Applying Property Testing to an Image Partitioning Problem

Igor Kleiner, Daniel Keren, Ilan Newman, and Oren Ben-Zwi

Abstract—Property testing is a rapidly growing field of research. Typically, a property testing algorithm proceeds by quickly determining whether an input can satisfy some condition, under the assumption that most inputs do not satisfy it. If the input is "far" from satisfying the condition, the algorithm is guaranteed to reject it with high probability. Applying this paradigm to image detection is desirable since images are large objects and a lot of time can be saved by quickly rejecting images which are "far" from satisfying a certain condition the user is interested in. Further, typically most inputs are, indeed, "far" from the sought images. We demonstrate this by analyzing the problem of deciding whether a binary image can be partitioned according to a template represented by a rectangular grid, and introduce a quick "rejector," which tests an image extracted from the input image, but whose size, as well as the time required to construct it, are constants which are independent of the input image size. With high probability, the rejector dismisses the inputs which are "far" from the template.

Index Terms—Property testing, image partitioning.

1 Introduction

Images are very large objects; therefore, it may take a great deal of time to test whether a certain property holds for a given image. In recent years, *rejection-based* algorithms were introduced; they rely on the assumption that most input images are quite "far" from the images which satisfy the property and can be quickly rejected.

We suggest a rejection-based approach which follows the paradigm of the rapidly growing field of *property testing*. A typical property testing algorithm very quickly rejects all of the inputs which are "far" from satisfying the sought property. A trivial example is the following: We wish to accept only images whose average gray level is in a certain small interval. From simple probabilistic considerations, if the average of a small random sample of pixels from an image I is far from this interval, then—with a very high probability—I's average is also far from the interval.

Here, we address a more involved problem. Given an image, we wish to determine whether it can be partitioned into a rectangular grid such that its blocks approximately conform to a given template. For example, both images in Fig. 1 conform to the template represented by

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

(a definition of "conforming to a template" will be provided in Section 4, but the idea is obvious).

This problem is highly nontrivial due to the very large number of possible partitions. This type of problem is important for image database search (for example, one may wish to find all images in which there is sand at the bottom and sea at the top). Such a partitioning is often a preliminary step in content-based image retrieval (CBIR) and, more specifically, region-based image retrieval (RBIR) [18].

Another reason for choosing this problem is the following: We wish to demonstrate that property testing can be applied to reduce a decision problem in computer vision to a problem whose size is very small relative to the original problem. For some tasks, such as face detection, one can try to achieve such size reduction by using "representative regions" (as in [17]), which, with high probability, can be used to determine whether the image is a face or not. The nature of the partitioning problem discussed here excludes the existence of such regions because the partition lines can be located anywhere in the image; hence, such regions (e.g., eyes) can be practically anywhere in the image and have to be found first. We wish to demonstrate that, nonetheless, the size of the problem can be reduced.

Let us emphasize that we are not suggesting an algorithm to *solve* the partitioning problem, but only to *reduce its size* so that the "small" problem will be—with high probability—equivalent to the original problem. Specifically, we assume that an algorithm to answer queries about the existence of a partition *already exists*, but that its running time depends on the image size, and, when given an arbitrary image I, we construct from it an image of *constant size* I' such that the question of whether I can be partitioned in a manner conforming to a given template can be reduced to the question of whether the much smaller I' can be thus partitioned.

For simplicity and conciseness, we assume that the images are binary, but the analysis is general and can be extended to gray level and color images.

1.1 Structure of the Paper

Next, we survey previous work (Section 2), and shortly discuss property testing (Section 3). Some definitions are

• *The authors are with the Department of Computer Science, University of Haifa, Haifa 31905, Israel. E-mail: dkeren@cs.haifa.ac.il.*

Manuscript received 6 May 2009; accepted 22 Oct. 2009; published online 19 Aug. 2010.
Recommended for acceptance by D. Forsyth.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-05-0282.
Digital Object Identifier no. 10.1109/TPAMI.2010.165.

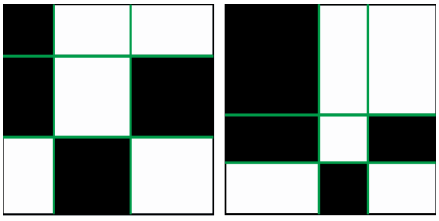

Fig. 1. Two images which conform to the same template.

provided in Section 4. In Section 5, the problem is defined and bounds on the sampling process derived. Section 6 includes the experimental results, and conclusions are offered in Section 7.

2 Previous Work

We are only aware of one paper, [12], which applies property testing to images. In that paper, binary images are treated as $0-1$ matrices, and the following questions are solved:

- Is the set of dark pixels a half-plane (i.e., the set of all pixels which lie above or below a straight line passing through the image)?
- Is the set of dark pixels convex?
- Is the set of dark pixels connected?

These questions are settled with an error probability smaller than $\frac{1}{3}$ by testing the values of s sample pixels; s does not depend on the size of the image but only on a user-defined parameter ϵ. The algorithm is guaranteed to accept images which satisfy the condition (e.g., being a half-plane) and to reject, with probability at least $\frac{2}{3}$, inputs which are more than ϵ-*far* from satisfying the condition (meaning that the values of at least an ϵ fraction of the pixels have to be flipped for the condition to hold; a more formal definition is provided in Section 3).

In [12], the conditions tested are rigid (e.g., even one pixel of the wrong color can cause a set to not be a half-plane). While such rigid conditions are usually easier to test, this definition may turn out to be too restrictive for real images due to noise or the presence of small objects of the "wrong" color; in this paper, we allow a "softer" definition (Section 4). Also, the allowable set of shapes we examine is very large, and not restricted to connected shapes.

Lindenbaum [11] deals with the question of how much data are required to recognize a rigid object in the presence of clutter and assuming the object has undergone some (e.g., affine) transformation. Our work differs in the fact that the transformation is not rigid (the images in Fig. 1 are not affinely equivalent) and in the general approach of using the property testing paradigm.

In recent years, there has been growing interest in image detection algorithms which are based on *fast rejection*. These algorithms use the observation that, typically, most inputs are far from satisfying the criteria for acceptance—e.g., in the case of face detection, large input images are scanned to see whether they contain faces—and most subimages are very far from being a face. Rejection-based algorithms usually employ a quick test (or a cascade of simple tests) which rejects most input images. Some examples are [2], [9], [14], [4], [8], [16], [15], [13].

