Testing Connectedness of Images

Piotr Berman
Unaffiliated Researcher

Meiram Murzabulatov
Computer Science Department, School of Digital Sciences,
Nazarbayev University, Astana, Kazakhstan

Sofya Raskhodnikova
Boston University, MA, USA

Dragos Ristache
Boston University, MA, USA

Abstract

We investigate algorithms for testing whether an image is connected. Given a proximity parameter $\epsilon \in (0,1)$ and query access to a black-and-white image represented by an $n \times n$ matrix of Boolean pixel values, a (1-sided error) connectedness tester accepts if the image is connected and rejects with probability at least 2/3 if the image is ϵ-far from connected. We show that connectedness can be tested nonadaptively with $O(\frac{1}{\epsilon^2})$ queries and adaptively with $O(\frac{1}{\epsilon^{3/2}}\sqrt{\log\frac{1}{\epsilon}})$ queries. The best connectedness tester to date, by Berman, Raskhodnikova, and Yaroslavtsev (STOC 2014) had query complexity $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ and was adaptive. We also prove that every nonadaptive, 1-sided error tester for connectedness must make $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ queries.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property testing, sublinear-algorithms, lower bounds, connectivity, graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.66

Category RANDOM

Funding *Meiram Murzabulatov*: Supported by the Nazarbayev University Social Policy Grant.

1 Introduction

Connectedness is one of the most fundamental properties of images [16]. In the context of property testing, it was first studied two decades ago [18], but the query complexity of this property is still unresolved. We improve the algorithms for testing this property and also give the first lower bound on the query complexity of this task.

We focus on black-and-white images. For simplicity, we only consider square images, but everything in this paper can be easily generalized to rectangular images. We represent an image by an $n \times n$ binary matrix M of pixel values, where 0 denotes white and 1 denotes black. To define *connectedness*, we consider *the image graph* G_M of an image M. The vertices of G_M are $\{(i,j) \mid M[i,j] = 1\}$, and two vertices (i,j) and (i',j') are connected by an edge if $|i-i'| + |j-j'| = 1$. In other words, the image graph consists of black pixels connected by the grid lines. The image is *connected* if its image graph is connected.

We study connectedness in the property testing model [20, 11], first considered in the context of images in [18]. A (1-sided error) property tester for connectedness gets query access to the input matrix M. Given a proximity parameter $\epsilon \in (0,1)$, the tester has to accept if M is connected and reject with probability at least 2/3 if M is ϵ-far from connected. An image is ϵ-far from connected if at least an ϵ fraction of pixels have to be changed to make it connected. The tester is *nonadaptive* if it makes all its queries before receiving any answers; otherwise, it is *adaptive*.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 66; pp. 66:1–66:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In [18], it was shown that connectedness can be tested adaptively with $O(\frac{1}{\epsilon^2}\log^2\frac{1}{\epsilon})$ queries. The adaptive complexity of testing connectedness was later improved to $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ in [8].

1.1 Our Results

1.1.1 Connectedness Testers

We give two new algorithms for testing connectedness of images: one adaptive, one nonadaptive. Both improve on the best connectedness tester to date in terms of query complexity. Previously, no nonadaptive testers for connectedness were proposed.

▶ **Theorem 1.1.** *Given a proximity parameter $\epsilon \in (0,1)$, connectedness of $n \times n$ images, where $n > \epsilon^{-2} \cdot 256$, can be ϵ-tested adaptively and with 1-sided error with query and time complexity $O(\frac{1}{\epsilon^{3/2}}\sqrt{\log\frac{1}{\epsilon}})$.*

It can be tested nonadaptively and with 1-sided error with query and time complexity $O(\frac{1}{\epsilon^2})$.

Previous algorithms for testing connectedness of images are modeled on the connectedness tester for bounded-degree graphs by Goldreich and Ron [12]: they pick a uniformly random pixel and adaptively try to find a small connected component by querying its neighbors. As discussed in [18], even though connectedness of an image is defined in terms of the connectedness of the corresponding (degree-4) image graph, these two properties are different because of how the distance is defined. In the bounded-degree graph model, the (absolute) distance between graphs is the number of edges that need to be changed to transform one graph into the other. In contrast, the (absolute) distance between two image graphs is the number of pixels (vertices) on which they differ; in other words, the edge structure of the image graph is fixed, and only vertices can be added or removed to transform one graph into another. However, previous connectedness testers in the image model did not take advantage of the differences.

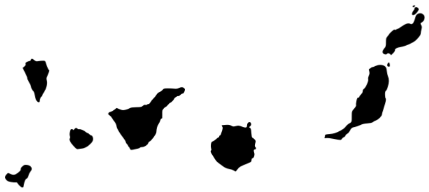

Figure 1 An image M.

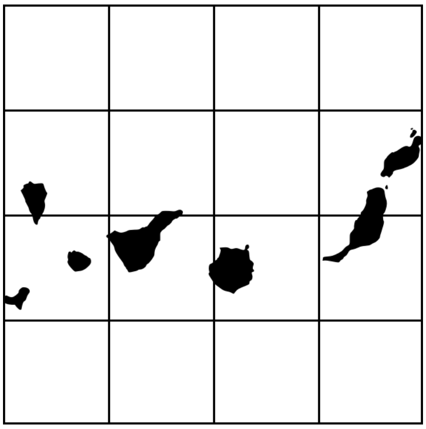

Figure 2 The same image with a grid.

As our starting point, we use an idea from [7] that gave an algorithm for approximating the (relative) distance to the nearest connected image with additive error ϵ with query $O(\frac{1}{\epsilon^4})$ and running time $\exp\left(O\left(\frac{1}{\epsilon}\right)\right)$. They observed that one can modify an image in a small number of pixels by drawing a grid on the image (as shown in Figures 1 and 2). In the resulting image, the distance to connectedness is determined by the properties of individual squares into which the grid lines partition the image.

