Pattern Recognition Letters 34 (2013) 2071–2078

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Scale detection via keypoint density maps in regular or near-regular textures

Edoardo Ardizzone [1], Alessandro Bruno [2], Giuseppe Mazzola *

Dipartimento di Ingegneria Chimica Gestionale Informatica e Meccanica, Università degli Studi di Palermo, Viale delle Scienze bd. 6, Palermo 90128, Italy

ARTICLE INFO

Article history:
Received 13 February 2012
Available online 4 July 2013

Communicated by J. Laaksonen

Keywords:
SIFT
SURF
Harris corner
Texture Scale
Texel
Texton

ABSTRACT

In this paper we propose a new method to detect the global scale of images with regular, near regular, or homogenous textures. We define texture "scale" as the size of the basic elements (texels or textons) that most frequently occur into the image. We study the distribution of the interest points into the image, at different scale, by using our Keypoint Density Maps (KDMs) tool. A "mode" vector is built computing the most frequent values (modes) of the KDMs, at different scales. We observed that the mode vector is quasi linear with the scale. The mode vector is properly subsampled, depending on the scale of observation, and compared with a linear model. Texture scale is estimated as the one which minimizes an error function between the related subsampled vector and the linear model. Results, compared with a state of the art method, are very encouraging.

1. Introduction

Texture indicates visual patterns in real and synthetic scenes. Due to complexity and diversity of natural images, texture analysis, description and synthesis are very challenging tasks.

Structural texture analysis focuses primarily on identifying position, size and placement rules of basic texture elements, known in literature as texels or textons. Although these terms have not a precise mathematical definition, Julesz (1981) defined texton as the "putative unit of pre-attentive human texture perception". In the following of this paper we accept the common definition of a texel as a basic repetitive element of a texture pattern, and use the words "texel" and "texton" as synonyms.

In terms of periodicity and regularity of their structure, textures may be classified in five classes: regular, near regular, irregular, near stochastic and stochastic.

- Regular textures are simply periodic patterns where the intensity of color and the shape of all texture elements are repeated in equal intervals;
- Near regular textures are a statistical distortion of a regular pattern (Liu et al., 2004a);
- Irregular textures present deformation fields from regular patterns;
- Near stochastic and stochastic textures show typically dots and shapes randomly scattered over all the image.

Many scenes in real world do not have a regular structure but a near regular one (buildings, wallpapers, floors, tiles, windows, fabric, pottery and decorative arts, animal fur, gait patterns, feathers, leaves, waves of the ocean, patterns of sand dunes), so in the following of this discussion we limit our attention to regular and near regular structures. It is important to underline (Forsyth and Ponce, 2003) that what we call a texture depends on the scale at which we are viewing the scene. In other words, the same texture (e.g. grass) can look significantly different at different scales. Thus for natural scenes, also in the case of near regular textures, one of the most challenging problem of texture analysis is to detect the scale. For example, as stated by Julesz (1981), "we are able to immediately distinguish several characteristics of images of these textons, such as orientation and *size*, and use them to distinguish different textures". So the texture scale, intended as the texel size value, may be considered a *per se* important feature, that may be used in many image processing and early vision applications, like image segmentation, texture description and discrimination, defect detection, content-based image retrieval and texture synthesis. In many cases the best descriptor is not just one that most accurately describes the whole texture content, but the one that best discriminates

* Corresponding author. Tel.: +39 091/23842513; fax: +39 091/6598043.
E-mail addresses: edoardo.ardizzone@unipa.it (E. Ardizzone), alessandro.bruno15@unipa.it (A. Bruno), giuseppe.mazzola@unipa.it (G. Mazzola).

[1] Tel.: +39 091/23862610; fax: +39 091/6598043.

[2] Tel.: +39 091/23842513; fax: +39 091/6598043.

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.06.018

different classes of texture, so that an accurate segmentation or classification can be performed. Moreover, the knowledge of the texture scale may lead to a better understanding of the texel sizes. In this paper we propose a novel approach to detect texture scale, in case of regular and near regular structures, by means of a new tool, the Keypoint Density Map. The rest of the paper is organized as follows: Section 2 discusses some state-of-the-art techniques; Section 3 briefly describes the algorithms we used to extract interest points from an image; Section 4 introduces the concept of Keypoint Density Map; Section 5 presents our scale detection method; in Section 6 experimental results are discussed and compared to a state-of- the art-method; a conclusive section ends the paper. In Appendix A we illustrate the method that we use to show visual results (as texels), once the scale has been detected from the input image.

2. State-of-the-art

The study of texture regularity is not a young research field in image processing, but today it remains one of the most challenging issues in Computer Vision. Many works dealt with the problem of extracting the shape of the basic elements of the texture (texels). In Malik et al. (1999) textons are discovered as frequently co-occurring combinations of oriented linear filter outputs. They built a universal texton vocabulary by processing a large number of natural images. Each pixel of the image is mapped to its nearest texton by using a K-means clustering approach. A similar approach was presented by Zhu et al. (2005) who adopted a generative image model, in which an image is a superposition of bases from an over-complete dictionary of textons.

Some other approaches studied the whole structure of a textured image, instead of searching for its basic element. In Ahuja and Todorovic (2007) a textured image is represented, by a multi-scale analysis, as a segmentation tree, whose structure captures the recursive embedding of regions. Hays et al. (2006) presented an approach for studying regularity in near-regular textures, by discovering the underlying lattice structure of the texture patterns. Lattice finding is formulated as a higher-order feature matching problem, which is resolved with an iterative approach. Liu et al. (2004b), proposed a set of algorithms that, analyzing a given periodic pattern, is able to automatically find its underlying lattice, to identify its symmetry group, and to extract its representative motifs. Park et al. (2009) proposed a Mean Shift Belief Propagation algorithm within a MRF graphical framework for the automatic detection of deformed 2D wallpaper patterns in real images.