The rejection-based approach shares some of the characteristics of *property testing*, which is a relatively new discipline within theoretical computer science [6], [5]. A typical property testing algorithm distinguishes with a certain probability (usually taken to be $\frac{2}{3}$) between an input with a certain property and an input which is "far" from satisfying the property; the sample size and/or the complexity depend on just how "far" it is. A more rigorous definition follows.

3 Property Testing and Rejection-Based Algorithms

We now present a rather brief introduction to property testing. For convenience, we replace the abstract notion of an "object" by that of an image.

The distance between two $n \times n$ binary images $\mathcal{I}$ and $\mathcal{J}$ is defined to be the number of pixels in which they differ. A property $\mathcal{P}$ is defined as some condition(s) which an image $\mathcal{I}$ can satisfy, in which case one also writes $\mathcal{I} \in \mathcal{P}$. The distance of an image $\mathcal{I}$ from the property is $\min_{\{\mathcal{J}|\mathcal{J}\in\mathcal{P}\}} d(\mathcal{I},\mathcal{J})$, where $d(\mathcal{I},\mathcal{J})$ is the distance between $\mathcal{I}$ and $\mathcal{J}$. The *relative distance* from $\mathcal{I}$ to $\mathcal{P}$ is the distance from $\mathcal{P}$ divided by the size of $\mathcal{I}$. This size will be assumed fixed hereafter, and for simplicity, it will be assumed that all images are binary and of size $n \times n$.

An image is δ-*far* from $\mathcal{P}$ if its relative distance to $\mathcal{P}$ is at least δ . A property is (m,δ)-*testable* if there is an algorithm that for every input $\mathcal{I}$ queries at most m pixels and, with probability at least $\frac{2}{3}$, distinguishes between images with the property and images which are δ-*far* from it.

One settles for the $\frac{2}{3}$ factor because a constant number of runs of the algorithm will reduce the error probability very quickly. If the test is run l times and we choose to reject or accept by a majority vote, an error can occur only if we err more than $\frac{l}{2}$ times in total, but the probability for a single error is $\leq \frac{1}{3}$. The probability of an error in the majority vote is known, in this case, to be bounded by $c_1 \frac{\exp(-c_2 l)}{\sqrt{l}}$ for some constants c_1, c_2, hence, it is of order $\exp(-\Theta(l))$.

4 Preliminaries

We now introduce some simple definitions relating to the property of being close and far from a given $k \times k$ binary partition.

Given an $n \times n$ binary matrix M (which represents an image), the set of columns of M is denoted by M_c, and the set of rows is denoted by M_r.

Definition 2.1. *Given an $n \times n$ binary matrix M, a* k-column partition *of M is a k-subset $S = \{1 \leq s_1, s_2, \ldots, s_k = n\}$ of $[n] = \{1, 2, \ldots, n\}$ (assume $s_0 = 0$). A* block *$B_i \subseteq M_c$ ($i \in [k]$) is the set of the columns $s_{i-1}+1, s_{i-1}+2, \ldots, s_i$. Define a* k-rows partition *in a similar manner.*

Definition 2.2. *Given an $n \times n$ binary matrix M, a $k \times k$-*partition *(hereafter called a partition) of M is the intersection of a k-column partition with a k-rows partition. The block $B_{i,j}$ of the partition is the intersection of the block B_i of the columns partition with the block B_j of the rows partition.*

Definition 2.3. *Given an $n \times n$ binary matrix M and a $k \times k$ binary matrix T called a* template *(usually $k \ll n$), we say that T* represents *M if there is a $k \times k$-partition of M to monochromatic blocks (blocks which have all entries in M with the same value), where every block $B_{i,j}$ in M has the same value as $T_{i,j}$.*

We note here that while the definition of "monochromatic" appears to be restrictive, it can be modified to, e.g., "having all gray levels between 80 and 90," without making any change in the algorithm, but we will stick with the simple definition as it is easier to follow. The second item in Section 7 briefly deals with more general treatment of gray-level images.

When dealing with real images, it is reasonable to relax Definition 2.3 by the following:

Definition 2.4. *Given an $n \times n$ binary matrix M, a $k \times k$ template T, and a $k \times k$-partition PT of M, PT is* ϵ-close *to T if no more than $\epsilon|B_{i,j}|$ pixels of every block $B_{i,j}$ in PT can be flipped, with the result being a partition satisfying Definition 2.3. Otherwise, we say that PT is* ϵ-far *from T.*

Definition 2.5. *Given an $n \times n$ binary matrix M and a $k \times k$ template T, we say that T is* ϵ-close *to representing M if there is a $k \times k$-partition PT of M, where PT is* ϵ-close *to T. Otherwise, we say that T is* ϵ-far *from representing M.*

It is reasonable to bound the minimal size of a block from below (else even one pixel will be considered a legitimate block, which clearly does not make sense). This motivates the following definition:

Definition 2.6. *Given an $n \times n$ binary matrix M and a parameter $0 < \mu < 1$, a partition PT of M is called a* μ-partition *(or* legal partition*) if every block in PT is of size at least $\mu n \times \mu n$.*

Definition 2.7. *Given an $n \times n$ binary matrix M and a $k \times k$ template T, we say that T is* (ϵ, μ)-close *to representing M if there is a $k \times k$ μ-partition of M which is* ϵ-close *to T. Otherwise, we say that T is* (ϵ, μ)-far *from representing M.*

Hereafter, we assume μ is fixed and deals only with μ-partitions, so we will use ϵ-close(far) instead of (ϵ, μ)-close(far). Also, we will usually drop the "close to represent..." and just say that M is close(far) from T.

5 Problem Definition and Analysis

Now, we can formalize the requirements from the algorithm:

Given an $n \times n$ binary input image M, a $k \times k$ template T, and three parameters, $\epsilon < \delta$ and μ, the output of the algorithm should be the following:

- If T is *ϵ-close* to M, the algorithm accepts with probability at least $\frac{2}{3}$.
- If T is *δ-far* from M, the algorithm rejects with probability at least $\frac{2}{3}$.

Typically, ϵ will be determined by the user as it controls how much the sought images can deviate from the given template. However, δ is a parameter that controls how many images will be rejected quickly versus the time and space complexity of the rejection stage. More on this in Sections 5.1.6, 5.1.7, and 6.