Our algorithms consider different partitions of the grid (a logarithmic number of partitions in $\frac{1}{\epsilon}$). For each partition, they sample random squares and try to check whether the squares satisfy the following property called *border-connectedness.*

▶ **Definition 1.2** (Border connectedness). *A (sub)image s is* border-connected *if for every black pixel (i,j) of s, the image graph G_s contains a path from (i,j) to a pixel on the border of s. The property* border connectedness, *denoted C', is the set of all border-connected images.*

Our nonadaptive algorithm reads all pixels in each sampled square. Our adaptive algorithm further partitions each square into diamonds, as shown in Figure 4. It queries all pixels on the diamond lattice and then adaptively tries to catch a witness (that is, a small connected component) in one of the diamonds, using the lattice structure. (We could have partitioned into squares again, but partitioning into diamonds makes the proof cleaner and saves a constant factor in the analysis.)

1.1.2 The Lower Bound

We also prove the first nontrivial lower bound for testing connectedness of images. Note that all nontrivial properties have query complexity $\Omega(1/\epsilon)$, even for adaptive testers. In particular, for connectedness, this is the number of queries needed to distinguish between the white image and an image, where we color black a random subset of size ϵn^2 of the set of pixels with both coordinates divisible by 3. By standard arguments, it implies a lower bound of $\Omega(1/\epsilon)$. Some properties of images, such is being a halfplane, can be tested nonadaptively (with 1-sided error) with $O(1/\epsilon)$ queries [5]. We show that it is impossible for connectedness.

▶ **Theorem 1.3.** *Every nonadaptive (1-sided error) ϵ-tester for connectedness of images must query $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ pixels (for some family of images).*

Every 1-sided error tester must catch a witness of disconnectedness in order to reject. This witness could include a connected component completely surrounded by white pixels. The difficulty for proving hardness is that, unlike in the case of finding a witness for disconnectedness of graphs, the algorithm does not have to read the whole connected component. Instead, it is sufficient to find a closed white loop with a black pixel inside it (and another black pixel outside it). As we discussed, it is sufficient for an algorithm to look for witnesses inside relatively small squares (specifically, squares with side length $O(1/\epsilon)$), since adding a grid around such squares, as shown in Figure 2, would change $O(\epsilon n^2)$ pixels. But no matter how you had a witness inside such a square, it can be easily captured with $O(1/\epsilon)$ queries if the border of the square is white.

To overcome this difficulty, we consider a checkerboard-like pattern with white squares replaced by many parallel lines, called *bridges*, with one white (disconnecting) pixel positioned randomly on each bridge. See Figure 5. To catch a white border around a connected component, a tester has to query all disconnecting pixels of at least one square. To make this difficult, we hide the checkerboard pattern inside a randomly positioned *interesting window.* The sizes of interesting windows and their positions are selected so that the tester cannot effectively reuse queries needed to succeed in catching the disconnecting pixels in each interesting window.

1.2 Other Related Work

In addition to [12], connectedness testing and approximating the number of connected components in graphs in sublinear time was explored in [9, 8, 4]. Other property testing tasks studied in the pixel model of images, the model considered in this paper, include testing whether an image is a half-plane [18], convexity [18, 6, 5], and image partitioning properties [13]. Early implementations and applications to vision were provided in [13, 14, 15, 17]. Finally, general classes of matrix properties were investigated, including matrix-poset properties [10], *earthmover resilient* properties [2], *hereditary* properties [1], and classes of matrices that are free of specified patterns [3].

Testing connectedness has also been studied by Ron and Tsur [19] with a different input representation suitable for testing sparse images.

2 Definitions and Notation

We use $[0..n)$ to denote the set of integers $\{0, 1, \dots, n-1\}$ and $[n]$ to denote $\{1, 2, \dots, n\}$.

2.1 Image Representation

We represent an image by an $n \times n$ binary matrix M of pixel values, where 0 denotes white and 1 denotes black. The object is a subset of $[0..n)^2$ corresponding to black pixels; namely, $\{(i, j) \mid M[i, j] = 1\}$. The *border of the image* is the set $\{(i, j) \in [0..n)^2 \mid i \in \{0, n-1\}$ or $j \in \{0, n-1\}\}$.

2.2 Property Testing Definitions

A *property* $\mathcal{P}$ is a set of images. The set of connected images is denoted C. The *absolute distance* from an image M to a property $\mathcal{P}$, denoted $Dist(M, \mathcal{P})$, is the smallest number of pixels in M that need to be modified to get an image in $\mathcal{P}$. The (relative) distance between an $n \times n$ image M and a property $\mathcal{P}$ is $dist(M, \mathcal{P}) = Dist(M, \mathcal{P})/n^2$. We say that M is ϵ-far from $\mathcal{P}$ if $dist(M, \mathcal{P}) \geq \epsilon$; otherwise, M is ϵ-close to $\mathcal{P}$.

3 Adaptive and Nonadaptive Property Testers for Connectedness

In this section, we present our testers for connectedness, proving Theorem 1.1. Both testers use the same top-level procedure, described in Algorithm 1. First, it samples random pixels to ensure that a black pixel is found. It will be used later to certify non-connectedness by producing a black pixel and an isolated black component. Then Algorithm 1 considers a logarithmic number of partitions of the image into subimages of the same size. For each partition, it samples a carefully selected number of these subimages and tests them for border connectedness (see Definition 1.2). This is where the two algorithms diverge. The nonadaptive algorithm tests for border connectedness using the subroutine *Exhaustive-Square-Tester* which queries all pixels in the sampled square and determines exactly if the square is border connected. The adaptive algorithm uses subroutine *Diagonal-Square-Tester* (Algorithm 2). If the top-level procedure finds a subimage that violates border connectedness and a black pixel outside that subimage, it rejects; otherwise, it accepts.

To simplify the analysis of the algorithm, we assume[1] that $n-1$ and $1/\epsilon$ are powers of 2. Next, we define terminology used to describe the partitions considered by Algorithm 1.

[1] This assumption can be made w.l.o.g. because if $n \in (2^{i-1}+1, 2^i+1)$ for some i , instead of the original image M we can consider a $(2^i+1) \times (2^i+1)$ image M', which is equal to M on the corresponding coordinates and has white pixels everywhere else. Let $\epsilon' = \epsilon n^2/(2^i+1)^2$. To ϵ-test M for connectedness, it suffices to ϵ'-test M' for connectedness. The resulting tester for M has the desired query complexity because $\epsilon' = \Theta(\epsilon)$. If $\epsilon \in (1/2^j, 1/2^{j-1})$ for some j, to ϵ-test a property $\mathcal{P}$, it suffices to run an ϵ''-test for $\mathcal{P}$ with $\epsilon'' = 1/2^j < \epsilon$.