Many works focused on the estimation of the local or the global scale of textured images, for many purposes, without explicitly extracting the basic element or the structure of the texture. Lindeberg (1998) investigated on the scale for texture description, suggesting that the properties of a texture strongly depends on the scale of observation. Leu (2001) and Jan and Hsueh (1998) proposed structural approaches to extract periodicity from a texture, based on autocorrelation functions. Both methods have been widely used but they achieved poor results in case of natural textures. Elder and Zucker (1998) and Gomez et al. (2000), proposed probabilistic approaches for estimating the local scales in an image to detect edges and segment the image. They showed that the knowledge of the local scale can be useful for edge detection, adaptive Gaussian filtering and texture segmentation. Several works proposed approach to estimate the local scale of textured images for segmentation purposes, by using total variation flow (Brox and Weickert, 2006), energy minimization model (Hong et al., 2008), and Gabor filtering (Fernandez-Valdivia et al., 1998). Ngan and Pang (2009) proposed a supervised approach to study the regularity of patterned textures for defect detection in fabric samples. Chang et al. (2009) proposed a method to segment images by extracting explicit local features (contrast, bias, scale, and orientation) from textured regions. This method uses the steerable pyramid of Simoncelli and Freeman and imposes smoothness on these attributes via Markov random fields.

Some works focused on the estimation of the global scale of a texture, that is also the primary goal of our work. Grigorescu and Petkov (2003) identified texture scale in regular textures by searching for the smallest window through which the minimum number of different visual patterns is observed, i.e. the size of the smallest window that minimizes the Rényi's generalized entropy. This method obtains fair results with periodic textures, but it does not work as well with near-regular ones. Lizarraga et al. (2011) detected the scale of texture by analyzing the entropy of the histograms of the pixel differences along different orientations.

3. Keypoint extraction

Our approach to texture scale detection is based on the analysis of the distribution of the interest points of an image. We explored three different types of algorithms for extracting and describing interest points: SIFT, SURF and Harris corner detector.

3.1. SIFT

SIFT (Scale Invariant Feature Transform) descriptors (Lowe, 2004) are generated by finding interesting local keypoints, in a greyscale image, by detecting the maxima and the minima of a Difference-of-Gaussian function in the scale-space pyramid. The SIFT algorithm produces different levels (octaves) of Gaussian blur on the input image, and computes the difference between the neighboring octaves. Information about orientation is then computed for each keypoint, and for each scale. Briefly, a SIFT descriptor is a 128-dimensional vector, which is computed by combining the orientation histograms of locations closely surrounding the keypoint in scale-space. The most important advantage of SIFT descriptors is that they are invariant to scale and rotation, and relatively robust to perspective changes. SIFT are widely used for many computer vision application: image registration, mosaicing, object recognition and tracking, etc. Their main drawback is the relatively high dimensionality, and the high computation time, which make them less suitable for nearest neighbor lookups against a training dataset.

3.2. SURF

The SURF (Speeded Up Robust Feature) (Bay et al., 2008) is a an interest point detector and descriptor, which has been designed to be robust to scaling and rotation, and to be faster than earlier methods. The speed gain is achieved by taking profit of integral images and a fast nonmaximum suppression algorithm. It is based on three major steps: interest points extraction, repeatable angle computation and descriptor computation. Interest points are detected by using the Fast-Hessian Detector. The determinant of the Hessian matrix provides the location and the scale of the points. Invariance to image rotation, is achieved by using the Haar wavelet responses in horizontal and vertical direction. The maximum angle of the gradients surrounding the interest point is chosen as the direction of the feature. Finally, a square region is generated around each interest point, aligned to the selected orientation and split into in 4×4 sub-regions. In each sub-region Haar wavelets are extracted at regularly spaced sample points. Wavelet responses in horizontal and vertical directions are summed up over each sub region and the resulting SURF descriptor vector is of length 64.

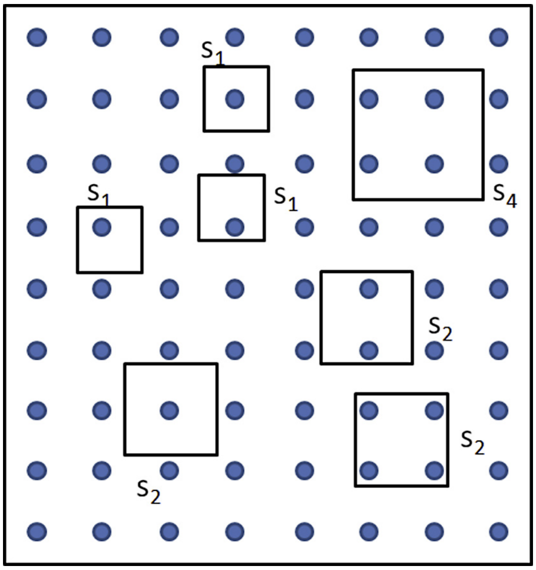


Fig. 1. An image with a very regular distribution of keypoints (blue dots). A window of size $S_1 \times S_1$ includes exactly 1 point regardless of its position. A window of size $S_2 \times S_2$, where $S_2 = \sqrt{2} \cdot S_1$, may include 1 or 2 or 4 points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.3. Harris corner detector

The Harris corner detector (Harris and Stephens, 1988) is a very simple and popular algorithm used to extract the interest points of an image. It is mainly based on the principle that, at a corner position, the image intensity has large variations along multiple directions. The algorithm works by shifting a local window onto the pixels of the image and computing the autocorrelation of the window with patches shifted by a small amount in different directions. Corner points are those around which the image intensity has its greatest variations. The Harris detector uses the trace and the determinant of the second moment matrix as the basis of its corner decisions.