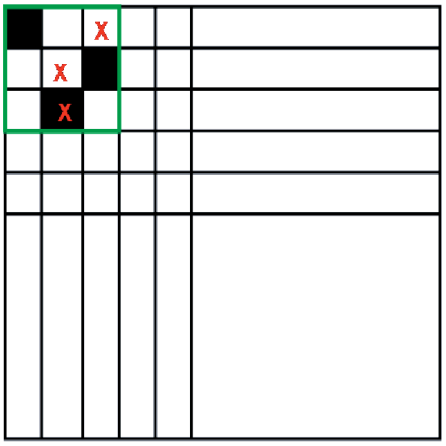

Fig. 2. Schematic description of the construction of M' given M for the case $\frac{n}{t} = 3$. The randomly sampled pixels are marked by a red X. The square outlined by the green edges represents the extent of $M'_{1,1}$, so $M'_{1,1} = (2, 3)$.

The algorithm commences by constructing a $t \times t$ image M', which is much smaller than M (t will depend only on ϵ, μ, and δ, and not on n, the size of M). The quick rejection stage searches for a partition of M', which is constructed as follows: First, we sample independently at random r pixels from M (where r does not depend on n either; both r and t will be determined in Section 5). Then, M is partitioned by a uniform $t \times t$ grid to a collection of cells $\mathcal{C}$, each of which contains $\frac{n}{t} \times \frac{n}{t}$ pixels of M. Then, the (i, j) entry of M' is defined to be a pair (s, w), where w is the number of samples which fell in the (i, j) cell of $\mathcal{C}$ and s is the number of 1s among them. Fig. 2 depicts a schematic description of how M' is constructed. Note that the time required to construct M' is a constant, independent of the size of M (as opposed to standard resolution reduction, which smooths an image and then subsamples it; this yields a small image, but the time required to construct it is proportional to the size of M).

As is evident from the definition and Fig. 2, the pixels of the constant-size image M' are not 0 or 1, but have two parameters each (the total number of samples which fell in the respective cell and the number of 1s among them). But, for the partitioning problem, this makes no difference; all we need to know is the percentage of 0s in a certain region, and that is trivial to compute from the pixels of M'.

It remains to choose the values of r and t. We achieve this by first computing the probability that the quick rejection stage fails, and then we proceed to set r and t so that this probability is below $\frac{1}{3}$.

5.1 Analysis

We split the analysis into two cases: first, assume M is *δ-far* from T, then assume M is *ϵ-close* to T. We show in both cases that the error probability is less than $\frac{1}{3}$.

5.1.1 Outline

Define $\delta' = \frac{9\delta+\epsilon}{10}$. In Sections 5.1.2 to 5.1.4, we will prove that for a suitable choice of t and r; if M is *ϵ-close* to a template T, then, with probability $\geq \frac{2}{3}$, M' is not *δ'-far* from T, and that if

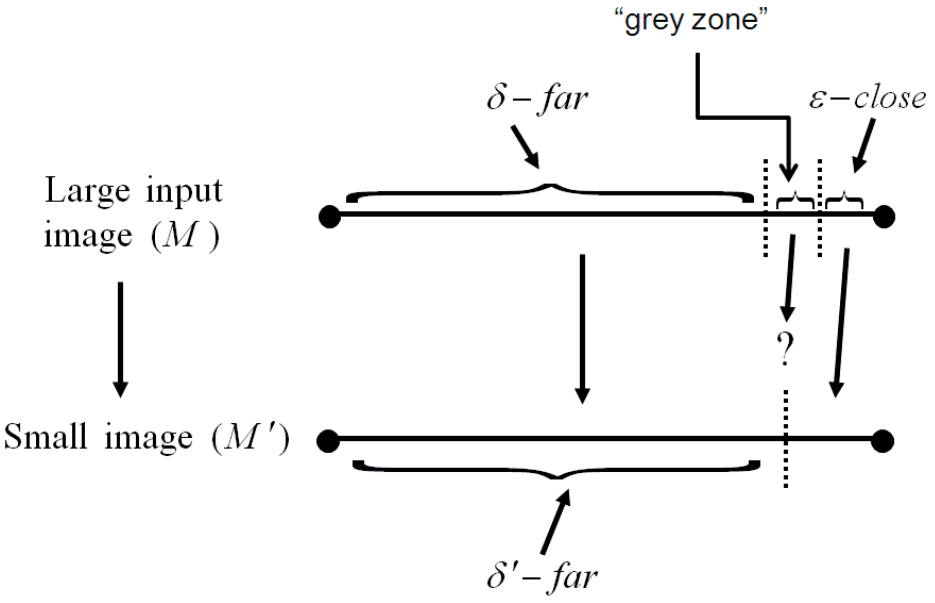

Fig. 3. A schematic description of the approach. δ-*far* and ϵ-*close* inputs are separated by whether the constant-size images constructed from them are, with high probability, δ'-*far* or not. We do not know what happens when the input is neither δ-*far* nor ϵ-*close*; the idea is to choose δ, so that not many such images exist (Sections 5.1.6, 5.1.7).

M is δ-*far* from T, then, with probability $\geq \frac{2}{3}$, M' is δ'-*far* from T (hereafter we will usually assume T is fixed and use ϵ-*close(far)* instead of ϵ-*close(far)* to T).

Therefore, δ' serves as a threshold value, which separates between δ-*far* and ϵ-*close* inputs (see Fig. 3). We want that, with high enough probability, if the input $n \times n$ image M is δ-*far*, then the constant-size image built from it, M', will be δ'-*far*; and that if M is ϵ-*close*, then, with high enough probability, M' will not be δ'-*far*. After this goal is obtained, the algorithm proceeds as follows: Given M, sample from it to construct M'. If M' is δ'-*far*, cast M away (remember that we are interested only in ϵ-*close* inputs and, with high probability, these do not become δ'-*far*). If M' is not δ'-*far*, then M passed the "quick rejection stage," and it has to be tested to see whether it is ϵ-*close*. This test can be lengthy since it is performed on the large input image M; our goal is to choose a good value for δ so that not too many Ms will pass the quick rejection stage. As δ gets closer to ϵ, fewer Ms will pass this stage—but, as we will see in Section 5.1.4, the closer δ gets to ϵ, the larger M' becomes and the more pixels have to be sampled.

What about images that are neither ϵ-*close* nor δ-*far*? These images lie in a "gray zone" and we cannot predict whether they will pass the quick rejection stage or not. The idea—common in both property testing and quick rejection methods developed for image detection—is to choose a δ so that most input images will be δ-*far*; this being possible since most input images are far from conforming to the template. We'll get back to the question of choosing δ in Sections 5.1.6 and 5.1.7.