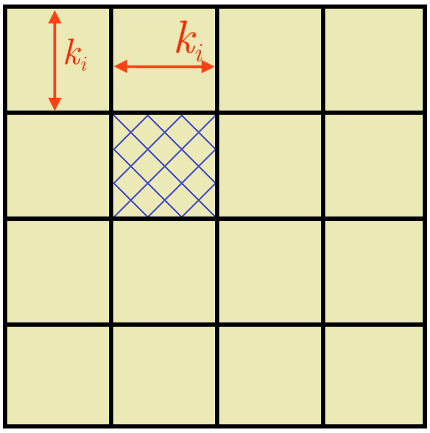

Figure 3 An illustration to Definition 3.1: black lines consist of grid pixels; the 16 yellow $k_i \times k_i$ squares represent squares of S_i. One of the squares includes diagonal lattice pixels from Definition 3.6 that are used in Algorithm 2.

▶ **Definition 3.1** (Grid pixels, squares of different levels, witnesses). *For $i \in [0..\log \frac{1}{\epsilon})$, let $k_i = \frac{4}{\epsilon} \cdot 2^{-i} - 1$. Call (x, y) a* grid pixel of level i *if $(k_i + 1)|x$ or $(k_i + 1)|y$. For all coordinates u, v, which are divisible by $k_i + 1$, the $k_i \times k_i$ subimage that consists of pixels $[k_i]^2 + (u, v)$ is called a* square of level i. *The set of all squares of level i is denoted $\mathcal{S}_i$.* Boundary pixels *of a square of level i are the pixels of the square which are adjacent to the grid pixels of level i. A square of any level that violates property C' (see Definition 1.2) is called a* witness.

Algorithm 1 *ϵ-tester* for connectedness.

```
input  : parameter $\epsilon \in (0, 1)$ ; access to a $n \times n$ binary matrix $M$.
1  Query $\frac{4}{\epsilon}$ pixels uniformly at random with replacement.
2  For $i = 0$ to $\log \frac{1}{\epsilon}$
   (a) Sample $2^{i+3}$ uniformly random squares of level $i$ (see Definition 3.1) with replacement.
   (b) For every sampled square $s$ from Step 2a, let $[k_i]^2 + (u, v)$ be the set of its pixels.
       Run the border-connectedness subroutine with inputs $i, u, v$: if the tester is nonadaptive,
       use *Exhaustive-Square-Tester*; otherwise use *Diagonal-Square-Tester* (Algorithm 2).
       If the subroutine rejects and Step 1 detected a black pixel outside $s$, **reject**.
3  **Accept**.
```

3.1 Effective Local Cost and the Structural Lemma

In this section, we prove our main structural lemma (Lemma 3.5) used in the analysis of Algorithm 1. It relates the distance to connectedness to the properties of individual squares, defined next.

▶ **Definition 3.2** (Local cost and effective local cost). *For a level i, consider a square $s \in \mathcal{S}_i$. The* local cost *of s is $lc(s) = Dist(s, C')$. The* effective local cost *of s is $elc(s) = \min(2k_i, lc(s))$.*

Next we state and prove two claims used in the proof of Lemma 3.5.

▷ Claim 3.3. For any square s of level $i \in [0..\log \frac{1}{\epsilon} - 1)$, let $ch(s)$ denote the set of its 4 children (i.e., squares of level $i+1$ inside it). Then $lc(s) \le elc(s) + \sum_{q \in ch(s)} lc(q)$.

Proof. If $lc(s) \le 2k_i$ then $elc(s) = lc(s)$. Since all costs are nonnegative, the inequality in Claim 3.3 becomes trivial.

Now assume that $lc(s) > 2k_i$. Then $elc(s) = 2k_i$. We can modify $\sum_{q \in ch(s)} lc(q)$ pixels in s so that all its children satisfy the property C'. (Note that here C' is the set of $k_i \times k_i$ (sub)images.) Then we can make black all pixels of s that partition it into its children, i.e., pixels $\{(x, y) \mid x = \frac{k_i+1}{2}$ or $y = \frac{k_i+1}{2}\}$. There are at most $2k_i$ such pixels, and after this modification s will satisfy C'. Hence, $lc(s) \le elc(s) + \sum_{q \in ch(s)} lc(q)$. ◁

▷ Claim 3.4 (Distance to Border Connectedness). Let s be a $k \times k$ image. Then

$$Dist(s, C') \le \frac{k^2}{4}.$$

Proof. If s contains at most $\frac{k^2}{4}$ black pixels, we can make all of them white, i.e., modify at most $\frac{k^2}{4}$ pixels and obtain an image that satisfies C'. Now consider an image s with more than $\frac{k^2}{4}$ black pixels, i.e., with less than $\frac{3k^2}{4}$ white pixels. Partition all pixels of s into 3 groups such that group $i \in \{0, 1, 2\}$ contains all pixels (x, y), where $y \equiv i \pmod 3$. Making all pixels of one group black produces an image that satisfies C'. By averaging, at least one group has less than $\frac{k^2}{4}$ white pixels. Making all these white pixels black results in an image that satisfies C'. This completes the proof. ◁

▶ **Lemma 3.5** (Structural Lemma). *Let M be an $n \times n$ image that is ϵ-far from C. Then the sum of effective local costs of all squares of all levels inside M is at least $\frac{\epsilon n^2}{2}$.*

Proof. To obtain a connected image, we can make all the $\frac{\epsilon n^2}{2}$ grid pixels of level 0 black and modify pixels inside every square of $\mathcal{S}_0$ to ensure it satisfies the property C'. Thus,

$$\sum_{s \in \mathcal{S}_0} lc(s) \ge Dist(M, C) - \frac{\epsilon n^2}{2} \ge \frac{\epsilon n^2}{2}.$$

Consequently, it suffices to show that $\sum_{i=0}^{\log \frac{1}{\epsilon} - 1} \sum_{s \in \mathcal{S}_i} elc(s) \ge \sum_{s \in \mathcal{S}_0} lc(s)$.