3.4. Keypoint Density Map

In this section we introduce the concept of Keypoint Density Map (KDM), which is the evolution of the SIFT Density Map that we proposed in our previous work (Ardizzone et al. 2011). A KDM is a representation of the density of a distribution of the keypoints in an image and can give essential information about the regularity of its structure.

Let us consider a $M \times N$ image I, with an extremely regular distribution of keypoints (Fig. 1). The average number of pixels per keypoint, N_P, is:

$$N_P = \frac{M \cdot N}{n}. \tag{1}$$

where n is the number of image keypoints. Given

$$s_1 = \sqrt{N_P} \tag{2}$$

a squared area of size $s_1 \times s_1$ will include only one keypoint, regardless of its position within the image (Fig. 1). In general, a squared area of size $s_k \times s_k$, where

$$s_k = \sqrt{k \cdot N_P} \tag{3}$$

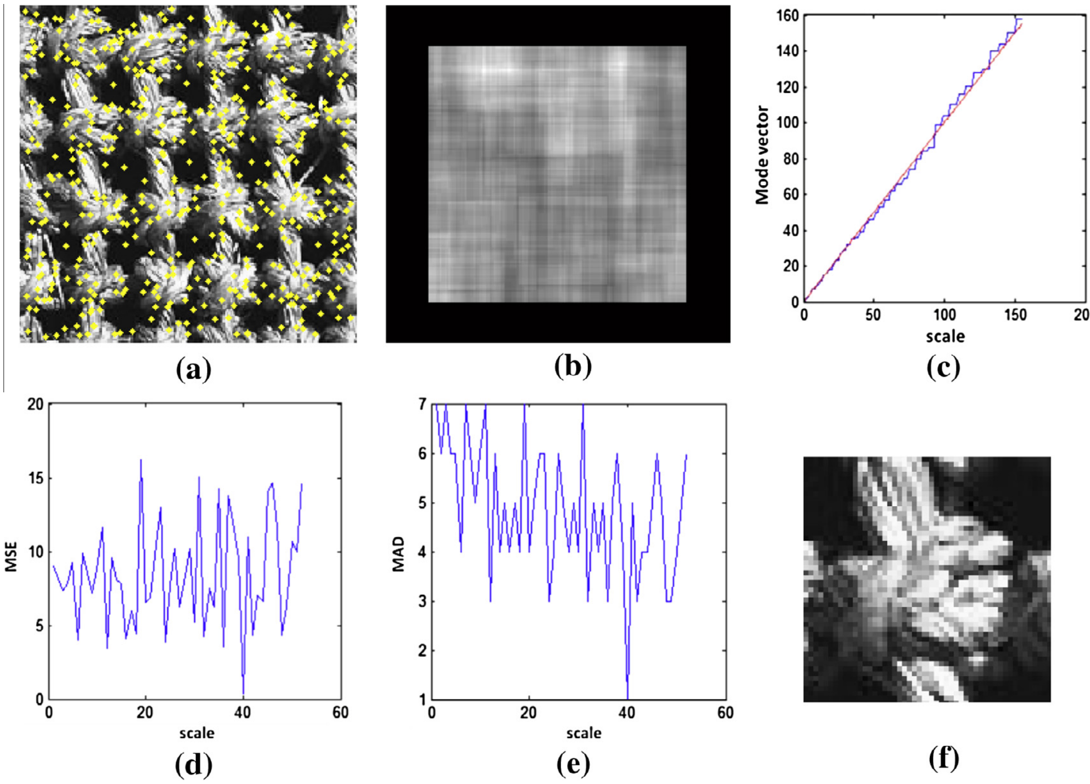


Fig. 2. An example of regular texture (a), the corresponding Keypoint Density Map (SIFT) with $k = 39$ (related to the estimated scale $s_k = 38$) (b), the Mode Vector (c), the *MSE* function error (d), the *MAD* function error (e) a magnified area at the estimated scale value (f). Note that the black area on (b) depends on the size of the sliding window, as it cannot overstep the image borders.

(as a matter of facts, all the values s_k are rounded to their nearest integer values) *on the average* will include k keypoints Due to construction, actually the window will include exactly k keypoints only when $\sqrt{k}$ is integer. Otherwise the number of keypoints included in the window will depend on its position (e.g. a window of size s_2, that is the square root of $2N_p$, may include 1, 2 or 4 keypoints of the image, as shown in Fig. 1).

A Keypoint Density Map KDM^k (Fig. 2b) is built by counting the number of keypoints into a sliding window of size s_k. Each point in the KDM^k indicates the number of keypoints falling into a squared area of size s_k, centered in the corresponding point of the image. We further define the KDM mode vector as the vector MV of the dominant modes m^k, $k = 1\ldots k_{\max}$, where m^k is the most frequent value in the map KDM^k (Fig. 2c). $k_{\max}$ is limited by the image size:

$$\begin{aligned} k_{\max} &= \tfrac{1}{N_P}\cdot s^2_{k_{\max}} \\ S_{k_{\max}} &= \tfrac{1}{2}\min(M,N) \end{aligned}. \qquad (4)$$

Due to rounding of s_k values, there is no 1–1 relationship between k and KDM^k, because the same value of s_k (and then the same KDM) could derive from different values of k. This reduces, in practice, the number of maps to be computed. In case of very regular textures, the KDM mode vector is linear, or quasi-linear, with k (for construction, linearity is assured only when $\sqrt{k}$ is integer). The actual progression of the mode vector will be used in the next section to estimate the image texture scale.

