# Detection of Textured Areas in Images Using a Disorganization Indicator Based on Component Counts

Ruth Bergman, Hila Nachlieli, Gitit Ruckenstein
HP Laboratories Israel

HPL-2005-175(R.1)
March 26, 2007*

texture detection, texture analysis, texture synthesis, image analysis

An algorithm is presented for the detection of textured areas in digital images. Texture detection has potential application to image enhancement, tone correction, defect detection, content classification and image segmentation. For example, texture detection may be combined with a color model and other descriptors to detect objects in an image, such as sky, which is generally smooth, and foliage, which is textured.

The texture detector presented in this paper is based on the intuition that texture in a natural image is "disorganized". The measure we developed to detect texture examines the structure of local regions of the image. This structural approach enables us to detect both structured and unstructured texture at many scales. Furthermore, it distinguishes between edges and texture, and also between texture and noise. Automatic detection results are shown to match human classification of corresponding image areas.

# Detection of Textured Areas in Images Using a Disorganization Indicator Based on Component Counts

Ruth Bergman,[*] Hila Nachlieli,[†] and Gitit Ruckenstein[‡]
*Hewlett Packard*

(Dated: March 26, 2007)

An algorithm is presented for the detection of textured areas in digital images. Texture detection has potential application to image enhancement, tone correction, defect detection, content classification and image segmentation. For example, texture detection may be combined with a color model and other descriptors to detect objects in an image, such as sky, which is generally smooth, and foliage, which is textured.

The texture detector presented in this paper is based on the intuition that texture in a natural image is "disorganized". The measure we developed to detect texture examines the structure of local regions of the image. This structural approach enables us to detect both structured and unstructured texture at many scales. Furthermore, it distinguishes between edges and texture, and also between texture and noise. Automatic detection results are shown to match human classification of corresponding image areas.

PACS numbers:

## I. INTRODUCTION

Consumers have definite preferences and expectations regarding the appearance of images. They prefer blue sky, lively green vegetation, smooth skin and sharp eyes. Unlike a person, image capture devices and image editors do not understand the content of the image. Instead of image understanding, specific image analysis tools are used to guide the selection of enhancement parameters. Today's image enhancement algorithms automatically adjust to smooth areas, image edges and noise. These algorithms usually do not discriminate texture from edges or smooth regions. Hence, when they treat texture it is frequently blurred (as if a smooth area) or over-sharpened (as if an edge). This paper proposes an image analysis tool for texture detection.

Tuceryan and Jain stipulate that "We recognize texture when we see it but it is very difficult to define" [1]. They go on to quote six different definitions of texture. Many definitions include the notion of repetition and local order of a so-called *texture element*. On the other hand natural texture has some randomness in the location, size, color, orientation, etc. of the texture element. Figure 1 shows a variety of image textures. Some of the textures are dominated by a repeating pattern and are therefore considered structured. Unstructured textures are dominated by randomness and are sometimes indistinguishable from noise. The same texture can look significantly different at different scales, as illustrated in Figure 2.

The definition of texture we use in this work is based on the observation that an image is perceived as a collection of segments, where each segment can be coarsely classified as either smooth or textured. In natural images, a smooth area typically contains gradual color change, as well as some noise. We therefore define an area as textured if it is neither smooth nor contains an edge between smooth image segments. The identification algorithm should distinguish between textures comprised of many edges, potentially with large intensity differences, and edges between different segments. It should also identify the difference between areas of unstructured texture and areas containing noise or weak edges.

Our texture detector differs from other research work on descriptors for texture analysis and synthesis in several ways. First, the obvious choices for detecting high activity, such as local contrast measures and high-pass filters, typically do not distinguish between texture and image edges. This distinction is the primary achievement of the detector described in this paper. Second, most texture descriptors respond to noise and texture similarly. Our detector is relatively robust to noise. Last, we do not use the texture descriptor to specifically characterize a particular texture or to imply a fine differentiation between texture types. We elaborate on this in Section II.

Our approach is based on the observation that texture should appear disorganized at some scale. To obtain the

[*]Electronic address: ruth.bergman@hp.com
[†]Electronic address: hila.nachlieli@hp.com
[‡]Electronic address: gitit.ruckenstein@hp.com

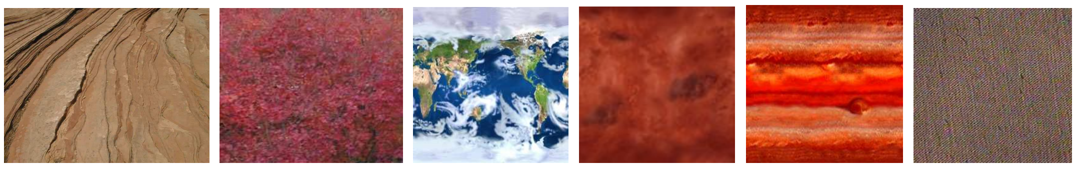


FIG. 1: Texture examples.

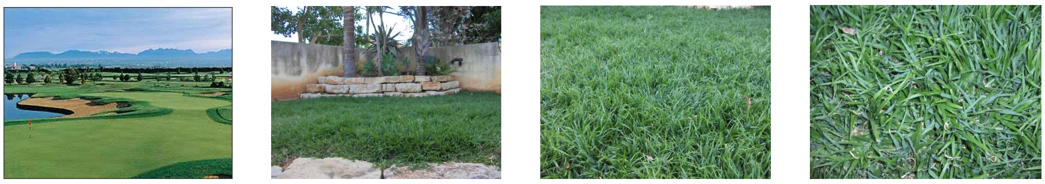


FIG. 2: Texture of grass at various scales.

underlying structure of the block, we coarsely quantize the gray-levels in the block to two levels. The number of connected components in the quantized block gives us a measure of disorganization. Similar block quantization forms the basis of the image compression technique known as Block Truncation Coding (BTC) [2, 3]. The image quality loss of BTC is acceptable. Our texture descriptor is computed by counting the components of blocks at several scales and incorporating a contrast measure. By thresholding the descriptor values, we obtain a *texture detection tool.*

The remainder of this report is organized as follows. Section II reviews the related work and covers a large number of texture descriptors. Section III describes the new texture descriptor. We conducted a texture detection experiment, in which texture patches were visually labelled as textured or smooth. We assess the ability of our detector to classify patches and compare its performance with a detector based on a common texture descriptor in Section IV. Applications are considered in Section V. We conclude in Section VI.

## II. RELATED WORK

Understanding texture is important to researchers of image enhancement and computer graphics. The two main problems in this field are texture synthesis and texture analysis. In texture synthesis a small patch of the texture is used to create a larger image with the same texture [4–6]. More advanced algorithms in computer graphics extend these solutions to texture mapping to 3-D surfaces [7–9].