Why is δ' chosen to be $\frac{9\delta+\epsilon}{10}$? First, note that due to the sampling process, we have to relax the δ-*far* condition, and indeed $\delta' < \delta$. Other values between ϵ and δ are possible (more on this in Section 5.1.4).

5.1.2 Far

First, we assume M is δ-*far* and show that the algorithm rejects with probability at least $\frac{2}{3}$.

Let *Blocks* be the set of all legal blocks in M' (a block is legal iff each of its sides is at least a portion μ of the entire image—see Definition 2.6). A simple upper bound is $|Blocks| \leq t^4$. We denote by $\#sam_b$ the number of samples in a block b.

In order to obtain the required bounds, we have to demand that a minimal number of samples fall in each legal block. The expected value of the number of samples in the smallest possible block is $\mu^2 r$ (recall that there are r samples altogether). We demand that, with high probability, every block will contain at least $\frac{\mu^2 r}{2}$ samples, denoting $\tau = \frac{\mu^2 r}{2}$. A sample in which each block has at least τ samples is called "good," else it is called "bad."

We union bound the probability for a bad sample by summing over all possible blocks to obtain:

$$\begin{aligned}\Pr(\textbf{bad sample}) &\leq \sum_{b \in Blocks} \Pr(\#sam_b \leq \tau) \\ &= \sum_{b \in Blocks} \Pr(\#sam_b \leq 2\tau - \tau).\end{aligned}$$

Applying Chernoff's inequality [3] (we use the form $\Pr(X < (1-\alpha)E(X)) < \exp(-\frac{\alpha^2 E(X)}{2})$, and note that the expectation for the number of samples in the smallest possible block is 2τ) yields:

$$\Pr(\textbf{bad sample}) \leq t^4 e^{-\mu^2 r/8}. \quad (1)$$

Next, we assume that the original image M is δ-*far*, and prove that if enough samples are taken, the constant-size image M' is with high probability δ'-*far*. This is an important result from a practical point of view since it guarantees that it is enough to slightly relax the condition

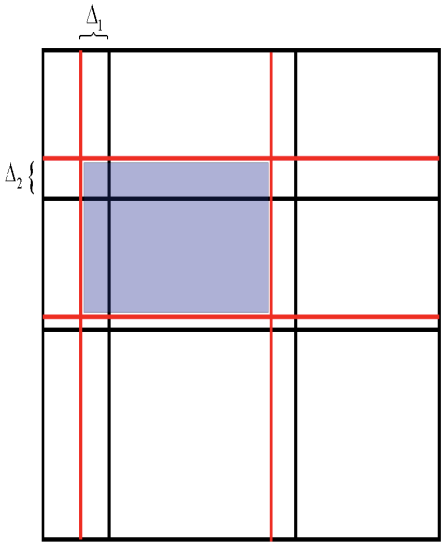

Fig. 4. A schematic description for the analysis in Section 5.1.3.

for an input image to be far, in order to guarantee that the constant-size image built from it will also be far—and the assumption is that the large majority of inputs are indeed far; this assumption is borne out in practice (Section 5.1.7).

To bound the probability that M' is δ'-*far*, note that every legal partition of M has at least one δ-*far* block. Every partition of M' induces a partition of M. To obtain the bound, we will use only one δ-*far* block in this induced partition and assume that only τ samples fell in it (clearly, these two assumptions can only increase the probability since τ is a lower bound on the number of block samples in a "good" sample).

Using Chernoff's inequality again, the probability that there will be no more than a ratio of δ' "bad pixels" in the sampled block is bounded as follows:

$$\begin{aligned}
&\Pr(\#(\textbf{badpixels}) \leq \tau\delta') \\
&\quad = \Pr\left(\#(\textbf{badpixels}) \leq \tau\delta - \frac{\tau(\delta-\epsilon)}{10}\right) \\
&\quad = \Pr\left(\#(\textbf{badpixels}) \leq \tau\delta\left(1 - \frac{\delta-\epsilon}{10\delta}\right)\right) \\
&\quad \leq \exp\left(-\frac{1}{2}\frac{\mu^2 r\delta}{2}\left(\frac{\delta-\epsilon}{10\delta}\right)^2\right);
\end{aligned}$$

this has to be multiplied by t^4 (to account for all blocks), yielding the bound

$$t^4 \exp\left(-\frac{1}{2}\frac{\mu^2 r\delta}{2}\left(\frac{\delta-\epsilon}{10\delta}\right)^2\right). \tag{2}$$

5.1.3 Close

Next, we bound the probability that an ϵ-*close* M will yield a δ'-*far* M'. In Fig. 4, let the dark lines denote an ϵ-*close* partition P of M, and let the red lines be those of a $t \times t$ "coarse" partition (that is, a partition of M'), which are the closest from above (left) to the horizontal (vertical) lines of P (Δ_1, Δ_2 denote the distances between the lines). Obviously, $\Delta_1, \Delta_2 \leq \Delta = \frac{n}{t}$. Let us look at a block of this coarse partition, shaded in blue. One may verify that the percentage of "bad" pixels in it is bounded from above by

$$\frac{\epsilon(\mu n)^2 + \Delta_1(\Delta_2 + \mu n) + \Delta_2(\Delta_1 + \mu n)}{(\mu n - \Delta + \Delta_1)(\mu n - \Delta + \Delta_2)} \tag{3}$$

(recall that the original block, bounded by the dark lines, has a ratio of at most ϵ "bad" pixels. Assume that the rectangular regions added to it contain only "bad" pixels, and that its area was the smallest possible, i.e., $\epsilon(\mu n)^2$. The expression in (3) is therefore an upper bound on the percentage of "bad" pixels in the block).

Clearly, the expression in (3) is bounded from above by

$$\frac{\epsilon(\mu n)^2 + 2\Delta(\Delta + \mu n)}{(\mu n - \Delta)^2}. \tag{4}$$

Define the size of the small image, M', to be $t = \frac{5}{\mu(\delta-\epsilon)}$, yielding $\Delta = \frac{n}{t} = \frac{\mu n(\delta-\epsilon)}{5}$. Then, the bound in (4) assumes the form

$$\frac{\epsilon + \frac{2(\delta-\epsilon)}{5}\left(1 + \frac{\delta-\epsilon}{5}\right)}{\left(1 - \frac{\delta-\epsilon}{5}\right)^2}. \tag{5}$$

A lengthy exercise involving some inequalities and calculus (which will be omitted) proves that the expression of (5) is bounded from above by $\frac{8\delta+2\epsilon}{10}$ (recall that, obviously, we assume $\delta > \epsilon$).