4. Texture scale detection

In Section 3 we showed that for a very regular distribution of keypoints (Fig. 1) the KDM mode vector is quasi-linear with the parameter k. For real images this assumption is not verified for all the possible values of k. To estimate the scale we look for the subset of values of k which better fits the linear assumption for the mode vector. In particular, we subsample the mode vector MV:

$$\begin{aligned} &M(k_i) = MV(n\cdot k_i) \\ &n = 1,2,3\ldots n_{\max} \\ &k_i\cdot n_{\max} \leqslant k_{\max} \\ &k_i = 1,2\ldots k_{\max}/p_{\min} \end{aligned}. \qquad (5)$$

where

- k_i is the index related to the observed scale
- $M(k_i)$ is a vector containing a subset of the values of MV starting from k_i, taken in arithmetic progression with step k_i, until the end of the vector
- $k_{\max}$ is defined in eq. 4
- $p_{\min}$ represents the minimum number of samples allowing the computation of the error function that will be used in the next step (see Eq. (10) below).

When increasing k_i, the size of $M(k_i)$ decreases, as the number of n values that satisfy the constraints of Eq. (5) decreases. To ensure that $M(k_i)$ will contain at least $p_{\min}$ values, the starting k_i is limited to the ratio of $k_{\max}$ and $p_{\min}$. This also limits the maximum scale observable with our method:

$$S_{\max} = \sqrt{\frac{k_{\max}\cdot N_P}{p_{\min}}}. \qquad (6)$$

A more accurate analysis should require terms taken in quadratic progression (see Fig. 1), that is:

$$n = 1,4,9,16\ldots n_{\max} \qquad (7)$$

but this would excessively limit the maximum observable scale, as we need to work with at least $p_{\min}$ points:

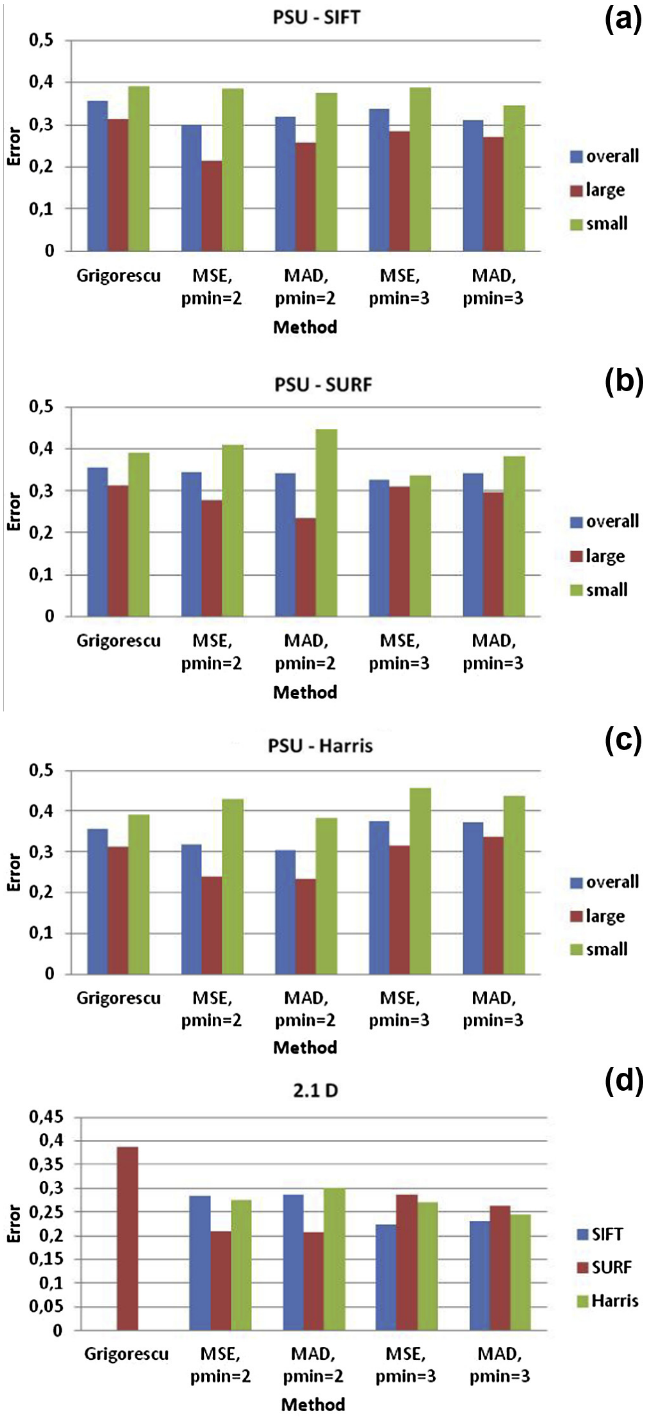
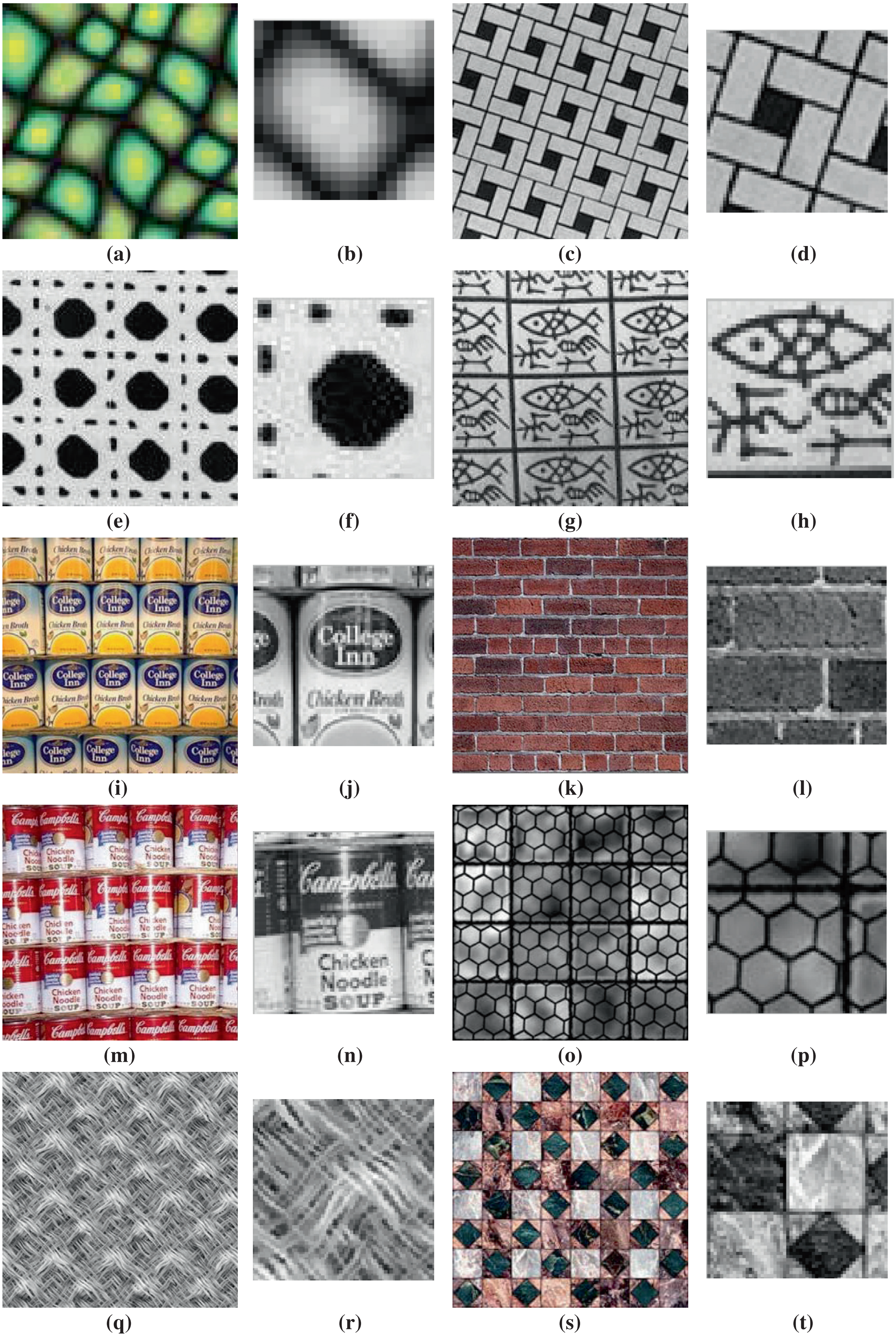