Let s be a square of level i. We use $desc(s, j)$ to denote the set of all squares of level $j \ge i$ inside s. (In particular, $desc(s, i)$ contains only s.) We will prove by induction that for any integer $j \in [i, \log \frac{1}{\epsilon} - 1)$,

$$lc(s) \le \sum_{h=i}^{j} \sum_{q \in desc(s,h)} elc(q) + \sum_{q \in desc(s,j+1)} lc(q). \tag{1}$$

For $j = i$ (base case), the inequality in (1) holds since it is equivalent to the statement in Claim 3.3. Assume that (1) holds for $j = m$, that is,

$$lc(s) \le \sum_{h=i}^{m} \sum_{q \in desc(s,h)} elc(q) + \sum_{q \in desc(s,m+1)} lc(q).$$

We will prove (1) holds for $j = m + 1$. By Claim 3.3,

$$lc(q) \le elc(q) + \sum_{f \in ch(q)} lc(f).$$

Thus,

$$\sum\nolimits_{q\in desc(s,m+1)} lc(q) \le \sum\nolimits_{q\in desc(s,m+1)} elc(q) + \sum\nolimits_{q\in desc(s,m+2)} lc(q)$$

and

$$\begin{aligned} lc(s) &\le \sum_{h=i}^{m} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,m+1)} lc(q) \\ &\le \sum_{h=i}^{m+1} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,m+2)} lc(q), \end{aligned}$$

completing the inductive argument.

By (1) applied with $j = \log\frac{1}{\epsilon} - 2$, we get that for every square s of level i,

$$\begin{aligned} lc(s) &\le \sum_{h=i}^{\log\frac{1}{\epsilon}-2} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,\log\frac{1}{\epsilon}-1)} lc(q) \\ &= \sum_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{q\in desc(s,h)} elc(q), \end{aligned} \tag{2}$$

where the final equality holds because in every square of level $i = \log\frac{1}{\epsilon} - 1$, we have $k_i = 7$, and consequently, by Claim 3.4, the local cost is at most $\frac{7^2}{4} < 2\cdot 7$, i.e., it is equal to the effective local cost of that square.

Summing up (2) for all squares in $\mathcal{S}_0$, we get

$$\sum\nolimits_{s\in\mathcal{S}_0} lc(s) \le \sum\nolimits_{s\in\mathcal{S}_0} \sum\nolimits_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{q\in desc(s,h)} elc(q) = \sum\nolimits_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{s\in\mathcal{S}_i} elc(s),$$

where the last equality is obtained by switching the order of summations and rearranging the second summation in terms of levels. ◀

3.2 Testing Border-Connectedness

In this section, we state and analyze our adaptive border connectedness subroutine after defining the concepts used in it. We state the guarantees of both border connectedness subroutines in Lemma 3.7.

For the adaptive subroutine, we partition the square into diamonds and fences surrounding them, as described in Definition 3.6. The subroutine queries all pixels on the fences and categorizes diamonds into those whose black pixels are potentially connected to the border (set B in Algorithm 2) and those whose black pixels are definitely not (set A in Algorithm 2). Then it tries to find a black pixel in a diamond from set A or (using BFS) an isolated black component in diamonds from set B. Observe that either of the two provides evidence that the square is not border-connected.

▶ **Definition 3.6** (Diagonal lattice pixels, diamonds and fences). *For a fixed value of i, consider a square s in $\mathcal{S}_i$. Let m_i be the largest odd integer less than or equal to $\lceil\sqrt{k_i/\log k_i}\rceil$.* Diagonal lattice pixels *of the square is the set of pixels $L = \{(x,y)\in s \mid m_i|(x+y)$ or $m_i|(x-y)\}$. Let D be a $k_i\times k_i$ image whose pixels with coordinates from $[k_i]^2 - L$ are black and the remaining pixels are white. A set of pixels of the square whose corresponding pixels in D form a connected component is called a* diamond *of the square. A set of all diagonal lattice pixels that have some neighbouring pixel(s) from a particular diamond is called the* fence *of that diamond.*

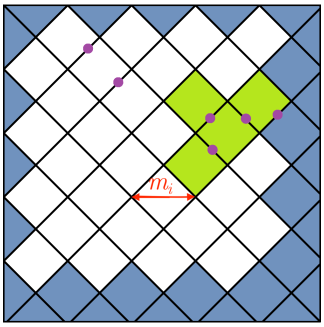

Figure 4 An example of execution of Algorithm 2. Black lines represent lattice pixels. The blue diamonds are included in B because they are adjacent to the border of the square. The purple dots represent black pixels in the lattice. The green diamonds are added to B during the BFS because their fences contain black pixels. The white diamonds remain in A.

Note that lattice pixels are not part of any diamond. Moreover, some diamonds are partial (those that have pixels from the border of the square.)

Algorithm 2 Border-connectedness subroutine *Diagonal-Square-Tester*.

```
input  : parameters $i$, $u$ and $v$; access to an $n \times n$ matrix $M$.
Let $s$ be a square of level $i$ that consists of pixels $[k_i]^2 + (u, v)$ and $m_i$ be the largest odd
  integer less than or equal to $\lceil \sqrt{k_i / \log k_i} \rceil$.
Query all the diagonal lattice pixels of $s$ (see Definition 3.6).
Initialize $B$ to be the set of all diamonds of $s$ that contain a border pixel of $s$. Initialize
  $A$ to be the set of the remaining diamonds of $s$.
While $\exists d_1 \in B$ and $\exists d_2 \in A$ such that $d_1$ and $d_2$ have a black pixel in the common
  portion of their fences, move $d_2$ from $A$ to $B$.
 Query $k_i m_i$ pixels in $s$ uniformly at random with replacement.
  (a) If a black pixel from a diamond in $A$ or its fence is discovered, reject.
  (b) If a black pixel from a diamond in $B$ is discovered, pick a natural number $x \in [k_i^2]$ from
      the distribution with the probability mass function $f(j) = \frac{1}{j(j+1)}$ for all $j \in [k_i^2 - 1]$
      and $f(j) = \frac{1}{j}$ for $j = k_i^2$. (Observe that $\Pr(x \geq j) = \frac{1}{j}$ for all $j \in [k_i^2]$.) Starting from
      the black pixel, perform a BFS of its connected component. If the search halted after
      discovering at most $x$ black pixels, none of which are on the border of $s$, reject. Else (if
      $x + 1$ black pixels were found for this component or a black pixel on the border of
      square $s$ is reached) stop the search and proceed with the remaining queried pixels.
Accept.
```

Recall from Definition 3.1 that a witness is a square of one of the levels that violates border-connectedness.