So, we proved that the expectation of the percentage of "bad" pixels in a block of the coarse image is no more than $\frac{8\delta+2\epsilon}{10}$. To bound the probability that the percentage in the sample from the block exceeds $\delta' = \frac{9\delta+\epsilon}{10}$, we again use a Chernoff bound to obtain (after multiplying by the number of blocks, k^2), that this probability is bounded by

$$\begin{aligned}
&k^2 \Pr(\#(\textbf{badpixels}) \geq \tau\delta') \\
&\quad = k^2 \Pr\left(\#(\textbf{badpixels}) \geq \frac{\mu^2 r}{2}\left(\frac{9\delta+\epsilon}{10}\right)\right) \\
&\quad = k^2 \Pr\left(\#(\textbf{badpixels}) \geq \frac{\mu^2 r}{2}\left(\frac{8\delta+2\epsilon}{10}\right)\left(\frac{9\delta+\epsilon}{8\delta+2\epsilon}\right)\right) \\
&\quad = k^2 \Pr\left(\#(\textbf{badpixels}) \geq \frac{\mu^2 r}{2}\left(\frac{8\delta+2\epsilon}{10}\right)\left(1 + \frac{\delta-\epsilon}{8\delta+2\epsilon}\right)\right) \\
&\quad \leq k^2 \exp\left(-\frac{\mu^2 r}{120}\frac{(\delta-\epsilon)^2}{4\delta+\epsilon}\right)
\end{aligned} \tag{6}$$

(this bound was obtained by the upper tail bound $Pr(X > (1+\alpha)E(X)) < \exp(-\frac{\alpha^2 E(X)}{3})$, which is correct, for $0 \leq \alpha \leq 1$ [7]).

Note that all the bounds (1), (2), (6) do not depend on n, the size of the original image. Intuitively, this is due to the fact that the property, we are testing, does not depend on n either, but only on the parameters k, δ, ϵ, μ.

5.1.4 Values for t and r

Given δ, we choose t, the size of M', and the number of sampled pixels r, so as to ascertain that the probability that M is δ-*far* and M' is not δ'-*far*, and the probability that M is ϵ-*close* and M' is δ'-*far*, are both smaller than $\frac{1}{3}$ (as indicated earlier, to render these probabilities small to an arbitrary degree, we run the tests l times, resulting in error

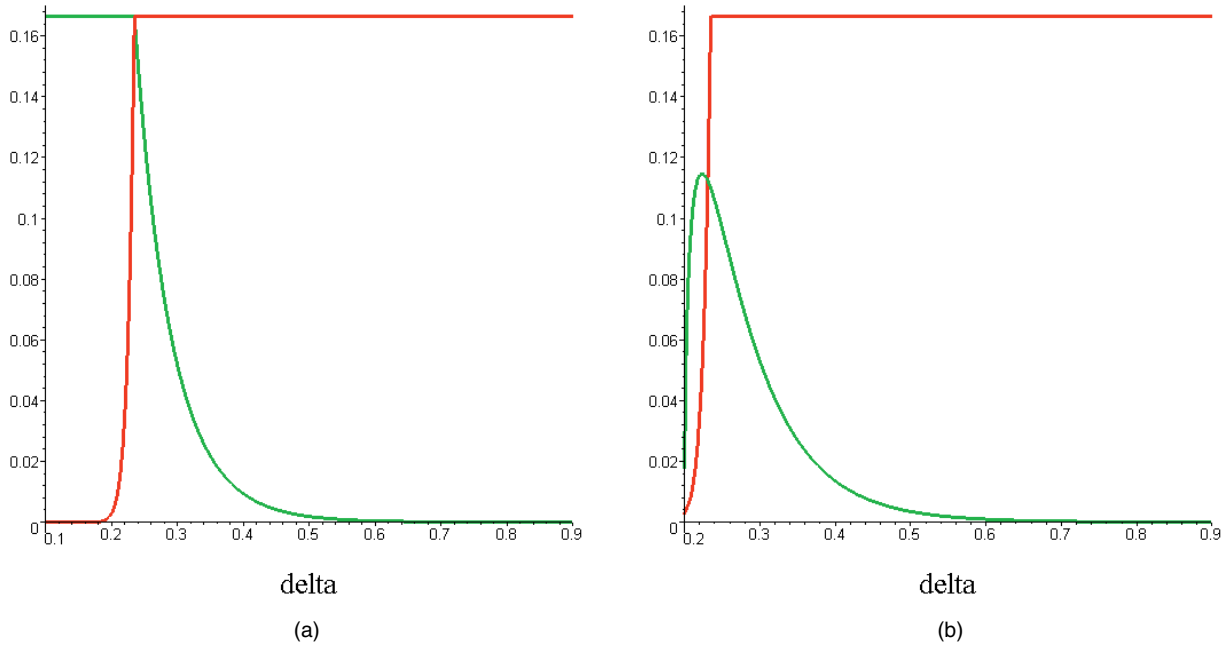

Fig. 5. (a) Plots of *false negative error* (green) and *false positive error* (red) for $k = 5$, $\epsilon = 0.05$, $\mu = 0.1$, and δ between 0.1 and 0.9. (b) The same as,(a) for $k = 4$, $\epsilon = 0.2$, $\mu = 0.1$, and δ between 0.2 and 0.9. In both cases, and for all values of δ, *small sample error* was much smaller than both of the plotted error probabilities, never exceeding 10^{-12}.

probabilities of $\exp(-\Theta(l))$; therefore, a constant number of runs reduces the error probability by an exponential factor). Since the bounds were derived under the assumption that the sample is "good," we demand that the bound in (1) be smaller than $\frac{1}{6}$, and we demand the same for the bounds in (2) and (6). Altogether, this yields the following requirements, obtained by extracting r from the inequalities defined by demanding that the expressions in (1), (2), (6) are smaller than $\frac{1}{6}$:

$$\begin{aligned} t &= \frac{5}{\mu(\delta-\epsilon)}, \\ r &> \frac{8c + 32\log(t)}{\mu^2}, \\ r &> \frac{400[c + 4\log(t)]\delta}{\mu^2(\delta-\epsilon)^2}, \\ r &> \frac{120[c + 2\log(k)](4\delta+\epsilon)}{\mu^2(\delta-\epsilon)^2}, \\ &\text{where } c = \log(6). \end{aligned} \tag{7}$$

As noted, the size of M' and the number of samples used to construct it do not depend at all on the size of the original image M. They do increase as the template size k increases, and as the relative length/width of the smallest allowable blocks μ, and the difference $\delta - \epsilon$, decrease. As δ approaches ϵ, it is harder to distinguish ϵ-*close* images from δ-*far* images.