Fig. 3. The relative absolute error, with respect of the ground truth, for the PSU (first three rows) and 2.1D (last row) datasets. Values are measured for reference (Grigorescu) and our method, with different parameters and different keypoint extraction algorithms.

$$S_{\max} = \sqrt{\frac{k_{\max}\cdot N_P}{p^2_{\min}}}. \qquad (8)$$

Therefore we decided to subsample the MV vector in arithmetic progression, although this may introduce some inaccuracy. The estimation of the texture scale is accomplished by selecting the subsampled vector M which minimizes an error function e, with respect to the linear model. The corresponding starting value k_s gives the estimated scale:



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

Fig. 4. Some visual examples of our results (original-texel) within the PSU dataset.

$$\begin{aligned} S_{k_s} &= \sqrt{k_s \cdot N_P} \\ k_s &= \arg\ \min\ e(M(k_i), M_l(k_i)) \end{aligned}. \qquad (9)$$

We evaluated two different error functions: the mean square error *MSE* (Fig. 2d) and the maximum absolute displacement *MAD* (Fig. 2e)

$$\begin{aligned} (MSE(k_i)) &= \sum_n (M_n(k_i) - n \cdot k_i)^2 \\ MAD(k_i) &= \max_n |M_n(k_i) - n \cdot k_i| \end{aligned} \qquad (10)$$

with the same constraints of Eq. (4). Experimental results obtained with both of them will be discussed in Section 5.

5. Experimental results and discussion

In this section we compare our results with those of the method presented by Grigorescu and Petkov (2003). Tests were made onto two different datasets: the PSU Near Regular Texture Dataset (PSU Dataset) which is composed of 45 images with regular and near regular structures, and 80 images from the 2.1D Textures Dataset (UIUC Dataset) which is composed of natural homogeneous textures. We create our ground-truth by manually selecting the scale for all the images in the datasets. Tests were executed on an Intel Core i7 PC (4 CPU, 1.6 GHz per processor, 4 GB RAM), exploiting the Matlab parallel library to make 4 workers run simultaneously. The two error functions described in Eq. (10), with two different values of the minimum number of points, and the three types of interest points described in Section 3 were used. We split the PSU dataset into "small" (scale $\leqslant 55$) and "large" (scale > 55) textures, to show the influence of the input parameters in the two cases (we arbitrarily selected 55 as threshold to split the dataset into two subsets with more or less the same number of images). To evaluate results, we measured the average relative absolute error:

$$E = \frac{|S_k - S_{GT}|}{S_{GT}} \qquad (11)$$

of our method and of the reference method, with respect to the ground-truth s_{GT}.