More recent work on image *in-painting* or *in-filling* aims to solve the texture synthesis problem in the setting of natural images [10, 11]. This problem is more challenging than synthesis from a patch because boundaries between image components must be maintained. The approach used by [10] is to separate the image to two images: a detail image containing the texture and a "coarse" image containing the overall structure of the image. Each of these images is reconstructed separately, the coarse image using the in-painting algorithm in [12] and the texture image using a texture synthesis algorithm.

In texture analysis, the goal is to describe a given texture with some (preferably small) set of descriptors. Like texture synthesis, early work on texture analysis concentrated on texture patches, e.g., retrieval of similar textures from a texture database [13, 14]. More recent work is applying texture analysis in the setting of natural images that contain texture. Texture analysis is used for all image enhancement operations, e.g., denoising and sharpening. Image segmentation work today is concentrating on *texture segmentation*; segmenting images that contain a variety of textured areas. *Image classification*, namely the problem of categorizing the content of an image, and the related problem of *image retrieval*, finding similar images from a database of images, also use texture analysis in addition to color and other image descriptors.

Sub-section II A describes a broad spectrum of texture descriptors found in the literature for both texture analysis and synthesis. In sub-section **??**, we discuss a recent method developed in our lab for context-aware image enhancement, which includes a texture detection component [15] .

## A. Texture descriptors

Descriptors for texture are motivated by human perception or by the search for a simple model or a statistical process that generates the texture. Much of the following background is taken from Tuceryan & Jain [1].

### 1. *Statistical*

Julesz first proposed the use of second order statistics as texture features [16]. *Gray level Co-occurrence matrices* (GLCM) estimate the second order statistics of images. The number of occurrences of pairs of pixels are counted, for example a value for the central pixel and a value for the pixel to the right of the central pixel. Typically the co-occurrence statistics are stored for several pixel displacements in a neighborhood. These counts can be normalized to obtain an estimated probability distribution. Co-occurrence matrices have been used for texture analysis, e.g., classification [17, 18], and texture synthesis [19] applications.

For image analysis applications several texture descriptors are often computed from the co-occurrence matrix [1, 20]. *Maximum probability* indicates the strongest response to the position operator. *Energy* is a measure of uniformity. *Entropy* is a measure of randomness. *Correlation* is used to assess the amount of regularity as well as the fineness/coarseness of the texture. *Contrast* and *Homogeneity* are useful features as well.

### 2. *Psychophysical*

Tamura et al defined the following textural features that correspond to human visual perception [21]. *Coarseness* is related to the scale and repetition of the texture. *Contrast* captures the dynamic range of the texture. *Directionality* measures how directional an image region is.

### 3. *Structural methods*

Structural methods describe textures in terms of primitives which can be combined according to placement rules. Prior work along these lines concentrates on discovering the texture element [22, 23] or the placement rules [24, 25]. These methods, in general, are only applicable to very structured, non-random textures.

Tuceryan and Jain [26] extracted texture tokens by using properties of of the Vornoi tesselation [27]. These features were used to segment binary images.

### 4. *Model Based*

Markov Random Fields are a popular model for images. The models make the assumption that the intensity at each pixel depends only on the intensities of nearby pixels. They are, therefore, able to capture local image information.

Texture modeling methods based on Markov Random Fields estimate parameters of the probability distribution of the texture. These models have been used for texture synthesis [28, 29], texture classification [30, 31], image segmentation [32, 33], image restoration [34] and image compression. For texture synthesis these models successfully generate synthetic textures and micro-textures, but fail with irregular and structured textures.

Fractal geometry [35] is useful in image processing because it models self-similarity across scales. The *fractal dimension* gives a measure of the roughness of a surface. Large fractal dimension indicates rougher texture. Another fractal property, *lacunarity*, indicates the scale of the texture. It is small for fine texture and large for coarse texture. Algorithms for estimating these fractal properties of texture are given in [36]

### 5. *Signal processing methods*

Signal processing approaches attempt to mimic preattentive perception [37, 38]. These techniques include filtering in the spatial domain, Fourier domain, and Wavelet domain. Typically filters are applied at multiple scales [8, 39]. Wavelets, which include both spatial and frequency information, are well-suited to this task. The Gabor filter applied at multiple scales and orientations is particularly popular in the literature.

Gabor filters were used successfully for segmentation and classification of textured images [40, 41]. Multi-scale models using a large bank of filters at each level give very good texture synthesis results [6, 8, 39, 42, 43].

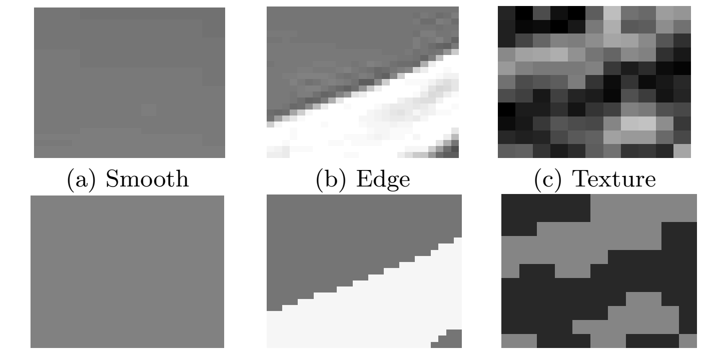


FIG. 3: Image block examples.

**ComponentCount(block)**

1. Separate the pixels to two groups according the their gray-level values.
2. Compute a representative gray level for each group.
3. Select the background group to be the larger group of pixels.
4. Construct a binary image of the block in which pixels from the background group are set to 0, and pixels from the other group are set to 1.
5. Compute the connected components in the background.

**Component count** = the number of connected components.

**Contrast** = the difference between the averages of the two groups of pixels.

FIG. 4: Algorithm for computing the Component Count and Contrast descriptors.

### *6. Patch-based*

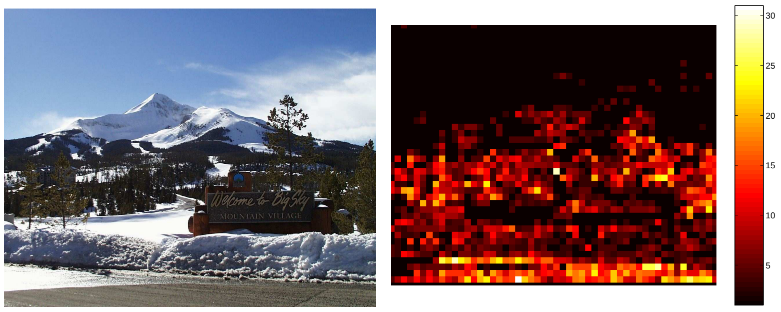
Recent work has avoided limited sets of descriptors that summarize the content of the image. Instead, patches, which are small blocks of the original textures, are the descriptors of the texture. The resulting model is large, compared with other models, and more expressive. This approach is successful for texture synthesis [5, 11], and has been used by Wolf et al. for texture edge detection and segmentation [44]. In [45], a patch-based approach is used for image denoising.

