of distant representatives. *Discrete Applied Mathematics*, 145:306–316, 2005.

[11] M. T. Goodrich, L. J. Guibas, J. Hershberger, and P. J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings 13th Annual Symposium on Computational Geometry*, pages 284–293, 1997.

[12] T. Harris and F. Ross. Fundamentals of a method for evaluating rail net capacities. Technical report, RAND Corporation, Santa Monica, California, U.S., 1955.

[13] W. Harvey. Computing two-dimensional integer hulls. *SIAM Journal on Computing*, 28(6):2285–2299, 1999.

[14] J. Hershberger. Stable snap rounding. *Computational Geometry*, 46(4):403–416, 2013.

[15] R. Klette and A. Rosenfeld. *Digital Geometry: Geometric methods for digital picture analysis.* Elsevier, 2004.

[16] R. Klette and A. Rosenfeld. Digital straightness - a review. *Discrete Applied Mathematics*, 139(1-3):197–230, 2004.

[17] M. Löffler and W. Meulemans. Discretized approaches to schematization. In *Proceedings 29th Canadian Conference on Computational Geometry*, 2017.

[18] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry: Theory and Applications*, 43(4):419–433, 2010.

[19] J. B. Orlin. Max flows in o(nm) time, or better. In *Symposium on Theory of Computing (STOC)*, pages 765–774, 2013.

[20] I. van der Hoog, M. van de Kerkhof, M. van Kreveld, M. Löffler, F. Staals, J. Urhausen, and J. L. Vermeulen. Mapping multiple regions to the grid with bounded hausdorff distance. In *Algorithms and Data Structures Symposium (WADS)*, 2021. To Appear.