▶ **Lemma 3.7.** *Fix level $i \in [0..\log \frac{1}{\epsilon})$. Let $s \in \mathcal{S}_i$ be a witness that consists of pixels $[k_i]^2 + (u, v)$. A border-connectedness subroutine of Algorithm 1 rejects s with probability at least $\frac{elc(s) \cdot \alpha}{2k_i}$, where $\alpha = 1$ for* Exhaustive-Square-Tester *and $\alpha = 1 - e^{-1}$ for* Diagonal-Square-Tester.

Proof. *Exhaustive-Square-Tester* determines that s is a witness with probability $1 \geq \frac{elc(s)}{2k_i} \cdot 1$.

Now we prove the statement for *Diagonal-Square-Tester*. Let A and B be defined as in Algorithm 2 after Step 3. Let a be the number of black pixels in all the diamonds of the set A and on the fences of those diamonds. Let b be the number of connected components of the image graph that are formed by black pixels in all the diamonds in the set B and in their fences and that contain no pixels on the border of s.

Next, we prove that

$$bm_i + a \geq lc(s) \geq elc(s). \tag{3}$$

We claim that we can connect all b connected components to each other and to the border of the square by modifying at most bm_i pixels. To see this, notice that any two black pixels in the same diamond can be connected to each other by changing less than m_i pixels (by taking any Manhattan-distance shortest path between the two pixels and making it all black). To prove the claim, we can connect the b connected components to the border of the square in the order their diamonds were added to B by the algorithm. The initial diamonds placed in B have at least one pixel on the border of the square, so their connected components can be directly connected to the border (using at most m_i pixels per connected component). Now assume that we already connected to the border all components that have pixels in the diamonds added to B so far. When the algorithm moves some diamond d_2 from A to B, it is done because there is already a diamond d_1 in B such that d_1 and d_2 have a black pixel β in the common portion of their fences. Then β must be already connected to the border. If there are any connected components in d_2 that are not connected to the border yet, we can fix that by connecting them to β (using at most m_i pixels per connected component). We proceed like this until all b connected components of the diamonds that were added to B by the algorithm are connected to the border of the square. At this point, we have changed at most bm_i pixels.

Recall that we have a black pixels in all the diamonds of the set A and on the fences of those diamonds. We make all of them white. After these at most $bm_i + a$ modifications, the square s satisfies C'. Thus, Equation (3) holds.

Observe that for $x \in [0, 1]$,

$$x \geq 1 - e^{-x} \geq x(1 - e^{-1}) > x/2. \tag{4}$$

By (4), the probability that a specific pixel from s is selected by Algorithm 2 in Item b is

$$1 - (1 - 1/k_i^2)^{k_i m_i} \geq 1 - e^{-\frac{k_i m_i}{k_i^2}} > \frac{m_i}{2k_i}.$$

Consider one of the b connected components defined above. Let p be the number of pixels in it. *Diagonal-Square-Tester* finds this component completely if in Step 4b of the subroutine, one of the p pixels from the component is selected and $x \geq p$ is chosen. This happens with probability at least $\frac{1}{p} \cdot \frac{m_i}{2k_i} \cdot p = \frac{m_i}{2k_i}$. The subroutine determines that s is a witness if it finds one of the b connected components or one of the a black pixels considered above. Thus, by (3) and (4), the probability that Algorithm 2 determines that square s is a witness is at least

$$1 - \left(1 - \frac{m_i}{2k_i}\right)^{a+b} \geq 1 - e^{-\frac{m_i(a+b)}{2k_i}} \geq 1 - e^{-\frac{bm_i + a}{2k_i}} \geq 1 - e^{-\frac{elc(s)}{2k_i}} \geq \frac{elc(s)(1 - e^{-1})}{2k_i},$$

completing the proof of Lemma 3.7. ◀

3.3 Proof of Theorem 1.1

In this section, we use Lemmas 3.5 and 3.7 to complete the analysis of our connectedness tester and the proof of Theorem 1.1.

Algorithm 1 always accepts connected images, since it sees no violation of C' in Step 2b. Consider an image M that is ϵ-far from connectedness. For every $i \in [0..\log \frac{1}{\epsilon})$, there are $\frac{n^2}{k_i^2}$ squares in $\mathcal{S}_i$. Let $\mathcal{W}_i$ be the set of all witnesses in $\mathcal{S}_i$. Let s be a square in $\mathcal{W}_i$. The probability that Algorithm 1 chooses s in Step 2a is

$$1 - \left(1 - \frac{k_i^2}{n^2}\right)^{2^{i+3}} \geq 1 - e^{\frac{k_i^2 \cdot 2^{i+3}}{n^2}} > \frac{k_i^2 \cdot 2^{i+2}}{n^2},$$

where the inequalities follow from (4). By Lemma 3.7, the probability that the algorithm rejects s after choosing it is at least $\alpha \cdot elc(s)/2k_i$. Thus, the probability that the algorithm samples s in Step 2a and then rejects in Step 2b is at least

$$\frac{2^{i+2}k_i \cdot \alpha \cdot elc(s)}{2n^2} \geq \frac{6\alpha \cdot elc(s)}{\epsilon n^2}.$$

Thus, the probability that Algorithm 1 does not catch $s \in \mathcal{W}_i$ as a witness is at most $1 - \frac{6\alpha \cdot elc(s)}{\epsilon n^2} \leq e^{-\frac{6\alpha \cdot elc(s)}{\epsilon n^2}}$. The probability that the algorithm does not catch any witness of $\mathcal{W}_i$ is at most

$$P_i = e^{-\frac{6\alpha}{\epsilon n^2} \cdot \sum_{s \in \mathcal{W}_i} elc(s)}.$$

Observe that $elc(s) = 0$, for $s \in \mathcal{S}_i - \mathcal{W}_i$. Thus, by Lemma 3.5,

$$\sum_{s \in \mathcal{W}_i} elc(s) = \sum_{s \in \mathcal{S}_i} elc(s) \geq \frac{\epsilon n^2}{2}.$$