## III. THE TEXTURE DETECTION ALGORITHM

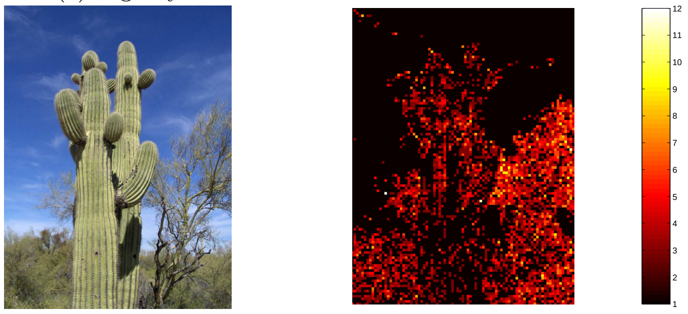
Figure 3 shows example blocks that we would perceive as smooth (a), edge (b) and textured (c). Intuitively, this discrimination can be made by counting the number of components in the block. A coarse quantization of the gray-levels in the block to two levels enables us to use connected component algorithms to automatically count the number of components in the block. Figure 3 (d)-(f) demonstrates the quantized blocks obtained from the images of Figure 3 (a)-(c) respectively.

### A. A component-count local descriptor of texture

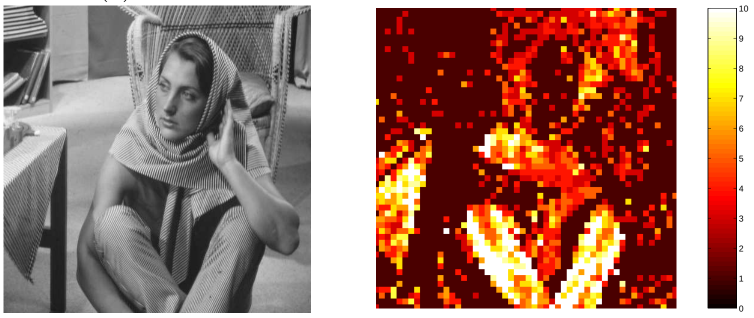
Figure 4 specifies the procedure for computing two local texture descriptors, the component-count and the contrast. There are several methods for separating pixels in a block to two groups. It is a clustering problem where the gray-levels of the pixels are the data. Any clustering algorithm can be used to separate the groups, e.g., K-means or EM [46]. We have tested a method with lower computational cost which is based on Block Truncation Coding (BTC) [2, 3]. To find the two groups, the average gray level in the block is computed. Pixels with gray levels above the average comprise the first group, and the rest comprise the second group. For compression purposes, each pixel value



(a) Big Sky



(b) Cactus



(c) Barbara

FIG. 5: Examples of the Component Count descriptor. Respective block sizes: (a) - 25 $\times$ 25, (b) - 10 $\times$ 10, (c) - 10 $\times$ 10.

should be replaced with the average of values in the corresponding group. The quantized block obtained preserves the mean and the first absolute central moment of the gray levels in the original block. This BTC variant is therefore known as Absolute Moment BTC.

Parameter choices when computing connected components (Step 5) affect the values of this measure. Applying 4-connected components gives more components than does 8-connected components. It is also common practice to compute the connected components once on the original black/white image, and again on its inverse. The sum of the resulting two numbers, namely the number of black components and white components, is a good option for the descriptor definition.

Figure 5 depicts the component count for the various blocks in several images. Note that the textured areas respond with high values, whereas the smooth areas have low value (typically 1). The large image edges, such as the mountain peak in the top image, and the table leg in the image Barbara also have low values. The cactus picture contains textures from three different foliage types. The descriptor responds to all the textures with high values.

The contrast measure is a natural by-product of the separation to two groups. It is computed as the difference between the average values of the two groups. We believe this measure is more relevant than other local contrast measures, such as the maximum minus the minimum gray levels or the standard deviation. For texture detection, the contrast measure is used to eliminate very low contrast texture which is not perceptible. It also provides some robustness in the face of noise.

```
TextureIndicator(image, blockSizeList, textureThreshold)

1.  For blockSize in blockSizeList
1a.     Partition image into blocks of size blockSize $\times$ blockSize.
1b.     For each block compute the component count and contrast measures.
2a. Convert the component count measure for each blockSize to a probability value in the range $[0, 1]$,
        let this value be $p_{blockSize}$.
2b. Convert the contrast measure for each blockSize to a probability value in the range $[0, 1]$,
        let this value be $c_{blockSize}$.

Texture Likelihood $= w_{25}c_{25}p_{25} + w_{10}c_{10}p_{10} + w_5c_5p_5$ where $w_{25} + w_{10} + w_5 = 1$.
Texture Indicator $= 1$ where the Texture Likelihood is above the textureThreshold, 0 otherwise.
```

FIG. 6: Algorithm for texture detection.

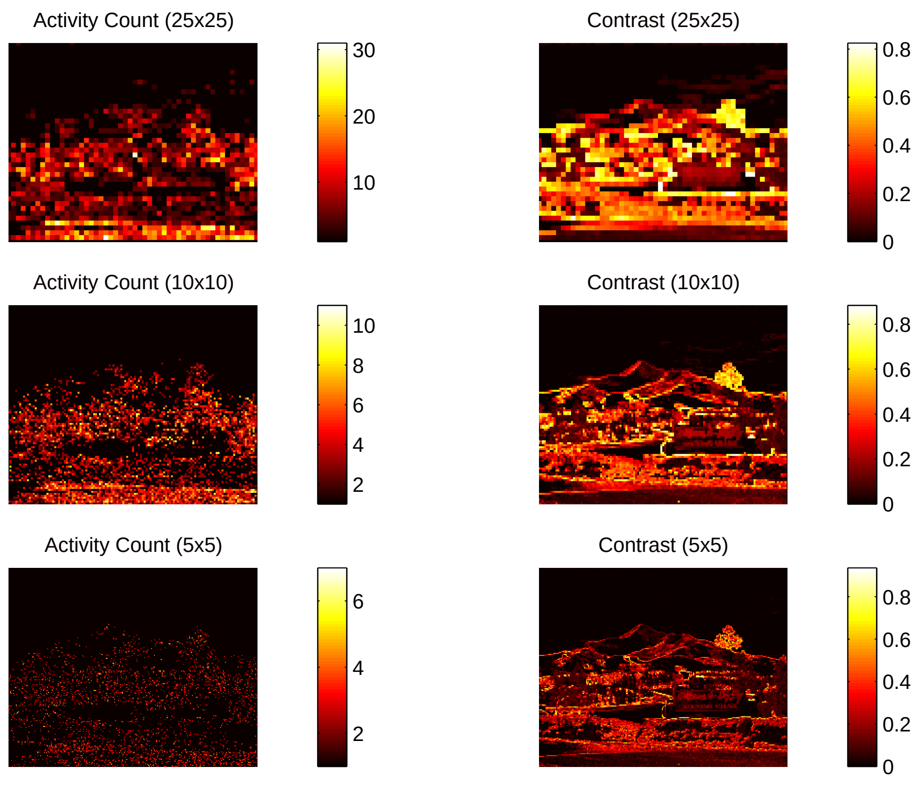


FIG. 7: Examples of Component Count and Contrast descriptors at different scales for the image Big Sky in Figure 5(a).