Fig. 3(a–c) shows the results within the PSU dataset. In case of large textures, in almost all the tests our method performs better than Grigorescu's method, but in some cases (i.e. using the SIFT keypoints, MSE error function and $p_{\min} = 2$) our method outperforms it (21% vs 31%). In case of small textures, the SURF and the SIFT based approaches achieve more or less similar results (38%) than Grigorescu, but better results are observed when using SURF, MSE and $p_{\min} = 3$ (33%). Harris based has the worst results. As expected, results with large textures improve if we use $p_{\min} = 2$, because larger scales can be observed. Otherwise results improve for smaller textures in case of $p_{\min} = 3$, as the accuracy increases (a larger number of points to compute the error function), while the maximum detectable scale is reduced.

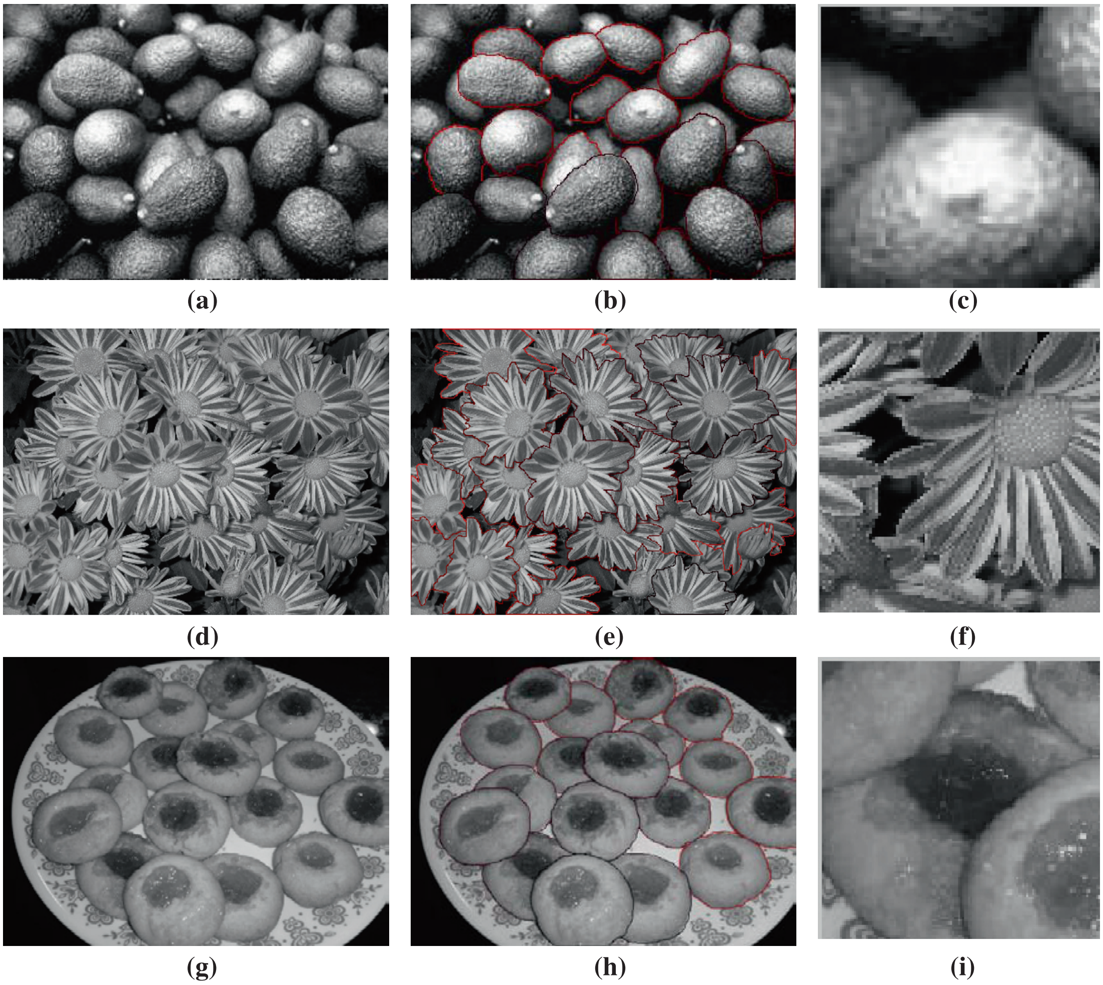


Fig. 5. Some visual examples, from dataset 2.1D. Original image (left), segmented image with (Ahuja and Todorovic, 2007) (center) and texel extracted with our method (right). Ahuja's results are shown as reference groundtruth.

Within the 2.1D Texture dataset (Fig. 3d) we achieved very satisfactory results, similar to those obtained within the PSU Dataset for larger scales, and remarkable if compared to those obtained with Gregorescu's method. In this case we do not split the dataset, as almost all the images show regularity at large scales. To show visual results, instead of manually selecting a squared area corresponding to the detected scale, or randomly choosing one, we used the method that is described in Appendix A.

Results show also that our method works as well with regular and near regular textures (Figs. 4 and 5), while Grigorescu's is optimized only for regular ones. Visual examples in Fig. 4 confirm that our methods gives very good results, in terms of estimated scale, both in case of very regular (c,e,g,q) and near regular (a,i,k,m,o,s) textures. In some cases textures may show regularity at different scales, e.g. Fig. 4o in which larger squares include smaller hexagons. In these case our method is able to find correctly the larger scale, as the keypoints are more regularly distributed along the lines of the square than along the lines of the hexagons. This can be explained as keypoint detectors typically extract more points from the stronger edges of the image (the lines of the square, in this case) than that in the thinner ones (the hexagons). Visual examples in Fig. 5 show that our method can be applied also for natural images which have homogenous textured structures. Also in this dataset some images (see Fig. 5g, the cookies and the jam core) may show regularities at different scales. Our method estimates the global scale of the texture that is related to the structure which presents a more regular distribution of keypoints (in this case the edge of the cookie rather than the edge of the jam). In Fig. 6 we show these two examples, and the superimposed extracted SIFT points.