One can use the derivations presented here to construct other bounds; for example, taking δ' to be closer to ϵ, or taking t to be larger (say, $\frac{10}{\delta-\epsilon}$) will reduce the number of samples. The optimal choice of bounds depends on what the user prefers (the tradeoff between smaller M' versus more samples), and the probability of a random image to be δ-*far* from the template. We touch on this formidable question in Sections 5.1.6, 5.1.7.

5.1.5 A Closer Look at the Bounds

In this section, we look at some typical values of the error probability bounds. There are three possible types of errors:

- *Small sample error* (1): The sample does not have enough pixels in each $M^{'}$ block in order to satisfy the assumptions guaranteeing the correctness of the next two bounds.
- *False positive error* (2): M is δ-*far* but $M^{'}$ is not δ'-*far*.
- *False negative error* (6): M is ϵ-*close* but $M^{'}$ is δ'-*far*.

In Section 5.1.4, we provided bounds for t (the size of M') and r (the number of samples from M used to construct M'). These bounds guarantee that all three probabilities are smaller than $\frac{1}{6}$, thus the overall probability for false negatives/positives is smaller than $\frac{1}{3}$. However, for typical values of the problem parameters $(k, \mu, \delta, \epsilon)$, at least one of the bounds is usually much smaller than $\frac{1}{6}$; hence, the actual error probabilities are small, and so it is usually not required to run many tests before reaching a decision about the input image. It also allows a simple and efficient algorithm, based on cascading the value of the user-determined parameter δ, to quickly and accurately reject inputs, which are far. Before describing this algorithm, some examples of the three bounds, for fixed input parameters k, μ, ϵ and varying values of δ, are presented in Fig. 5 (the rationale behind choosing δ as the varying parameter is that it is chosen by the user). Note that the real values of the bounds can be easily calculated for each set of parameters, using (1), (2), and (6).

5.1.6 Cascading the Value of δ

As demonstrated in Fig. 5, for large values of δ, the type of error we want to prevent—false negatives—is very small. This suggests the following algorithm: Start with a large δ_0 (e.g., 0.9). Recalling that a larger δ results in a smaller t (the

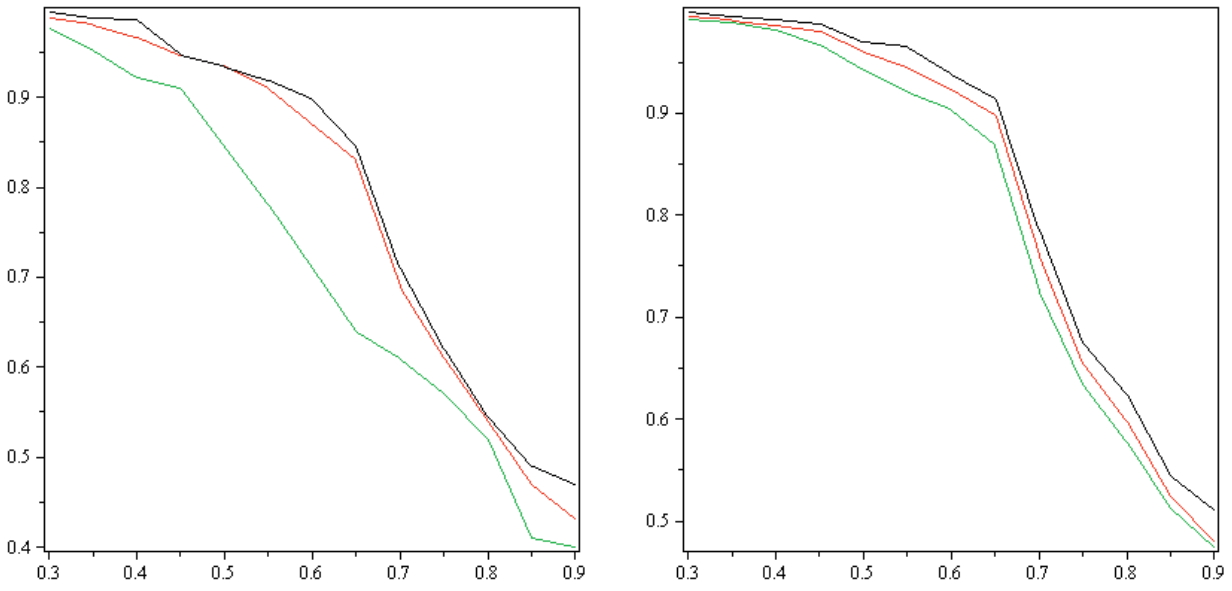

Fig. 6. Plots of the percentage of binary images which are δ-*far* from the template.

first of (7)), this step can be performed quickly, since the main bottleneck for testing M' is its size t. All of the images which have passed the initial value δ_0 are tested with a smaller value, δ_1, and so on. Input images which do not conform to the template T will be gradually discarded, with those farther from T (δ-*far* for larger δ) being rejected in earlier stages, at a lower computational cost; also, the probability that this process erroneously discards images which match the template is exceedingly small. A greater computational effort must be dedicated to images which have made it through more stages of the cascade since t will then be larger; however, as we empirically demonstrate in Section 5.1.7, only a very small fraction of images make it through.

In order to determine the efficiency of the cascade algorithm, we need to know what percentage of the inputs will be rejected for each value of δ. Note that this question hardly depends on the value of the user-supplied ϵ since the algorithm is designed to reject images which are δ'-*far*, where $\delta' = \frac{9\delta+\epsilon}{10}$; clearly, δ' is barely influenced by ϵ.

5.1.7 Percentage of Natural Images which are δ-far from a Given Template

In this section, we empirically test, for various values of δ and a few templates T, the percentage of natural images which are δ-*far* from T. Knowledge of this percentage allows us to design the δ-cascade described in Section 5.1.6 and to estimate the running time.

The question, therefore, is: For a given template T and values of μ and δ, what percentage of inputs are δ-*far* from T (when only μ-*partitions* are considered)? The answer can possibly be estimated from work on the distribution of natural images [10], [19], but it is beyond the scope of this paper. Instead, we calculated it in some test cases. The results computed for 5,000 binary images (some of which are depicted in Fig. 7) are presented in Fig. 6.