By a union bound over all levels, the probability that Algorithm 1 fails to catch any witness is at most

$$\sum_{i \in \{0\} \cup [\log(1/\epsilon)]} P_i \leq e^{-\frac{6\alpha}{\epsilon n^2} \cdot \frac{\epsilon n^2}{2}} = e^{-3\alpha}.$$

Therefore, the probability that the algorithm detects at least one witness is at least $1 - e^{-3\alpha}$. Since at least an ϵ fraction of pixels in M are black and every square of every level contains at most $\frac{16}{\epsilon^2} < \frac{\epsilon n^2}{2}$ pixels, the probability that Algorithm 1 detects a black pixel outside of that witness in Step 1 is at least $1 - (1 - \frac{\epsilon}{2})^{\frac{4}{\epsilon}} > 1 - e^{-2}$. Thus, for both values of α, the probability that Algorithm 1 rejects M is at least

$$(1 - e^{-2})(1 - e^{-3\alpha}) \geq 2/3.$$

3.3.1 Query Complexity

We prove that Algorithm 1 has query complexity $O(\frac{1}{\epsilon^2})$ if it uses *Exhaustive-Square-Tester* as a subroutine. Algorithm 1 samples 2^{i+1} squares of level $i \in \{0\} \cup [\log \frac{1}{\epsilon}]$ and, for each sampled square, it calls *Exhaustive-Square-Tester* which makes $(\frac{4}{\epsilon} \cdot 2^{-i} - 1)(\frac{4}{\epsilon} \cdot 2^{-i} - 1) < \frac{16}{\epsilon^2 2^{2i}}$ queries in each sampled square of level i. Thus, the query complexity of Algorithm 1 is

$$\sum_{i=0}^{\log \frac{1}{\epsilon}} 2^{i+1} \cdot \frac{16}{\epsilon^2 2^{2i}} < \sum_{i=0}^{\log \frac{1}{\epsilon}} \frac{32}{\epsilon^2 2^i} = O\left(\frac{1}{\epsilon^2}\right).$$

When Algorithm 1 uses *Diagonal-Square-Tester*, it queries at most $\frac{2k_i^2}{m_i}$ diagonal lattice pixels inside each square of level i (in Step 1 of the subroutine). After that, in Step 4 of the subroutine, it selects $k_i m_i$ pixels and a number $x \in [k_i^2]$ from the specified distribution and then makes at most $4x$ queries for each selected pixel. Observe that $\mathbb{E}(x) = O(\log k_i)$. Thus, the expected number of queries inside a square of level i is at most $\frac{2k_i^2}{m_i} + k_i m_i \cdot 4 \cdot O(\log k_i) = O(k_i^{3/2}\sqrt{\log k_i})$. The expected total number of queries is $\sum_{i=0}^{\log(1/\epsilon)} O(k_i^{3/2}\sqrt{\log k_i}) \cdot 2^{i+1} = O(\epsilon^{-3/2}\sqrt{\log \frac{1}{\epsilon}})$.

By standard arguments, the adaptive version of Algorithm 1 can be converted to an algorithm that makes asymptotically the same number of queries in the worst case, and has the same accuracy guarantee and running time.

3.3.2 Running Time

The time complexity of Step 1a and Step 2 of Algorithm 1 is $O(\frac{1}{\epsilon})$. Therefore, the total time complexity of Algorithm 1 is $O(\frac{1}{\epsilon})$+time complexity of Step 1b. In Step 1b, Algorithm 1 uses either *Exhaustive-Square-Tester* or *Diagonal-Square-Tester*. Both of them perform a breadth first search within each sampled square. Breadth first search is linear in the sum of the number of edges and the number of nodes of the graph. Every pixel of a sampled square has at most 4 neighboring pixels. Thus, the number of edges in the image graph of every sampled square is linear in the number of pixels inside it and the time complexity of Step 1b is linear in the number of all queried pixels, i.e., $O(\frac{1}{\epsilon^2})$ for *Exhaustive-Square-Tester* and $O(\epsilon^{-3/2}\sqrt{\log(1/\epsilon)})$ for *Diagonal-Square-Tester*. This completes the proof of the theorem.

4 Lower Bound for Testing Connectedness

In this section, we give a lower bound on the query complexity of testing connectedness, proving Theorem 1.3. We use the standard set up of constructing a distribution $\mathcal{N}$ on ϵ-far inputs such that every deterministic nonadaptive algorithm that makes $q \leq \frac{c}{\epsilon}\log\frac{1}{\epsilon}$ queries (for some constant c) has probability of error greater than 1/3. By Yao's Principle [21], it is sufficient to prove Theorem 1.3.

4.1 Construction of $\mathcal{N}$

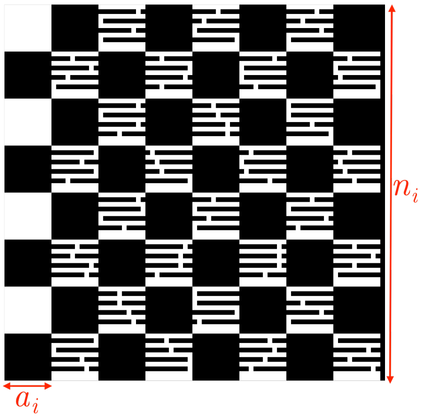

Figure 5 Our construction of $\mathcal{N}$: an interesting window together with a column of black pixels immediately to the right of it. All other pixels in the constructed image are white.

The construction is parameterized by n and the proximity parameter ϵ. It gives a distribution supported on $(n+1) \times (n+1)$ images that are ϵ-far from connectedness. We assume that ϵ is sufficiently small and that $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$. We also assume that n is a power of 2 and $1/\epsilon$ is an even power of 2 and that both of them are sufficiently large, so that all indices in our construction are integer. (See also Footnote 1 for the discussion of integrality issues.) Our construction starts by selecting a *level.* The indices of the levels range from the low index $\ell = \frac{1}{8}\log\frac{1}{\epsilon}$ to the high index $h = \frac{1}{4}\log\frac{1}{\epsilon}$. For all $i \in \{\ell, \ell+1, \ldots, h\}$, define $a_i = 2^i$ and $n_i = 16 \cdot \sqrt{\epsilon} \cdot n \cdot a_i = \sqrt{\epsilon} \cdot n \cdot 2^{i+4}$. First, we pick a uniformly random integer $i \in \{\ell, \ell+1, \ldots, h\}$. Consider the $n \times n$ image resulting from removing the last row and the last column of the $(n+1) \times (n+1)$ image. We partition this $n \times n$ image into $(\frac{n}{n_i})^2$ squares with side length n_i called *windows of level i*; that is, each window of level i is an $n_i \times n_i$ subimage. We pick one of the windows of level i uniformly at random and call it an *interesting window.* We make the n_i pixels immediately to the right of the interesting window black, representing a vertical black line segment, and all other pixels outside of the interesting window white.