In terms of efficiency, execution time strongly depends on the texture scale, as in case of lower scales our method needs to build more KDMs. In fact while Grigorescu's method investigates all the possible scales from a minimum value to the maximum value (a quarter of the image size), our method analyzes only a subset of the possible scales, according to the rule in Eq. (3) (in our tests, 50% of all the possible scales investigated by Grigorescu's method), therefore saving a lot of computational effort. Grigorescu's method execution time is independent of the texture scale, depending only of the image size. Furthermore, with respect to our previous version, Ardizzone et al. (2011), we improved in terms of efficiency our implementation of the map building algorithm. For a given scale, instead of shifting a window along the pixels of the image and counting the number of keypoints it includes, we work considering the keypoints. For each keypoint we update simultaneously the values of all the windows that will include it, drastically reducing the execution time. Finally, our method is two orders of magnitude faster than the reference one: few tenths of a second versus half a minute for the smallest sizes of the images in the datasets, few seconds versus a couple of minutes for medium sized images, ten seconds versus ten minutes for the largest ones (excluding the time for extracting keypoints that is, in the worst case (SIFT), about 2 s). Moreover experiments showed that the execution time our method does not vary significantly with the number of extracted keypoints.

6. Conclusions

Detecting scale in textured images is a very hard task, and a relatively unexplored problem. In our work we studied the distribution of the interest points in the image, by means of the Keypoint Density Maps, a novel instrument for Image Analysis applications. Our results are very encouraging with respect to the reference method. Experiments showed that our method achieves similar precision in case of small textures, but outperforms it in cases of large textures and in case of near-regular textures (Fig. 3). Furthermore, Keypoint Density Maps (with SIFT) have been yet successfully used in visual saliency detection, and may be a versatile instrument for several image analysis application: texture discrimination or description, image segmentation, etc.

Appendix A:. Texel visualization

In this appendix we describe the method that we use to extract the texel, in order to show visual results, once the scale has been detected from the input image. If s_k is the estimated scale, we consider the Keypoint Density Map KDM^k corresponding to the related index k, and search for all the values in the map which are equal to the most frequent value $MV(k)$. Each value in the map KDM^k corresponds to a block $B_i(k)$, of size s_k, which includes exactly $MV(k)$ keypoints. Coordinates of the central points of blocks $B_i(k)$ are then grouped by Mean Shift Clustering. For each cluster C we select

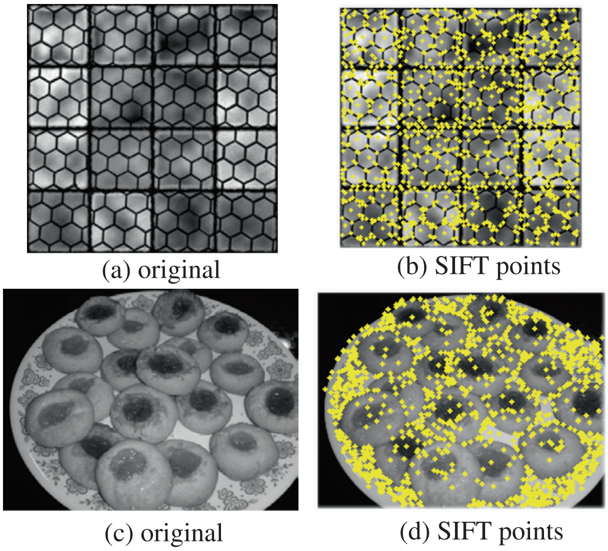


Fig. 6. Some examples of images with textures that show regularity at different scales: larger squares with inner smaller hexagons (a), cookies with the center of jam (c) and the extracted SIFT points (b,d).

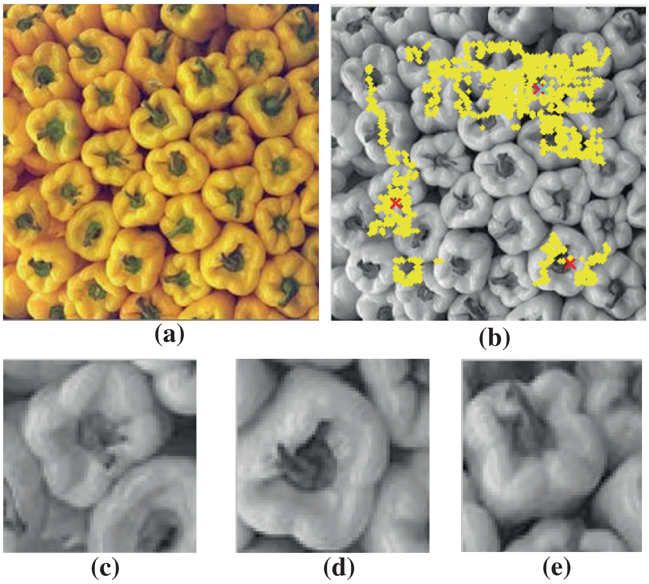


Fig. A.1. Input texture (a), clustered blocks (b), and centroids (c,d,e). the selected texel is e.

the block $C_C(k)$ whose center is nearest to its centroid (see Fig. A.1). We then compare the selected blocks by matching their keypoints, using the related descriptors information (only in case of SIFT and SURF). This allows for a strong reduction of computational effort.