As the results demonstrate, candidates for the more "complicated" pattern are easier to reject than candidates for the simpler pattern. The reason for this is clear—there are more images, which are close to a "simple" pattern than to a "complicated" one. Last, note that the large majority of input images are rejected quickly, requiring a very small amount of computation.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{(left)} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \text{(right)}$$

Horizontal axis represents δ, vertical is percentage. Green is for $\mu = 0.05$, red is $\mu = 0.1$, and black is $\mu = 0.15$. Note that, for the more "complicated" template on the right, a higher percentage of natural images are farther from it and, hence, will be rejected quickly.

Fig. 7. Some of images used to compute the empirical statistics of the quick rejection rate.

TABLE 1
Ratios of False Negatives for Values of t Smaller than t_0, the Bound in (7)

Template B	$\mu=0.2$			$\mu=0.15$			$\mu=0.1$		
$\epsilon=0.1$	0.001	0.013	0.042	0.002	0.023	0.052	0.003	0.029	0.064
$\epsilon=0.2$	0.0006	0.009	0.024	0.001	0.019	0.043	0.002	0.008	0.042
$\epsilon=0.3$	0.0003	0.005	0.018	0.001	0.009	0.018	0.003	0.009	0.023

Each entry contains the result for $0.8t_0$ (left), $0.6t_0$ (center) and $0.4t_0$ (right).

6 Experiments

In order to test the compatibility of the theoretical analysis with real images, we ran tests on 5,000 binary images, all of which were larger than $1{,}000 \times 1{,}000$ pixels (average number of pixels was $2.43 \cdot 10^6$), and compared the results to those obtained on the subsampled, constant-size images.

6.1 Experimental Setup

In order to ascertain the correctness of the experiments, the testing was "brute force": In order to test whether an image conforms to a given $k \times k$ template, we checked all possible partitions of the image to a $k \times k$ grid. It is intractable to actually go through all partitions (for example, for $k = 4$ and $\mu = 0.1$, a $1{,}000 \times 1{,}000$ image has about $2.5 \cdot 10^{21}$ legal partitions, from a total of 10^{24} partitions). The computational complexity, however, can be reduced by observing that, given a partition and a block which is δ-*far* from the appropriate block in the template, it is possible to provide a simple lower bound on how far the block will become after one of its partition lines is translated: If the original block's area was A and δ of its pixels (that is, δA) were of the wrong color, then, if its area is increased by B, at least $\frac{\delta A}{A+B}$ pixels in the new block will also be of the wrong color. Thus, not all partitions should be checked. However, the computational burden for large images is still very high. We used two strong servers and a PC cluster for the experiments. The cascade algorithm for successive rejection (Section 5.1.6) was used, with δ starting at a value of 0.9 and being reduced by 0.1 at every stage. The following templates were tested, with various values of μ and ϵ:

$$\mathrm{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \ \mathrm{B} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \ \mathrm{C} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathrm{D} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathrm{E} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

6.2 Results

Since input images which make it through all stages of the cascade rejector are rigorously tested (that is, not the reduced image, but the image itself is checked against the template), no "false positives" are possible in this final testing stage. This could be modified—e.g., we can decide to reject an image with a parameter ϵ by testing a δ which is very close to ϵ—but that is not a crucial difference since such images, according to the first of (7), will be rather large anyway, and testing them does not save a great deal of computation versus testing the original. The error percentage of this heuristic depends on the size of the "gray zone" (Fig. 3 and discussion in Section 5.1.1) of images, which are not ϵ-*close* nor δ-*far*.

Thus, the correctness is measured by the number of "false negatives," that is, images M which are ϵ-*close* to the template but for which the reduced-size image M' is rejected for some δ.

As noted in Section 5.1.5, the probability for such "false negatives" is very low for large δ, and is bounded by $\frac{1}{6}$ for smaller values of δ when only a very small percentage of images remain (since *small sample error* is very small for typical parameter values). This allows us to run the testing stage for small δ a few (e.g., five) times, reducing the probability of error to $(\frac{1}{6})^5 = 1.28 \cdot 10^{-4}$ without significantly affecting the overall running time. Also, $\frac{1}{6}$ is a theoretical bound, computed assuming a worst- case scenario (that all of the "missed area" in the proof in Section 5.1.3 is of the wrong color). The "false negative" probability is therefore very small, and false negatives were not encountered in any of the experiments we ran.

6.3 Further Reducing the Size of M'

Since using the image size t and sample complexity r of (7) yielded no errors, a natural question is: How much "slack" is present in the bounds for r and t? Especially, can t be reduced while maintaining fidelity? To test this, we ran experiments not only with the value of t as given in (7), but also with smaller values, and checked the results against the ground truth. The results for the templates B, D, E are presented in Tables 1, 2, and 3 for some values of μ and ϵ. Each table entry contains the percentage of input images which conform to the template, but were erroneously classified as negative, for values of t which are $0.8t_0$ (left), $0.6t_0$ (center), and $0.4t_0$ (right), where t_0 is the theoretical bound. Recall that there were no such misclassifications for the correct $t = t_0$ value.

TABLE 2
The Same as Table 1, for a 4×4 Template

Template D	$\mu=0.2$			$\mu=0.15$			$\mu=0.1$		
$\epsilon = 0.1$	0.002	0.017	0.048	0.002	0.024	0.057	0.005	0.031	0.068
$\epsilon = 0.2$	0.001	0.011	0.026	0.001	0.022	0.032	0.003	0.018	0.044
$\epsilon = 0.3$	0.001	0.006	0.021	0.001	0.009	0.022	0.002	0.011	0.033

TABLE 3
The Same as Table 1, for a 5×5 Template

Template E	$\mu=0.15$			$\mu=0.1$			$\mu=0.05$		
$\epsilon = 0.1$	0.002	0.019	0.051	0.002	0.027	0.059	0.007	0.033	0.075
$\epsilon = 0.2$	0.001	0.015	0.032	0.002	0.023	0.033	0.003	0.019	0.048
$\epsilon = 0.3$	0.002	0.007	0.025	0.002	0.011	0.023	0.003	0.013	0.043

As these results indicate, the algorithm performs with reasonable accuracy for reduced images whose sizes are smaller than the theoretical bound. The accuracy decreases when the template size k increases, when the minimal block area decreases, and most importantly, when the factor by which the theoretical bound is reduced decreases. Still, at a value of only 40 percent of the theoretical bound, the accuracy is reasonable. In Fig. 8, an example of a misclassified image is provided.