## B. Texture detection using the component descriptor

Figure 6 summarizes the steps required for identifying texture using the component count and the contrast descriptors. Since texture is very dependent on scale, the descriptors should be computed at several scales. The required block sizes depend on the size of the texture element which depends on the image resolution. We tested our algorithm on images around 1-1.5 mega pixels, and we used the block sizes of $25 \times 25$, $10 \times 10$ and $5 \times 5$. Figure 7 shows the values of the two descriptors, computed for the three block sizes.

Each of the two descriptors is normalized to the format of a likelihood measure, getting real values between zero and one. The likelihood measure obtained from the component count is indicative of the presence of texture, where values close to 1 give high likelihood of texture and values near 0 give low likelihood of texture. The normalization is done by dividing the number of components by some maximal integer representing a component count that definitely indicates texture (values above 1 should be clipped to 1). It may be useful to apply an additional correction function which compresses high counts and stretches low counts. For example, one can use the correction function $\sqrt{\log()}$.

The likelihood measure obtained from the contrast descriptor should indicate how much the contrast is perceptible. We clipped to zero very low contrast values representing gray level differences of a few levels. Values that represent high differences were clipped to 1. The values in between were corrected to fit a linear function.

The two likelihood measures are multiplied to accentuate those areas that have high activity and perceptible contrast. The evidence from the three different scales are combined in a weighted sum, where we typically give the most weight to the medium scale and the least weight to the lowest scale. We point out that the local computation in every block is rather fast, and the overall running time of the detection algorithm is short.

## IV. EXPERIMENTAL RESULTS

Figures 8-9 show some results of the texture detection algorithm on natural images. We have tested images with a wide variety of content. Overall, the indicator seems to capture our intuitive discrimination of texture. Smooth areas are not marked as texture, e.g., the image of Lena and Barbara. Large edges, likewise, are not marked. There are examples of this in most of the images, particularly the mountain, Lena and Barbara. The baboon and yard images show that the indicator will respond to images irrespective of how much of the image is textured. The cactus image is a good demonstration of the response of the indicator to different textures.

In the remainder of this section we describe an experiment where likelihood measures are computed by our texture detection algorithm and a detector based on a common texture descriptor, and compared to the results of an independent manual classification. Eighty one rectangular image patches are used for this test, where every patch is taken from a single image segment. The patches originate at twenty four natural images that were selected for tests of automatic image enhancement. This image collection includes various content types, and some of the images suffer from considerable noise or JPEG artifacts.

Manual classification is done while viewing each patch in the context of its image. The task is to divide the patches to the following four classes that represent descending levels of perceptual texture: texture, fine texture, smooth, and uniform. Since some patches were hard to classify, a fifth middle class was defined for ambiguous patches. An additional classification of the patches divides them to clean versus noisy (some of those contain heavy JPEG artifacts). Examples of clean patches classified as texture, fine texture and uniform are shown in Figure 11. The distribution of all eighty one patches is described in Table I.

| | Texture | Fine Texture | Ambiguous | Smooth | Uniform | Overall |
|---|---|---|---|---|---|---|
| Clean | 23 | 11 | 1 | 3 | 13 | 51 |
| Noisy | 3 | 3 | 3 | 2 | 19 | 30 |

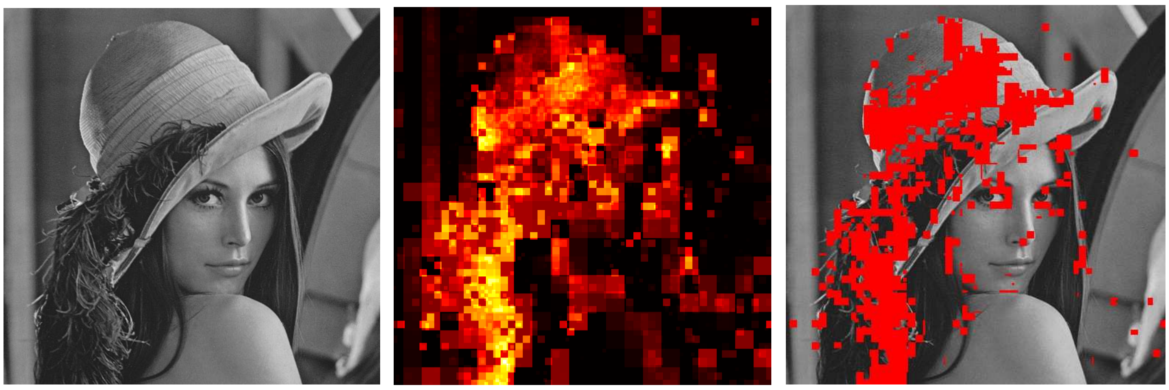
TABLE I: Manual classification results for test patches

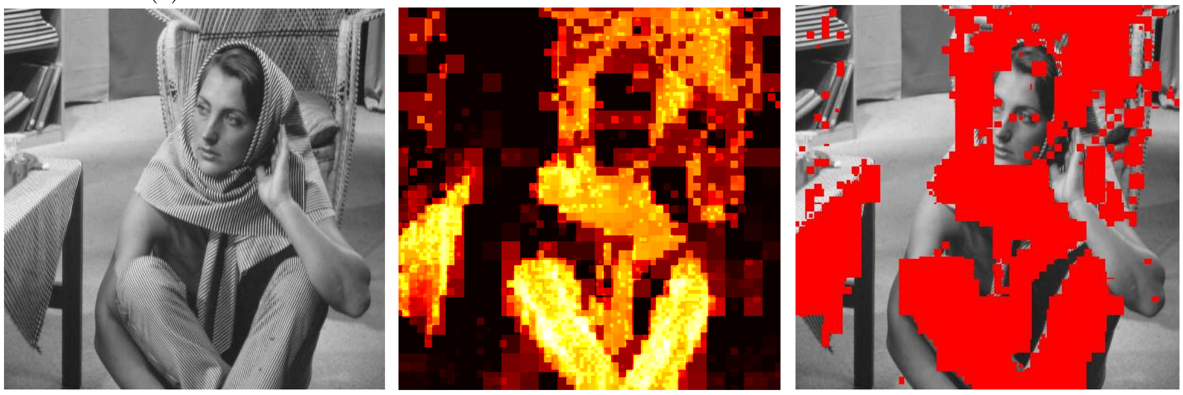
The diagrams in Figure 12(a) and (c) illustrate the comparison between the manual classification and our texture detector output. In figure 12(a), results are shown for the entire set of patches, while Figure 12(c) shows the results of non-noisy patches only. Every circle represents a single patch, where the $x$-coordinate corresponds to the manually matched class and the $y$-coordinate corresponds to the likelihood value computed by the texture detection algorithm. Specifically, the likelihood values are computed by multiplying the values of the component count and the contrast descriptors, computed in several scales, as explained in the end of subsection III B.