Now we describe how to color the interesting window. See Figure 5 for an illustration. We fill the interesting window with a checkerboard pattern of squares of size $a_i \times a_i$ pixels. Inside each white checkerboard square that is not in the first column, number the rows starting from 0 and make every odd row, excluding the last, fully black, except for one randomly selected pixel for each row. The resulting $\frac{a_i}{2} - 1$ black lines, each containing one white pixel, are called *bridges.* The white pixels on the bridges are called *disconnecting pixels.* We refer to each checkerboard square with the bridges as a *bridge square.*

The intuition behind the construction is the following. Each black checkerboard square is in its own connected component. However, to "catch" this connected component as a witness of disconnectedness, a tester would have to query all the disconnecting pixels in the bridge squares to the left and/or to the right of the black square. Since the positions of the disconnecting pixels are random, it would have to query $\Omega(a_i^2)$ pixels in at least one relevant bridge square. Since the windows are selected at random, the algorithm would have to do it for many windows of each level. The key feature of our construction is that interesting windows of different levels are either disjoint or contained in one another, so potential witnesses for one window can't significantly help with another.

4.2 Analysis of the Construction

We start by showing that all images in the support of $\mathcal{N}$ are far from connected.

▶ **Lemma 4.1.** *Every image in the support of $\mathcal{N}$ is ϵ-far from connected.*

Proof. There are $\frac{(n_i/a_i)^2}{2} = 128\epsilon n^2$ black regions, each one in a separate connected component except for the last $\frac{n_i/a_i}{2} = 8\sqrt{\epsilon} \cdot n$ which are connected by the vertical line. Thus, the image graph has at least $128 \cdot \epsilon n^2 - 8\sqrt{\epsilon} \cdot n$ connected components. Changing one pixel from white to black corresponds to adding a node of degree at most 4 to the image graph. This decreases the number of connected components by at most 3. Removing a pixel decreases the number of connected components by at most 1. Consequently, overall, we need to change at least $\frac{1}{3}(128\epsilon n^2 - 8\sqrt{\epsilon} \cdot n)$ pixels. This is at least ϵn^2 for sufficiently large n; in particular, it holds for $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$, which was our assumption in the beginning of Section 4.1. ◀

Next we show that if the number of queried pixels is small, then every 1-sided error deterministic algorithm detects a violation of connectedness in an image distributed according to $\mathcal{N}$ with insufficiently small probability.

▶ **Lemma 4.2.** *Let M be an image distributed according to $\mathcal{N}$. Fix a deterministic nonadaptive 1-sided error algorithm $\mathcal{A}$ for testing connectedness of images. Let $\mathbf{Q}$ be the set of pixels queried by $\mathcal{A}$ and let $q = |\mathbf{Q}|$. For sufficiently small constant $c > 0$, if $q \leq \frac{c}{\epsilon}\log\frac{1}{\epsilon}$, then $\mathcal{A}$ detects a violation of connectedness in M with probability less than $1/3$.*

Proof. We define an event E that must happen in order for the algorithm $\mathcal{A}$ to succeed in finding a witness of disconnectedness in an image distributed according to $\mathcal{N}$. Then we show that the probability of E is too small. To define E, we define a special group of pixels.

▶ **Definition 4.3** (A revealing set and event E)**.** *Let M be an image distributed according to $\mathcal{N}$. The set of all disconnecting pixels from one bridge square of M is called a* revealing set *for the window containing the bridge square. Let E denote the event that $\mathbf{Q}$ contains a revealing set for the interesting window of M.*

Since the tester $\mathcal{A}$ has 1-sided error, it can reject only if it finds a violation of connectedness. In particular, for an image distributed according to $\mathcal{N}$, in order to succeed, it must find a revealing set for the interesting window of the image.

▷ Claim 4.4. If the number of queries $q \leq \frac{c}{\epsilon}\log\frac{1}{\epsilon}$ for sufficiently small constant c, then $\Pr[E] < 1/3$.

Proof. An important feature of our construction is that the largest bridge square is smaller than the smallest window. Indeed, the side length of the largest bridge square is $a_h = (\frac{1}{\epsilon})^{1/4}$, whereas the side length of the smallest window is $n_\ell = 16\sqrt{\epsilon}\cdot(\frac{1}{\epsilon})^{1/8}n = 16\epsilon^{3/8}n$. Thus, $a_h < n_\ell$ as long as $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$, as we assumed in the beginning of Section 4.1. The consequence of this feature and the fact that both a_i's and n_i's are powers of 2 is that each bridge square of level i is contained in one window of level j for all levels i and j.

A (potential) bridge square of level $i \in \{\ell, \ell+1, \ldots, h\}$ is *covered* if $\mathbf{Q}$ contains at least $a_i^2/8$ pixels from that square. A window of level i is *good* if it contains a covered bridge square of level $j \geq i$; otherwise, it is *bad*. For each good window w, we pick a covered bridge square of the highest level contained in w and call it the covered bridge square *associated with* w. All windows of the same level are associated with different covered bridge squares, because each bridge square is contained in exactly one window of a given level.