7 Some Possible Extensions

- *3D data*: All of the definitions and theorems in this paper can be immediately extended to 3D data (such as medical images and video spatiotemporal slabs). The only modification of the bounds in (7) is that the power of μ will be 3 instead of 2, and that the coefficients of $\log(t)$ will be multiplied by a factor of $\frac{3}{2}$ (since there are now t^6 possible blocks).
- *Grey-level images*: The treatment of gray-level images proceeds much like for the binary case. One can use the Hoeffding-Azuma inequality [1], which is better suited to a continuous and bounded domain. For example, one may define a partition of a gray-level image as one in which the average of a certain block is bounded in some interval, and the probability that this condition holds needs to be approximated from a small sample. The following form of the Hoeffding-Azuma inequality can be used, where X_i are the grey-levels of the samples: If $X_1 \ldots X_n$ are random variables bounded by B, then $\Pr(\sum_{i=1}^{n} X_i \geq L) \leq \exp(\frac{-L^2}{2nB^2})$ (a further condition is that distinct X_i's be uncorrelated, but this can be achieved by normalizing them to zero mean). Thus, if B is a hypothetical bound on the gray-levels of the block, its validity can be tested by the above inequality, proceeding along similar lines as for binary images.
- *More general shapes or partitions*: Dealing with more general partitions—for example, by allowing partitioning lines which are not straight—can be dealt with in much the same way as in this work since it is area-based; the shapes of the regions do not make a real difference.

Fig. 8. An image, which conforms to template D ($\epsilon = 0.2, \mu = 0.15$) but which failed the test for the value of t which is 0.4 of the theoretical bound.

8 Conclusions and Future Research

We presented a sampling scheme based on the idea of property testing that can be used to reduce the decision whether an input image of arbitrary size conforms to a certain template to the same decision for an image whose size depends only on the size of the template, the area of the

minimal partition block allowed, and a measure of correspondence with the template—not on the input size. Extensive experiments were carried out to test the validity of the method and the tightness of the bound on the reduced-size images. A simple cascade algorithm was presented which very quickly rejects inputs that are farther away from the template.

Future work will address extending the testing scheme to more general cases (Section 7), as well as applying property testing to other computer vision problems.

ACKNOWLEDGMENTS

This paper greatly benefited from the comments and corrections of four anonymous reviewers. This research was supported by Israel Science Foundation grants 1011/06 and 1220/04, and Israel Ministry of Science grant 3/3422.

REFERENCES

[1] N. Alon and J. Spencer, *The Probabilistic Method.* John Wiley and Sons, 2000.
[2] S. Baker and S.K. Nayar, "Pattern Rejection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 544-549, 1996.
[3] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Statistics,* vol. 23, pp. 493-507, 1952.
[4] M. Elad, Y. Hel Or, and R. Keshet, "Rejection Based Classifier for Face Detection," *Pattern Recognition Letters,* vol. 23, no. 12, pp. 1459-1471, Oct. 2002.
[5] E. Fischer, "The Art of Uninformed Decisions: A Primer to Property Testing," *Bull. European Assoc. for Theoretical Computer Science, Computational Complexity Column,* vol. 75, pp. 97-126, 2001.
[6] O. Goldreich, S. Goldwasser, and D. Ron, "Property Testing and Its Connection to Learning and Approximation," *J. ACM,* vol. 45, pp. 653-750, 1998.
[7] T. Hagerup and C. Rueb, "A Guided Tour of Chernoff Bounds," *Information Processing Letters,* vol. 33, no. 6, pp. 305-308, 1990.
[8] Y. Hel-Or and H. Hel-Or, "Real-Time Pattern Matching Using Projection Kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 9, pp. 1430-1445, Sept. 2005.
[9] D. Keren, M. Osadchy, and C. Gotsman, "Antifaces: A Novel, Fast Method for Image Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 7, pp. 747-761, July 2001.
[10] A.B. Lee, K.S. Pedersen, and D. Mumford, "The Nonlinear Statistics of High-Contrast Patches in Natural Images," *Int'l J. Computer Vision,* vol. 54, nos. 1-3, pp. 83-103, Aug. 2003.
[11] M. Lindenbaum, "An Integrated Model for Evaluating the Amount of Data Required for Reliable Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 11, pp. 1251-1264, Nov. 1997.
[12] S. Raskhodnikova, "Approximate Testing of Visual Properties," *Proc. Sixth Int'l Workshop Approximation Algorithms for Combinatorial Optimization Problems,* pp. 370-381, 2003.
[13] M. Ratsch, G. Teschke, S. Romdhani, and T. Vetter, "Wavelet Frame Accelerated Reduced Support Vector Machines," *IEEE Trans. Image Processing,* vol. 17, no. 12, pp. 2456-2464, Dec. 2008.
[14] S. Romdhani, P. Torr, B. Schoelkopf, and A. Blake, "Efficient Face Detection by a Cascaded Support-Vector," *Proc. Royal Soc. London,* vol. 460, no. 2501, pp. 3283-3297, 2004.
[15] H. Sahbi and D. Geman, "A Hierarchy of Support Vector Machines for Pattern Detection," *J. Machine Learning Research,* vol. 7, pp. 2087-2123, 2006.
[16] J. Sun, J.M. Rehg, and A.F. Bobick, "Automatic Cascade Training with Perturbation Bias," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 276-283, 2004.
[17] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, May 2004.
[18] J. Vogel and B. Schiele, "Semantic Modeling of Natural Scenes for Content-Based Image Retrieval," *Int'l J. Computer Vision,* vol. 72, no. 2, pp. 133-157, Apr. 2007.
[19] Y. Weiss and W.T. Freeman, "What Makes a Good Model of Natural Images?" *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

Igor Kleiner is a PhD candidate under the supervision of Professors Ilan Newman and Yuri Rabinovich in the Computer Science Department at the University of Haifa. His research interests include finite metric spaces and image processing.

Daniel Keren received the PhD degree from the Hebrew University in 1991, and then spent three years as a postdoctoral researcher at Brown University. Since then, he has been with the Computer Science Department at the University of Haifa. His main research interests include computer vision, regularization, and monitoring of large-scale distributed systems.

Ilan Newman received the BSc and MSc degrees from the Technion and the PhD degree from the Computer Science Department at the Hebrew University in 1992. He has been with the faculty at the University of Haifa since 1992. His current research interests include finite metric embeddings, combinatorial algorithms, property testing, and computational complexity.

Oren Ben-Zwi is a PhD candidate under the supervision of professor Ilan Newman in the Computer Science Department at the University of Haifa. His research interests include algorithmic game theory, property testing, combinatorics, and theory of computer science.