In both diagrams, there is an apparent separation of the likelihood values into two ranges corresponding to the classification results, where only a few values do not match their class; one range corresponds to the texture and fine texture classes, and the other range corresponds to the smooth and uniform classes. In Figure 12(a), most of the patches that mismatch their range are classified as noisy. Overall, the correspondence of the detector to the manual classification seems to validate the algorithm. Since the majority of the noisy patches pass the classification test, these results demonstrate some ability to differentiate texture from noise using local data.

As a comparison, we evaluate the ability of a commonly used texture descriptor to perform this task. We used local standard deviation (SD) which is a measure of the contrast of the texture. In our experiment, we used the standard deviation measure of $10 \times 10$ blocks. Figure 10 demonstrates the response of this descriptor to the images in Figure 8. It is clear from these images that this measure responds to the texture. However, it responds more strongly to image edges. In this sense, it does not capture our intended notion of texture in natural images. For this test, we apply it to patches with homogeneous textured or smooth content, so response to edges does not enter into the evaluation.

Figure 12(b) and (d) show the result of texture detection using the local standard deviation measure to the full set of eighty one patches and the fifty non-noisy patches, respectively. The agreement between manual classification


(a) Lena


(b) Barbara

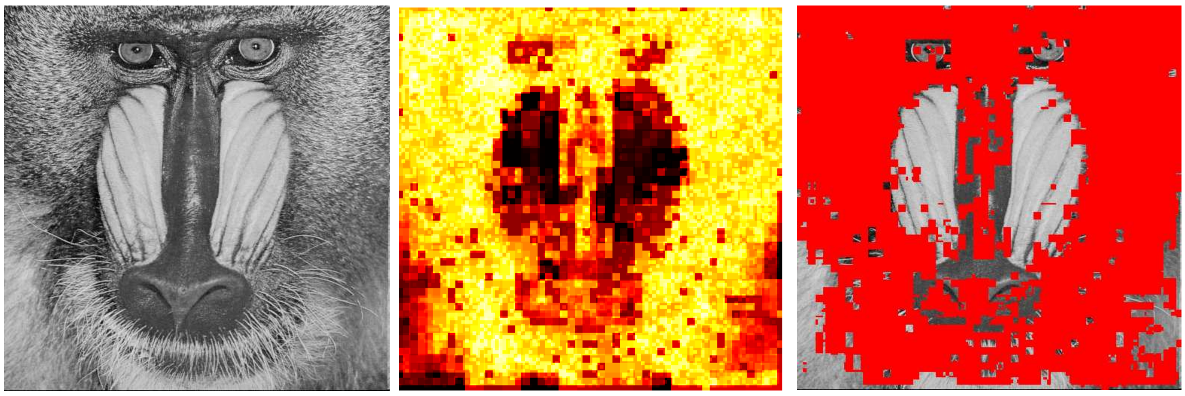
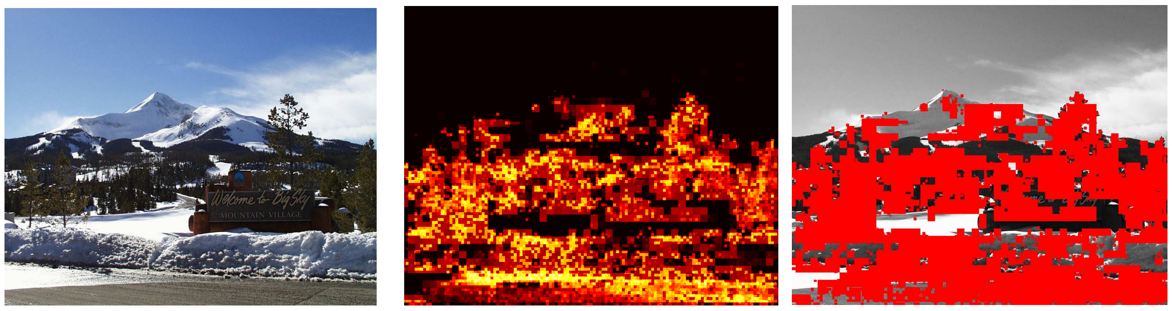

(c) Baboon

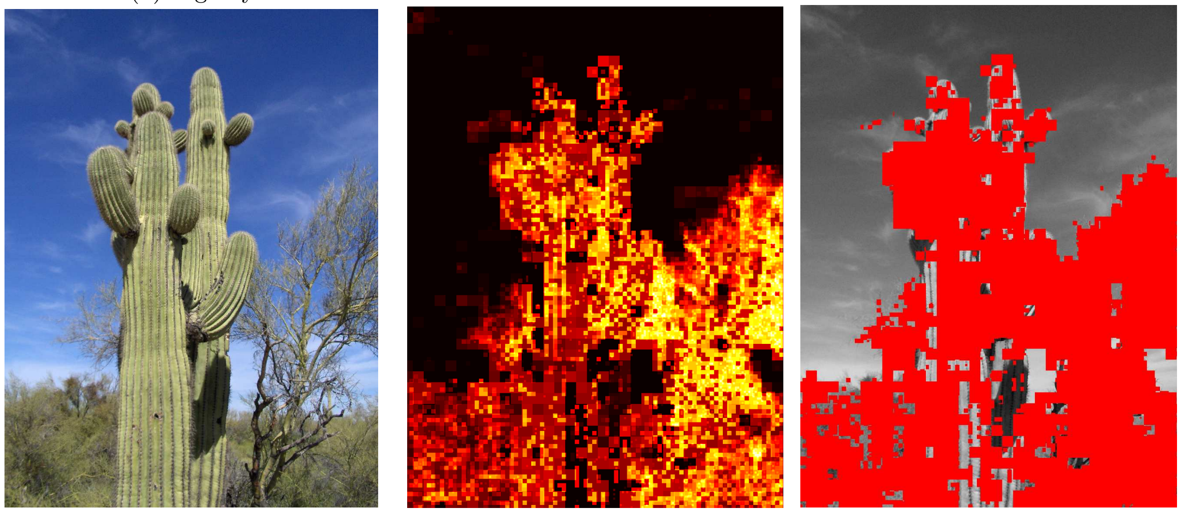
FIG. 8: Examples of texture detection. Likelihood measures were thresholded to obtain the rightmost images.

and automatic classification by our detector is better than that of the detector using the local SD descriptor. In particular, this descriptor misclassifies more uniform patches as textured. Looking at the non-noisy images, our detector mis-classifies 8% of the uniform patches, compared with 46% misclassified by the local standard deviation. Figure 13 shows two example patches that are misclassified by the contrast descriptor, but classified correctly by our algorithm. Patch (a) contains JPEG compression artifacts and patch (b) has a slight vignette. In further testing, we found that our algorithm is robust to JPEG artifacts and noise.