If C_i and C_j are two blocks to be compared, the descriptor d_i of each keypoint $P_i \in C_i$ is compared with the vectors d_j of all the keypoints $P_j \in C_j$. For the sake of efficiency, rather than using Euclidean distance, matching is achieved through the computation of the angle between the two vectors:

$$\alpha_{ij} = \arccos(\bar{d}_i \cdot \bar{d}_j) \tag{A.1}$$

and searching for the two minimum angles:

$$\begin{aligned} \alpha_1 &= \min_{j \in B_j} \alpha_{ij} \\ \alpha_2 &= \min_{j \in B_j - j_1} \alpha_{ij} \end{aligned} \tag{A.2}$$

where j_1 is the point corresponding to α_1. To increase robustness, matches are accepted only if the ratio of the two minimum angles, α_1 and α_2, is less than a threshold. In our implementation the threshold has been set to 0.6, as suggested in (Lowe 2004). Note that, for small angles, this is a close approximation to the ratio of Euclidean distances. We further apply RANSAC (RANdom SAmple Consensus) (Fishler and Boles, 1981) to the matching points to filter outliers. Finally, we select as the representative texture element the block C_C which maximizes the average number of matches with all the other candidates. Note that the matching does not depend on rotation or scaling, as SIFT and SURF descriptors are invariant to geometrical transformations.

References

Ardizzone, E., Bruno, A., Mazzola, G., 2011. Visual saliency by keypoints distribution analysis. Image Anal. Process. ICIAP 2011, 691–699.

Ahuja, N., Todorovic, S., 2007, Extracting texels in 2.1 D natural textures. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14–20, pp. 1–8.

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., 2008. Speeded-up robust features (SURF). Comput. Vis. Image Underst. 110 (3), 346–359.

Chang J., Fisher J.W. III, 2009, Analysis of orientation and scale in smoothly varying textures. In: IEEE International Conference on Computer Vision (ICCV 2009), Kyoto, Japan, Sept 27 – Oct 4, 2009, pp. 881–888.

Brox, T., Weickert, J., 2006. A TV flow based local scale estimate and its application to texture discrimination. J. Vis. Commun. Image Represent. 2006, 1053–1073.

Elder, J.H., Zucker, S.W., 1998. Local scale control for edge detection and blur estimation. IEEE Trans. Pattern Anal. Mach. Intell. 20 (7), 699–716.

Fernandez-Valdivia, J., Garcia, J.A., Martinez-Baena, J., Fernandez-Vidal, Xose, R., 1998. The selection of natural scales in 2D images using adaptive gabor filtering. IEEE Trans. Pattern Anal. Mach. Intell. 20 (5), 458–469.

Fishler, M.A., Boles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Comm. Assoc. Comp. Mach. 24 (6), 381–395.

Forsyth, D.A., Ponce, J., 2003. Computer Vision: A Modern Approach. Prentice Hall, chapter 11 , 287–326.

Gomez, G., Marroquın, J.L., Sucar, L.E., 2000. Probabilistic estimation of local scale. In: Proc. International Conference on Pattern Recognition, vol. 3, Barcelona, Spain, 2000, pp. 798–801.

Grigorescu, S.E, Petkov, J.M.F., 2003. Texture analysis using Renyi's generalized entropies. In: International Conference on Image Processing, ICIP 2003, vol.1, pp. 241–244.

Harris, C., Stephens, M., 1988. A combined corner and edge detector. In: 4th Alvey Vision Conference, pp. 147–151.

Hays, J., Leordeanu, M., Efros, A.A., Liu, Y., 2006. Discovering texture regularity as a higher-order correspondence problem. In: 9th European Conference on Computer Vision, ECCV 2006, pp. 522–535.

Hong, B.W., Soatto, S., Ni, K., Chan, T., 2008. The scale of a texture and its application to segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8.

Jan, S.R., Hsueh, Y.C., 1998. Window-size determination for granulometrical structural texture classification. Pattern Recogn. Lett. 19 (5–6), 439–446.

Julesz, B., 1981. Textons, the elements of texture perception and their interactions. J. Nat. 290, 91–97.

Leu, J.G., 2001. On indexing the periodicity of image textures. Image Vis. Comput. 19 (13), 987–1000.

Lindeberg, T., 1998. Feature detection with automatic scale selection. Int. J. Comput. Vis. 30 (2), 79–116.

Liu, Y., Lin, W.C., Hays, J.H., 2004a. Near-regular texture analysis and manipulation. J. ACM Trans. Graphics 3, 368–376.

Liu, Y., Collins, R.T., Tsin, Y., 2004b. A computational model for periodic pattern perception based on frieze and wallpaper groups. IEEE Trans. Pattern Anal. Mach. Intell. 26 (3), 354–371.

Lizarraga-Morales, R.A., Sanchez-Yanez, R.E., Ayala-Ramirez, V., 2011. Periodicity and texel size estimation of visual texture using entropy cues. Computación y Sistemas 14 (3), 309–319.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60 (2), 91–110.

Malik, J., Belongie, S., Shi, J., Leung, T., 1999. Textons, contours and regions: cue integration in image segmentation. In: International Conference on Computer Vision, ICCV 1999, vol. 2, pp. 918–925.

Ngan, H.Y.T., Pang, G.K.H., 2009. Regularity analysis for patterned texture inspection. IEEE Trans. Autom. Sci. Eng., 131–144.

Park, M., Brocklehurst, K., Collins, R.T., Liu, Y., 2009. Deformed lattice detection in real-world images using mean-shift belief propagation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 31 (10), 1804–1816.

PSU Dataset: http://vivid.cse.psu.edu/texturedb/gallery/album13..

UIUC Dataset: http://vision.ai.uiuc.edu/~sintod/TexturesDataset.html.

Zhu, S.C., Guo, C., Wang, Y., Xu, Z., 2005. What are textons? Int. J. Comput. Vision 62 (1), 121–143.