Let G be the event that the interesting window in M is good. Then, by the law of total probability,

$$\Pr[E] = \Pr[E \mid G]\cdot\Pr[G] + \Pr[E \mid \overline{G}]\cdot\Pr[\overline{G}] \leq \Pr[G] + \Pr[E \mid \overline{G}].$$

Next, we analyze event G. For every level $i \in \{\ell, \ell+1, \ldots, h\}$, let g_i be the number of good windows of level i associated with covered bridge squares of level i and let t_i be the total number of good windows of level i. Then $g_h = t_h$ and $g_i = t_i - t_{i+1}$ for all $i \in [\ell, h-1]$. Observe that, for each level $i \in [\ell, h]$, the covered bridge squares associated with the good windows of level i are distinct. By definition, each of them contributes at least $a_i^2/8$ towards $\mathbf{Q}$. Therefore, the number of all pixels in $\mathbf{Q}$ satisfies

$$\begin{aligned} q &\geq \sum_{i=\ell}^{h}\frac{a_i^2 g_i}{8} = \frac{1}{8}\left(\sum_{i=\ell}^{h-1} a_i^2(t_i - t_{i+1}) + a_h^2 t_h\right) = \frac{1}{8}\left(\sum_{i=\ell}^{h-1} 4^i(t_i - t_{i+1}) + 4^h t_h\right) \\ &\geq \frac{3}{32}\sum_{i=\ell}^{h}(4^i t_i). \end{aligned} \tag{5}$$

Recall that $q \leq \frac{c}{\epsilon} \log \frac{1}{\epsilon}$, that the total number of levels is $h - \ell + 1 = \Theta(\log \frac{1}{\epsilon})$, and that the number of all windows for each level i is $(\frac{n}{n_i})^2 = \Theta(\frac{1}{4^i \epsilon})$. Thus,

$$\begin{aligned}\Pr[G] &= \frac{1}{h - \ell + 1} \sum_{i=\ell}^{h} \frac{t_i}{(n/n_i)^2} = \frac{1}{\Theta(\log(\frac{1}{\epsilon}))} \sum_{i=\ell}^{h} (t_i \cdot \Theta(4^i \epsilon)) = \frac{1}{\Theta(\frac{1}{\epsilon} \log(\frac{1}{\epsilon}))} \sum_{i=\ell}^{h} (4^i t_i) \\ &< \frac{q}{\Theta(\frac{1}{\epsilon} \log(\frac{1}{\epsilon}))} < 1/6,\end{aligned}$$

where Equation (5) was used to obtain the first inequality, and the last inequality holds for sufficiently small constant c.

It remains to analyze $\Pr[E \mid \overline{G}]$, that is, the probability of E, given that the interesting window is bad. Consider a bad window of level $i \in \{\ell, \dots, h\}$. It has at most q bridge squares that contain a queried pixel. Consider one of such bridge squares. Recall that this bridge square has $a_i/2 - 1$ bridges. Number these bridges using integers $1, 2, \dots, a_i/2 - 1$. Let x_k denote the number of pixels of $\mathbf{Q}$ on the bridge number $k \in [a_i/2 - 1]$ of the bridge square. Then the probability that this bridge square has a revealing set for the window is

$$\begin{aligned}\prod_{k=1}^{a_i/2-1} \frac{x_k}{a_i} &\leq \left(\frac{1}{a_i/2 - 1} \cdot \sum_{k=1}^{a_i/2-1} \frac{x_k}{a_i} \right)^{a_i/2-1} \leq \left(\frac{a_i/8}{a_i/2 - 1} \right)^{a_i/2-1} \leq \left(\frac{1}{2} \right)^{a_i/2-1} \\ &\leq 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}},\end{aligned}$$

where the first inequality follows from the inequality between geometric and arithmetic means, the second inequality holds since $\sum_{k=1}^{a_i/2} x_k < a_i^2/8$, and the third inequality holds since ϵ is sufficiently small and, consequently, we can assume that the minimum value of a_i is at least 4. By a union bound over all bridge squares in this window that contain a query,

$$\Pr[E \mid \overline{G}] \leq 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}} \cdot q \leq 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}} \cdot \frac{c}{\epsilon} \cdot \log \frac{1}{\epsilon} < \frac{1}{6},$$

for sufficiently small ϵ and constant c.

Thus, $\Pr[E] \leq \Pr[G] + \Pr[E \mid \overline{G}] < \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$, as claimed. This completes the proof of Claim 4.4. ◁

This concludes the proof of Lemma 4.2. ◀

Theorem 1.3 follows from Lemma 4.2.

References

1 Noga Alon, Omri Ben-Eliezer, and Eldar Fischer. Testing hereditary properties of ordered graphs and matrices. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 848–858. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.83.

2 Omri Ben-Eliezer and Eldar Fischer. Earthmover resilience and testing in ordered structures. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 18:1–18:35. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.18.

3 Omri Ben-Eliezer, Simon Korman, and Daniel Reichman. Deleting and testing forbidden patterns in multi-dimensional arrays. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.9.

4 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Inf. Process. Lett.*, 114(11):639–642, 2014. doi:10.1016/j.ipl.2014.05.008.

5 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. *Algorithmica*, 81(3):1247–1266, 2019.

6 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing figures under the uniform distribution. *Random Struct. Algorithms*, 54(3):413–443, 2019.

7 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. *ACM Transactions on Algorithms (TALG)*, 18(4):1–39, 2022.

8 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 164–173. ACM, 2014. doi:10.1145/2591796.2591887.

9 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, pages 1370–1379, 2005. doi:10.1137/S0097539702403244.

10 Eldar Fischer and Ilan Newman. Testing of matrix-poset properties. *Comb.*, 27(3):293–327, 2007. doi:10.1007/s00493-007-2154-3.

11 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

12 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

13 Igor Kleiner, Daniel Keren, Ilan Newman, and Oren Ben-Zwi. Applying property testing to an image partitioning problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):256–265, 2011. doi:10.1109/TPAMI.2010.165.

14 Simon Korman, Daniel Reichman, and Gilad Tsur. Tight approximation of image matching. *CoRR*, abs/1111.1713, 2011. arXiv:1111.1713.

15 Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. Fast-match: Fast affine template matching. In *CVPR*, pages 2331–2338. IEEE, 2013. doi:10.1109/CVPR.2013.302.

16 Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, September 2017. doi:10.7551/mitpress/11301.001.0001.

17 Pritam Paral, Amitava Chatterjee, and Anjan Rakshit. Vision sensor-based shoe detection for human tracking in a human–robot coexisting environment: A photometric invariant approach using dbscan algorithm. *IEEE Sensors Journal*, 19(12):4549–4559, 2019.

18 Sofya Raskhodnikova. Approximate testing of visual properties. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2003. doi:10.1007/978-3-540-45198-3_31.

19 Dana Ron and Gilad Tsur. Testing properties of sparse images. *ACM Trans. Algorithms*, 10(4):17:1–17:52, 2014. doi:10.1145/2635806.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

21 Andrew C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.