## V. APPLICATIONS

Since the texture detection algorithm shows a good correlation to the human perception of texture, it can be used as an image analysis measure to aid image enhancement algorithms. The enhancement of human faces in digital images and prints is particularly important, because customers have strong expectations regarding the appearance of


(d) Big Sky


(e) Cactus

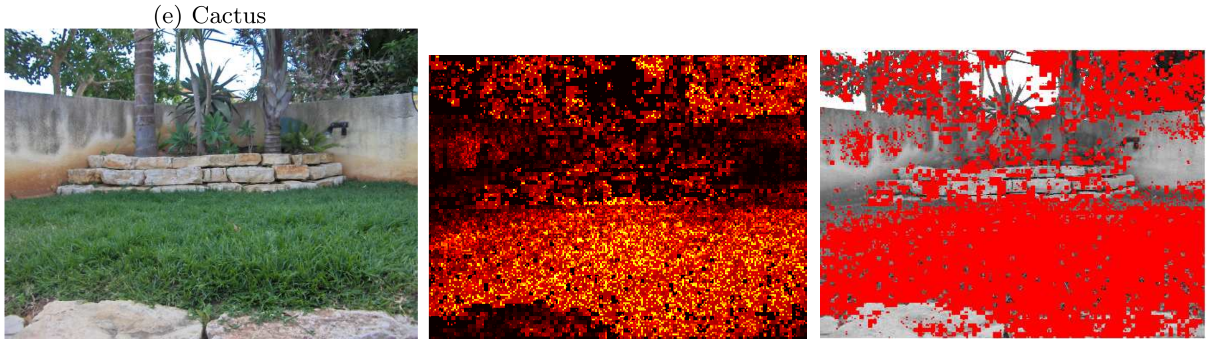

(f) Yard

FIG. 9: More examples of texture detection.

faces. Often, these expectations lead to the need for smoothing some parts of a face (as the skin), while sharpening others (like the hair, eyes and lips). Texture detection could follow a face detection scheme and further assist in tuning the local enhancement to facial features. This capability is illustrated in Figure 14 which shows a texture map corresponding to the image in Figure 11. See also Figure 8 (a) and (b).

Another application of great interest in our laboratory is the removal of dust and scratch defects from scanned images. This problem is challenging because those defects must be distinguished from image features given a single image. We have developed an algorithm to detect dust and scratch defects, as well as some algorithms for the defect reconstruction [47]. The detection algorithm has two phases, pixel-labelling and regional classification. The pixel labelling phase consists of a local algorithm that identifies small bright details. This algorithm responds to some textures, e.g., leaves, pinstripes, with a large number of false detections. If a reconstruction algorithm is applied to those false detections, the overall running time may be very long and the resulting image texture may be blurry. For this reason, the regional classification phase uses larger scale considerations to refine the labelling obtained in the first phase.

In our original dust and scratch removal algorithm [47], we avoided those false detections by identifying blocks with

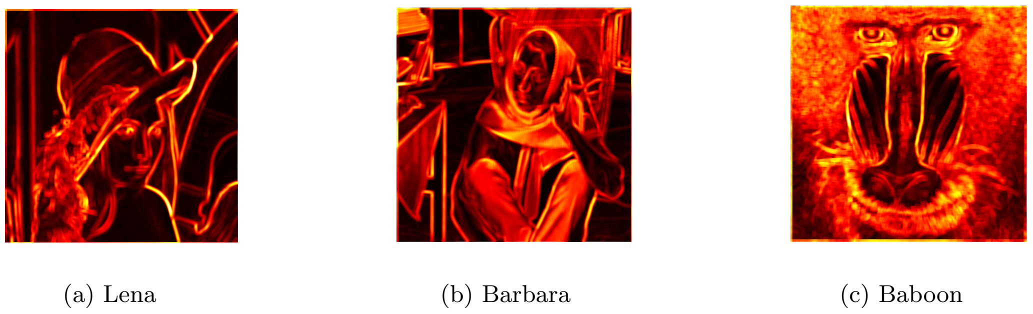

(a) Lena (b) Barbara (c) Baboon

FIG. 10: Examples of the response of the local standard deviation measure for the images from Figure 8.

a high detection rate. All the pixels in such blocks were re-labelled as non defective. The drawback of this simple solution is the difficulty in selecting a good threshold number of detections per block. On the one hand, this threshold should be low enough to ignore false detections in textured regions but, on the other hand, it should be high enough to retain detections of large scratches in smooth regions.

Based on the texture detection algorithm presented in this report, we were able to improve the dust and scratch detection results. In the regional classification phase, pixels are re-labelled as non defective if the corresponding block is identified as both textured and containing a high number of detections. Though defects are missed if they are located in textured areas, the visible and bothersome defects, that are typically located in smooth areas, are detected and removed.

## VI. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

We developed a new measure for image blocks that encapsulates the block structure in a way that enables texture identification. Textured blocks are differentiated from both smooth blocks and blocks that contain edges between smooth image segments. The detection of texture has been successfully tested on various natural images and compared to human classification of image blocks. The texture detection tool in its current format can be used in various image enhancement applications. It allows a coarse differentiation between image contents in applications where processing should be tuned to local neighborhood.

In future work, we plan to further develop our texture descriptor to allow more sophisticated differentiation between image contents. One important task is texture segmentation, where edges are drawn between image segments containing different textures. Another desired capability is texture classification where some texture classes are defined and characterized. Instances of such textures, obtained from photos in various resolutions, zoom parameters, and illumination conditions can then be correctly matched to their classes.

To achieve the above goals, we first try to characterize the *distribution* of the texture descriptor values in a specific texture region. It may be useful to combine our texture descriptor with some of the descriptors mentioned in Section II to obtain a good characterization of specific textures.

[1] M. Tuceryan and A. K. Jain, *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)* (World Scientific Publishing Co., 1998).

[2] E. J. Delp and O. R. Mitchell, IEEE Transactions on Communication **27**, 1335 (1979).

[3] P. Franti, O. Nevalainen, and T. Kaukoranta, Computer **37**, 308 (1994).

[4] A. Efros and T. Leung, in *Proceedings of IEEE Internation Conference on Computer Vision* (1999).

[5] A. Efros and F. W.T, in *Proceedings of SIGGRAPH 2001* (2001), pp. 341–346.

[6] J. S. D. Bonet, in *Proceedings of ACM SIGGRAPH* (1997).

[7] Y. Hel-Or, T. Malzbender, and D. Gelb, in *Texture 2003 - 3rd International Workshop on Texture Analysis and Synthesis* (2003).

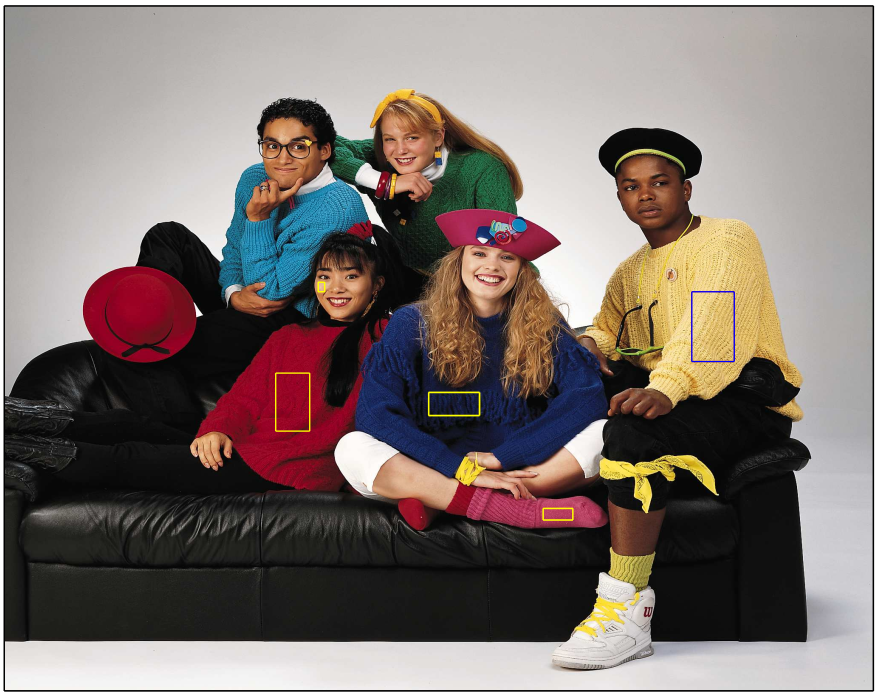
[8] D. Heeger and J. Bergen, in *Proceedings of SIGGRAPH* (1995), pp. 229–238.

[9] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick, ACM Transactions on Graphics, SIGGRAPH 2003 **22**, 277 (2003).

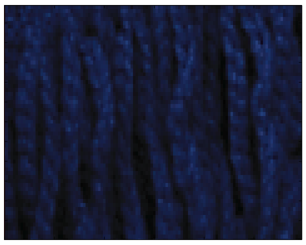
[10] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, *Simultaneous structure and texture image inpainting.*

[11] Y. Wexler, E. Shechtman, and M. Irani, in *Proceedings of Computer Vision and Pattern Recognition* (2004).

[12] V. C. M. Bertalmio, G. Sapiro and C. Ballester, in *Proceedings of SIGGRAPH 2000* (2000).
[13] M. N. Do and M. Vetterli, IEEE Transactions on Image Processing **11**, 146 (2002).
[14] J. S. de Bonet and P. Viola, in *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE Computer Society, Washington, DC, USA, 1998), p. 641, ISBN 0-8186-8497-6.
[15] P. Kisilev, Tech. Rep. HPL-2005-7, HP Laboratories (2005).
[16] B. Julesz, IRE Trans. Inf. Theory **8**, 84 (1962).
[17] R. Haralick, in *Proceedings of the IEEE* (1979), vol. 67, pp. 786–804.
[18] P. Ohanian and R. Dubes, Pattern Recognition **25**, 819 (1992).
[19] A. C. Copeland, G. Ravichandran, and M. M. Trivedi, Optical Engineering **40**, 2655 (2001).
[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing* (Addison-Wesley, 1992).
[21] H. Tamura, S. Mori, and T. Yamawaki, IEEE Trans. Systems, Man, and Cybernetcs **SMC-8(6)**, 460 (1978).
[22] H. Voorhees and T. Poggio, in *Proceedings of the first International Conference on Computer Vision* (1987), pp. 250–258.
[23] D. Blostein and N. Ahuja, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-11**, 1233 (1989).
[24] K. S. Fu, *Syntactic Pattern Recognition and Applications* (Prentice-Hall, 1982).
[25] S. W. Zucker, Computer Graphics and Image Processing **5**, 190 (1976).
[26] M. Tuceryan and A. K. Jain, *Texture segmentation using Voronoi polygons*, vol. PAMI-12 (1990).
[27] N. Ahuja, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-4**, 336 (1982).
[28] G. Cross and A. Jain, IEEE Transactions PAMI **5**, 25 (1983).
[29] L.-Y. Wei and M. Levoy, in *Proceedings of SIGGRAPH* (2000).
[30] R. Chellapa and S. Chetterjee, IEEE Transactions on Acoustic, Speech, and Signal Processing **ASSP-33**, 959 (1987).
[31] A. Khotanzad and R. Kashyap, IEEE Transactions on Systems, Man, and Cybernetics **17**, 1087 (1987).
[32] F. S. Cohen and D. B. Cooper, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-9**, 192 (1987).
[33] C. W. Therrien, Computer Vision, Graphics, and Image Processing **22**, 313 (1983).
[34] S. Geman and D. Geman, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-6**, 721 (1984).
[35] B. B. Mandelbrot, *The Fractal Geometry of Nature* (Freeman, 1983).
[36] R. Voss, *Random fractals: Characterization and measurement* (Plenum, 1986).
[37] J. R. Bergen, *Theories of visual texture perception*, vol. 10B (Macmillian, New York, 1991).
[38] J. Malik and P. Perona, Journal of the Optical Society of America **Series A, 7**, 923 (1990).
[39] J. S. D. Bonet and P. Viola, *A non-parametric multi-scale statistical model for natural images* (1997).
[40] A. K. Jain and F. Farroknia, Pattern Recognition **24**, 1167 (1991).
[41] M. R. Turner, Biological Cybernetics **55**, 71 (1986).
[42] S. C. Zhu, Y. Wu, and D. Mumford, Int. J. Comput. Vision **27**, 107 (1998), ISSN 0920-5691.
[43] J. Portilla and E. Simoncelli, International Journal of Computer Vision **40**, 49 (2000).
[44] L. Wolf, X. Huang, I. Martin, and D. Metaxas, in *submission* (2005).
[45] A. Buades, B. Coll, and J. Morel, *A review of image denoising algorithms, with a new one.*
[46] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction* (Springer-Velag, 2001).
[47] R. Bergman, R. Maurer, H. Nachlieli, G. Ruckenstein, P. Chase, and D. Greig, submitted to Journal of Electronic Imaging (2007).

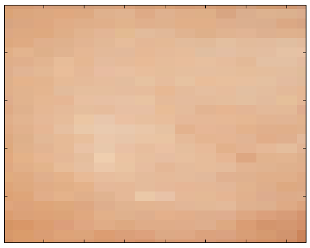


(a) red sweater - texture

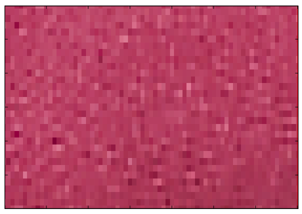


(b) blue sweater - fine texture

(c) yellow sweater - texture

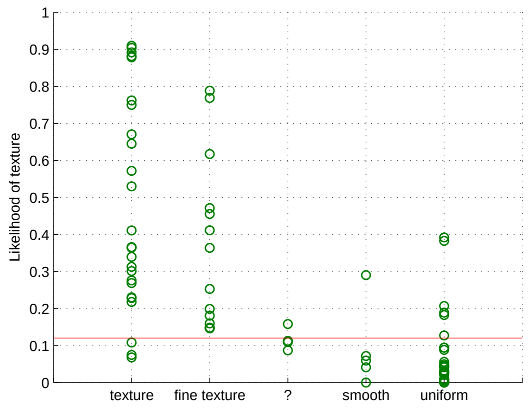


(d) face - uniform

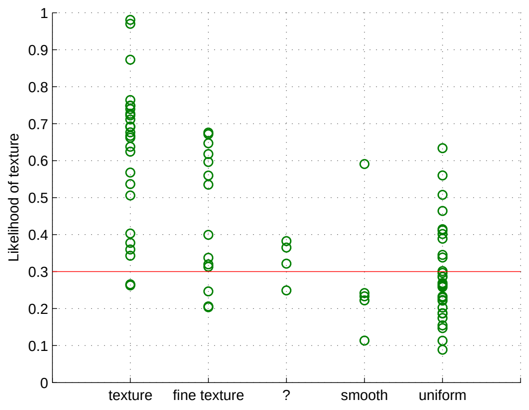


(e) pink sock - fine texture

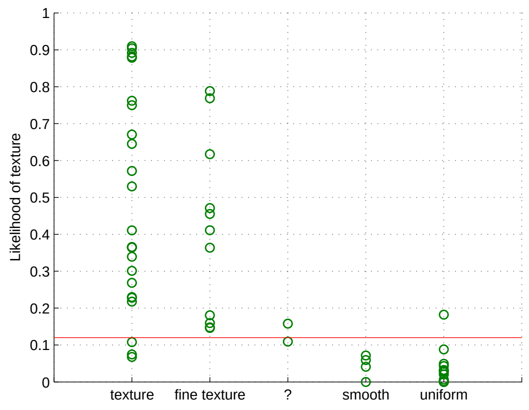
FIG. 11: Example of clean image patches for texture classification test.



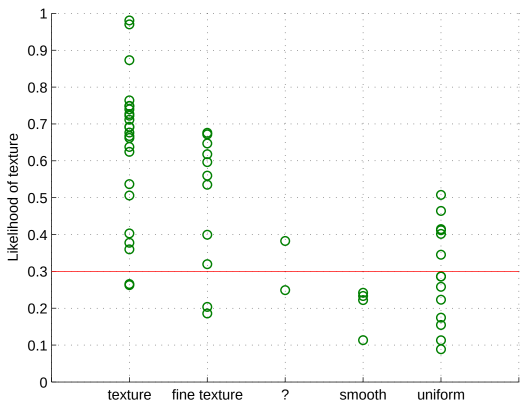
(a) Classification using the component count measure



(b) Classification using the local SD measure

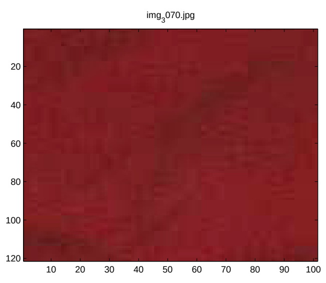


(c) Classification of non-noisy patches using the component count measure

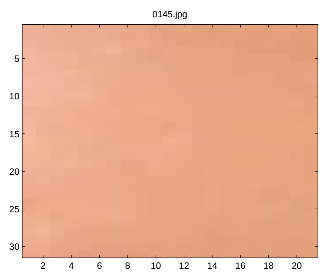


(d) Classification of non-noisy patches using the local SD measure

FIG. 12: Results of texture classification test.



(a) Patch with JPEG blocking artifacts



(b) Patch with vignette

FIG. 13: Example of image patches that are mis-classified as texture by the contrast texture measure.

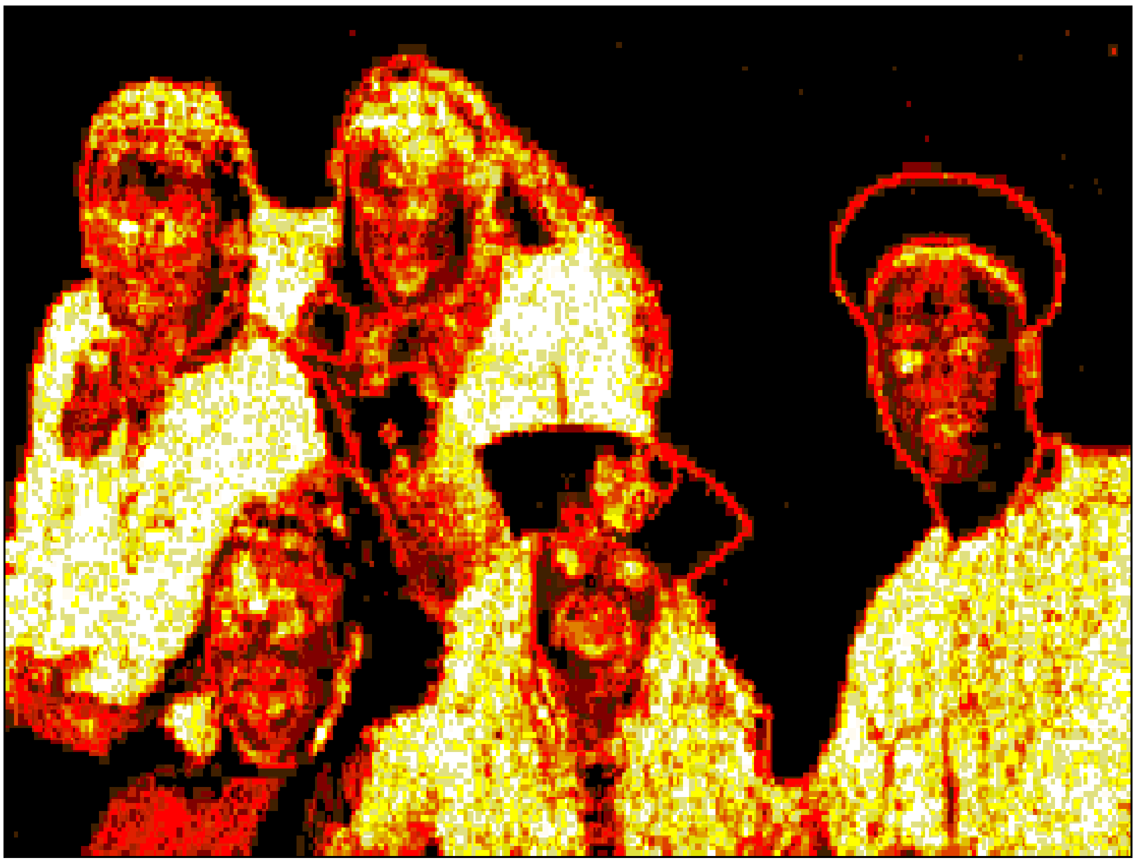


FIG. 14: Texture detection map for parts of the image in Figure 11. Bright colors represent high likelihood of texture evidence.